University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Games and Full Abstraction for Nondeterministic Languages

Russell S. Harmer

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the Diploma of the Imperial College.

1999

Abstract

Nondeterminism is a pervasive phenomenon in computation. Often it arises as an emergent property of a complex system, typically as the result of contention for access to shared resources. In such circumstances, we cannot always know, in advance, exactly what will happen. In other circumstances, nondeterminism is explicitly introduced as a means of abstracting away from implementation details such as precise command scheduling and control flow. However, the kind of behaviours exhibited by nondeterministic computations can be extremely subtle in comparison to those of their deterministic counterparts and reasoning about such programs is notoriously tricky as a result. It is therefore important to develop semantic tools to improve our understanding of, and aid our reasoning about, such nondeterministic programs.

In this thesis, we extend the framework of game semantics to encompass nondeterministic computation. Game semantics is a relatively recent development in denotational semantics; its main novelty is that it views a computation not as a static entity, but rather as a dynamic process of interaction. This perspective makes the theory well-suited to modelling many aspects of computational processes: the original use of game semantics in modelling the simple functional language PCF has subsequently been extended to handle more complex control structures such as references and continuations.

To keep our study focused, we concentrate not on a complex language exhibiting contention, but on a simpler language where nondeterminism can be explicitly programmed. This language is a straightforward extension of Reynolds' Idealized Algol, *i.e.* an imperative, block-structured language with full higher-order functions. However, we emphasize that this should not be regarded as a "realistic" programming language; rather, it is a metalanguage for describing certain kinds of computational behaviour. The main technical contribution of this thesis is to develop a fully abstract game semantics for this language which takes account, not only of possible convergent behaviours, but also of possible divergent behaviour.

Declaration & Acknowledgements

Declaration The bulk of the work reported in this thesis (chapters 3 and 4) is the result of a collaboration with Guy McCusker of COGS, University of Sussex. In particular, many of the definitions and (sketches of) the proofs of chapter 4 were worked out (over coffee) by both of us together. The precise details of proofs and presentation throughout are my own work.

Acknowledgements First, I'd like to thank my supervisor, Pasquale Malacaria, for getting me interested in game semantics in the first place, for keeping me interested in it and for showing me the right way to approach research. Cheers! I'd also like to thank my second supervisor, Chris Hankin, for his advice, timely words of encouragement and for generous bankrolling on various occasions.

Next up, I must thank my friend and collaborator, Guy McCusker, for his tireless efforts in putting to rights my many misunderstandings, for a great deal of valuable advice and for being an ever-willing drinking partner. Amiante!

I've met many people during the course of my PhD who have, directly or indirectly, influenced my thinking and work. I'd particularly like to thank the following people: Roger Bailey, Steffen van Bakel, Martin Berger, Tom Burt, David Clark, Vincent Danos, Abbas Edalat, Lindsay Errington, Martin Escardo, Jim Laird, Luis Lamb, Andy Moran, Corin Pitcher, Joules Rathke, Prahladavaradan Sampath, Harold Schellinx and Julian Webster.

Finally, many thanks to my family and friends for their support, encouragement, *etc.*

Contents

Chapter 1

Introductory remarks

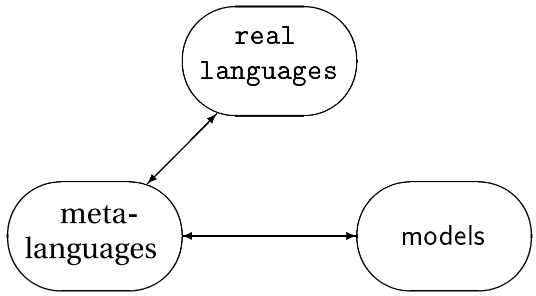
1.1 Denotational semantics

denote *v.t.* to indicate by a sign: to signify or mean (Chambers)

Preliminary remarks The traditionally, although not originally, stated aim of denotational semantics is to characterize a given notion of equivalence between the programs of a given language. The posing of this question can be traced back to early work of Scott [88] and Plotkin [81] in the context of (extensions of) the typed λ-calculus, and of Hyland [48] and Wadsworth [92] in the context of the pure, untyped λ-calculus. It has led to a vast body of research with several standard textbooks [41, 90, 93] and, more recently, a comprehensive multiple-author survey article [33].

The usual viewpoint of denotational semantics takes the language as the basic object of study. Hence, a model is *sound* iff all equivalences in the model are reflected in the language and a model is *complete* iff all equalities in the language are reflected in the model. A model satisfying both of these is often referred to as *fully abstract*, a terminology that derives from the perspective that such a model is effectively a syntax-free *definition* of the original language. A denotational model therefore offers an answer to the question: when are two programs the same? By "the same", we mean "equivalent under our given notion of equivalence". Many typical notions of equivalence are based on the *black box* view of processes: if one black box can be replaced by another in any (larger) system, with no *observable effect* on the overall behaviour of the system, then the two boxes are considered equivalent. The ubiquity of this style of definition—whether it be phrased in terms of black boxes, Morris-style contextual equivalence [71] or in some other way (*e.g.* the Turing test! [91])—means that denotational semantics is generally concerned with *static* properties of programs, such as *what* a given program computes as opposed to exactly *how* it goes about its business.

The metalanguage approach Denotational semantics is, therefore, an attempt to abstract away from certain details of programs with the hope of gaining deeper insight into the general structure of programming languages and the processes bthey describe. One of its most important methodological insights is the utility of *metalanguages* and *idealized languages* in the study of computational phenomena. The former is simply a formal language for describing the elements of a semantic "universe". The latter is a formal language equipped with an operational interpretation, *i.e.* a notion of reduction or, at the very least, a notion of equivalence between terms. In theoretical practice, these two notions are usually conflated by equipping the metalanguage with an operational interpretation, yielding what we might term an *idealized metalanguage*, *i.e.* a semantically coherent, "stripped down" programming language. A real programming language (ML, Java, *etc.*) can then be studied by translating it into this metalanguage and considering the *induced* model.



This methodology has the further advantage that well-designed metalanguages can be used in the study of many real languages: the model—and the effort involved in setting it up—can be "recycled". This technical efficiency measure has a (slightly unexpected) conceptual payoff: if we have two *a priori* quite different looking programming languages that turn out to be closely related to the same metalanguage, this suggests a hitherto unnoticed connection between the two languages: perhaps they embody essentially the same computational mechanisms, just expressed in different ways.

We see, then, that the metalanguage approach opens up at least the possibility of a semantic taxonomy of programming languages. Such a pursuit would clearly benefit from a (fairly) general semantic theory with sufficient flexibility to be able to model many different kinds of metalanguage; *e.g.* purely functional, traditional imperative Algol- or Pascal-like languages and more complex features such as lazy evaluation or the references and continuations found in Scheme. A relatively recent semantic innovation, known as *game semantics*, at least partially fulfils these requirements. This thesis is a study in the application of game-theoretic ideas to nondeterministic processes.

1.2 Game semantics

game *n.* a competitive amusement according to a system of rules (Chambers)

1.2.1 The prehistory of game semantics

Game semantics has a distinguished line of ancestry: in (rough) order of appearance, we have Scott's domain theory, Kahn-Plotkin sequentiality, stable domain theory, sequential algorithms, process calculus and the geometry of interaction. On the logical side, early work by Lorenzen on dialogue games for intuitionistic logic can also be seen as an ancestor.

Kahn-Plotkin sequentiality, stable domain theory and sequential algorithms were all developed as an attempt to characterize "sequential functionals"; this came to mean the class of functionals definable in the applied typed λ-calculus now known as PCF [81]. This search for (a "semantic" presentation of) the [65] fully abstract model of PCF gave rise to two distinct approaches: the "extensional" and the "intensional". The first—exemplified by the early work of Plotkin [81] and Berry [13], and continued by Bucciarelli & Ehrhard [18, 19]—encountered various problems of definability: the settings were, in some sense, too generous. There are functions—such as "parallel or" and Gustav—in the models not realizable by any sequential process. Interestingly, this isn't the case for the strongly stable semantics of Bucciarelli & Ehrhard which is shown in [32] to be the "extensional collapse" of the sequential algorithms model developed by Berry & Curien [14].

This latter model exemplifies the "intensional" approach by attempting to characterize what it is to *be* a PCF-definable functional by characterizing what such a functional can *do*. This model was an adaptation, to take account of PCF's higher types, of the notion of sequential function introduced by Kahn & Plotkin [54]. It eliminates the non-sequential functionals found in Plotkin's and Berry's semantics, but runs into new problems of definability: this time there are elements of the model that exhibit non-applicative behaviour, *e.g.* being sensitive to the order in which arguments are evaluated. The state of research was summarized in [24] and [74], shortly before the advent of the game models [1, 47, 72] in the early 90s.

1.2.2 Full abstraction for PCF

Game semantics views computation as an interaction between two "agents", one representing a System and the other its Environment. There are two extant formulations of this idea: the HO-style (after Hyland & Ong) and the AJM-style (after Abramsky, Jagadeesan & Malacaria); good expository introductions can be found in [2, 9, 49]. The flexibility of the approach hinges on the use of *compositional constraints* on the behaviour of (the agent representing) the System which can be related back to familiar concepts in programming.

This identification of "structural properties" of algorithms, viewed as restrictions on the System-agent's behaviour, was a decisive step towards the solving of the full abstraction problem for PCF. This observation has subsequently paved the way for a systematic approach to programming language semantics, as hinted at with Abramsky's "semantic cube" [9]. To date, fully abstract models have been presented for FPC [63], Idealized Algol [5, 7], first-order [55] and full-blown [56] continuations and for general references [3], each model imposing different constraints on the System-agent's acceptable behaviour, *i.e.* its admissible *strategies*.

Higher types & composition If a key insight of game semantics was its delineation of these structural properties, equally important was its treatment of the higher types in PCF. This had always been a thorny problem for mathematical models of such languages and, with the advent of the game models, for the first time a truly convincing account was given. This seems to relate to an intriguing aspect of game semantics: the notion of "arrow" between two games is intrinsic, not extrinsic; the notion of strategy is defined for all games—quite apart from (our intention of) it giving rise to a notion of arrow—and this is exactly what we exploit to define an arrow as a strategy on a *particular kind* of game. So we can see abstract properties of strategies as possibly (but perhaps not always) possessing computational meaning*.

Another interesting aspect of game semantics is that its notion of composition relates rather well [4, 26] to the dynamics of a well-known conceptual model of computation, namely higher-order dataflow. This complements the logical and mathematical emphasis given by Girard and others in the *Geometry of Interaction* programme of research [25,28,36–38,61,82]. Perhaps this relationship is less surprising when we observe that the notion of composition in game semantics derives from Milner's analysis of communicating processes [66,67], but hopefully it can deepen our insight into the strengths, and particularly the weaknesses, of higher-order dataflow, maybe leading to improved conceptual frameworks.

Philosophical remark Game semantics is often informally introduced by positing two *protagonists*, Opponent and Player, and saying that our observations of their behaviour playing a game (of our choice) can be construed as a description of some phenomenon of interest. It's probably a little pedantic to criticize this; but this is a rather insidious intuition as it begs the question of who is playing, why and *how*. We can duck this metaphysics simply by saying that Opponent and Player are "agents" *defined* in terms of their capabilities. In other words, we adopt an *operational* view and identify our agents with their possible actions, regarding their status as "protagonists" as a colourful, but limited, metaphor.

*For the reader familiar with some of the jargon of game semantics, we have history-freedom and innocence relating to applicative terms, the visibility condition tied to Algol-like references, various flavours of the bracketing condition relating to various flavours of control operators...

1.2.3 The strategy of game semantics

The most significant methodological insight of game semantics has been the use of *factorizations* to reduce questions of definability in larger models to those of definability in smaller models. The use of catalysts to accelerate (or trigger) chemical reactions is a good analogy: we start with the class of strategies satisfying some combination of constraints and add a "catalyst" strategy violating one (or more) of these. If we've made a good choice of catalyst, *every* strategy in the more liberal class can be generated by judicious combination with the catalyst.

It's worth noting that most of the factorizations in the game semantics literature are based on fairly classical ideas: the encoding of history with state [7]; the use of Peirce's law—*i.e.* $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$—to expand intuitionistic provability to classical provability [55]; and the use of reader-writer protocols to schedule inter-process communication [3]. Early studies of indeterminacy, *e.g.* [76–78], isolated the idea of an indeterminate *oracle* being the only required source of "randomness" to an otherwise determinate process. As we'll see later on, this idea can be used as the basis of factorizations for nondeterminism in game semantics.

Remark It is sometimes remarked that the methods of game semantics seem to be "inside out": we tend to start with a highly restricted model and expand it gradually, rather than aim for a large, widely applicable semantics. This is a fair criticism. Nevertheless, this immanent approach may well bear fruit when it comes to the perennial thorny problems of computing, such as "concurrency". Of course, this remains to be seen; only time will tell.

1.3 Nondeterminism

determinism *n.* the doctrine that all things, including the will, are determined by causes (Chambers)

1.3.1 Why study nondeterminism?

In this thesis, the emphasis we'll place on nondeterminism is as a *semantic abstraction* of certain aspects of the behaviour of a computational process. After all, it's relatively uncommon to want to ask a machine to "pick one of these tasks at random and go away and do it". However, this is not unheard of: certain *probabilistic* algorithms, such as randomized quicksort, have this flavour and are used in various applications. It's also true that adding nondeterminism to a language can offer up useful insights; for example, adding a nondeterministic choice operator to a PCF-like language allows us to observably distinguish call-by-name evaluation from call-by-need, as formalized by Hennessy & Ashcroft [44] with the distinction between *run-* and *call-*time choice.

But, more often than not, nondeterminism arises in the behaviour of a language as a reaction to possible contention for access to some shared resource. In other words, if we can use our language to describe processes comprised of more-or-less autonomous subprocesses, it's possible that the way in which the language's evaluator *schedules* execution may have a direct bearing on the outcome.

Interleaving For instance, consider the following program fragment.

$$x := 0 \ \| \ x := 1$$

This calls for consecutive execution of the two assignment statements, but also *underspecifies* the order in which this should happen: one execution could end with x containing 0, while another leaves it with 1. As a consequence, the resultant contents of x cannot be determined prior to execution.

But we can say that, after execution, x will certainly contain *either* 0 *or* 1; we thus represent *a priori* uncertainty with nondeterminism. We can understand this phenomenon by considering the assignment statements as competing for access to their "scheduling process" (in this case, the $\|$ language construct) with the overall effect that the notion of "program point" is ambiguous.

Guarded commands Another example in the same vein might be a language permitting "overlapping guards" such as Dijkstra's guarded command language [31]. Consider, for example, the following function on the integers.

$$f(x) = \begin{cases} x + 1 & \text{if } x \geq 0, \\ x - 1 & \text{if } x \leq 0. \end{cases}$$

Since the two guards overlap at $x = 0$, we have no idea—in advance—which sub-computation will be selected. So once again, nondeterminism can arise from possible ambiguity of the "program point" and, as in the previous example, the root cause of this lies in contention for a shared resource.

To keep our study focussed, we prefer not to consider a language exhibiting contention of this kind. Rather, we study a simpler language where nondeterminism can be expressed explicitly. This choice is made in the spirit of the metalanguage methodology of denotational semantics: such a language is not intended to be a realistic vehicle for programming; merely a metalanguage for the description of certain kinds of computational behaviour.

Specification? Nondeterminism also arises in the area of program specification, as an abstraction to avoid speaking too explicitly of the control-flow of the program being designed. It would be interesting to investigate whether or not the ideas in this thesis could be used to model a specification language, such as B or Gamma, thereby lending some semantic support to reasoning about properties of specifications and refinement.

1.3.2 Overview of this thesis

The thesis is organized in the following way.

- Chapter 2 introduces the languages we'll study and gathers one or two useful syntactic results together.
- Chapter 3 gives a comprehensive development of HO-style game semantics, with a particular emphasis on nondeterminism, leading to a full abstraction result for a notion of program equivalence based on safety properties.
- Chapter 4 reworks and extends the previous chapter by refining the notion of strategy to take account of liveness properties of processes, culminating in a full abstraction result for a natural notion of equivalence based on safety and liveness properties.
- Chapter 5 takes an alternate look at nondeterminism in game semantics by examining various problems that arise in the AJM-style of games, relating this to some problems in the HO-world and sketching a possible solution.
- Chapter 6 concludes and briefly describes some work in progress and some possibilities for future research.

The main contribution of the thesis comes in chapters 3–5, most particularly in chapter 4. In chapter 3, the basic elements of HO-style game semantics are presented. At the end of this chapter, we focus on one specific axis of Abramsky's *semantic cube* [7], that of *determinism*. A factorization result is established, showing that any nondeterministic strategy can be decomposed as a deterministic strategy with access to a simple oracle. A straightforward corollary of this is the full abstraction result for *may* contextual equivalence.

In chapter 4, these results are extended to a full abstraction result for *may/must* contextual equivalence. This requires a substantial reworking of the basic definitions: extra information must be added to strategies to take account, not only of possible convergent behaviour, but also of possible divergent behaviour. The main technical obstacle is to show that the revised notions of strategy and composition of strategies still gives rise to a category. In particular, it is shown that composition is only associative up to a natural notion of equivalence and, furthermore, relies crucially on the technical condition of *finite-branching*. Both of these points are strikingly analogous to similar difficulties encountered in CSP [87]. Subsequently, an alternative factorization theorem is proved, decomposing a (finite-branching) nondeterministic strategy as a deterministic strategy and an oracle, and the full abstraction result is then a corollary.

Finally, in chapter 5, we present an alternative view of game semantics and analyse certain problems from this new perspective. This chapter is less conclusive and is included to shed a little further light on some outstanding technical difficulties with applicative nondeterminism.

Chapter 2

Erratic nondeterminism

In this chapter, we formalize our operational intuitions about nondeterministic processes by focussing on two common theoretical languages: the functional PCF and the imperative Idealized Algol [84]. After completing the formality of their standard presentations, we extend them with additional syntax and evaluation rules expressing erratic nondeterminism.

2.1 PCF & Idealized Algol

2.1.1 PCF

We begin by describing the higher-order functional (meta)language PCF*. This is an applied, simply typed λ-calculus equipped with constructors appropriate for describing arithmetic, recursion and conditional branching.

Its types are defined inductively, starting from a *base* (or *ground*) type `Nat` for the (natural) numbers:

$$T \quad ::= \quad \mathtt{Nat} \mid T \to T.$$

Other base types and/or type constructors can be added to the language without undue complication—see, for example, McCusker's thesis [63] for a presentation of PCF with product, sum and recursive types, also known as FPC—but we prefer to keep things simple.

The terms, ranged over by M, N, ..., are inductively defined by:

$$\begin{array}{rcl} M & ::= & x \mid \lambda x^{T}.\, M \mid MM \mid \\ & & \mathtt{n} \mid \mathtt{succ}\, M \mid \mathtt{pred}\, M \mid \\ & & \mathtt{if0}\ M\ M\ M \mid \mathbf{Y}_{T} M. \end{array}$$

*It should perhaps be noted that the name PCF has, over time, assumed a rather generic nature: many more-or-less equivalent formulations now fall under its banner.

The x ranges over an inexhaustible supply of variables and $\mathtt{n}$ over the *numerals*, *i.e.* $n \in \mathbf{N}$. There are many essentially equivalent presentations of the language, the most obvious resulting from the (entirely natural) addition of an explicit type and terms for Boolean expressions. (In fact, we'll sometimes write down "PCF terms" containing Booleans in our informal discussions.)

Typing rules We define a ***context*** to be a finite list of pairs $x_1 : T_1, \ldots, x_n : T_n$ where the x_is are distinct variables and the T_is are types; we use $\Gamma, \Delta, \ldots$ to range over contexts. A typing rule takes the form:

$$x_1 : T_1, \ldots, x_n : T_n \vdash M : T.$$

The rules for the whole language are listed below. A term M is ***closed*** iff $\vdash M : T$ is derivable (for some type T).

variables

$$\frac{}{x_1 : T_1, \ldots, x_n : T_n \vdash x_i : T_i}\; i \in \{1, \ldots, n\}$$

lambda

$$\frac{\Gamma, x : T \vdash M : U}{\Gamma \vdash \lambda x^T.\, M : T \to U} \qquad \frac{\Gamma \vdash M : T \to U \quad \Gamma \vdash N : T}{\Gamma \vdash MN : U}$$

arithmetic

$$\frac{}{\Gamma \vdash \mathtt{n} : \mathtt{Nat}} \qquad \frac{\Gamma \vdash M : \mathtt{Nat}}{\Gamma \vdash \mathtt{succ}\, M : \mathtt{Nat}} \qquad \frac{\Gamma \vdash M : \mathtt{Nat}}{\Gamma \vdash \mathtt{pred}\, M : \mathtt{Nat}}$$

conditional & recursion

$$\frac{\Gamma \vdash M : \mathtt{Nat} \quad \Gamma \vdash N : \mathtt{Nat} \quad \Gamma \vdash L : \mathtt{Nat}}{\Gamma \vdash \mathtt{if0}\ M\ N\ L : \mathtt{Nat}} \qquad \frac{\Gamma \vdash M : T \to T}{\Gamma \vdash \mathbf{Y}_T\, M : T}$$

We'll follow the usual conventions of λ-calculus, *e.g.* identifying terms up to α-equivalence, $\lambda x.\, M$ binding x in M and dropping of type tags where ambiguity is unlikely.

Operational semantics There are two standard ways of defining an operational interpretation of PCF terms: the "small step" or *structured* operational semantics; and the "big step" or *natural* semantics. We choose here to present the latter although, since they turn out to define the same interpretation, this is largely an arbitrary choice.

The method proceeds by defining a relation, denoted by $\Downarrow$, between closed terms and ***canonical forms*** which are terms of the form:

$$V \quad ::= \quad \mathtt{n} \mid \lambda x^{T}.\, M.$$

The relation is defined inductively according to the rules that are tabulated below; the (meta)notation $M[N/x]$ denotes the capture-free substitution of N for x in M [12].

canonical form

$$\frac{}{V \Downarrow V}$$

lambda

$$\frac{M \Downarrow \lambda x.\, M' \quad M'[N/x] \Downarrow V}{MN \Downarrow V}$$

arithmetic

$$\frac{M \Downarrow \mathtt{n}}{\mathtt{succ}\, M \Downarrow \mathtt{n}+\mathtt{1}} \qquad \frac{M \Downarrow \mathtt{n}+\mathtt{1}}{\mathtt{pred}\, M \Downarrow \mathtt{n}} \qquad \frac{M \Downarrow \mathtt{0}}{\mathtt{pred}\, M \Downarrow \mathtt{0}}$$

conditional & recursion

$$\frac{M \Downarrow \mathtt{0} \quad N \Downarrow V}{\mathtt{if0}\ M\ N\ L \Downarrow V} \qquad \frac{M \Downarrow \mathtt{n}+\mathtt{1} \quad L \Downarrow V}{\mathtt{if0}\ M\ N\ L \Downarrow V} \qquad \frac{M(\mathbf{Y}\, M) \Downarrow V}{\mathbf{Y}\, M \Downarrow V}$$

Contextual equivalence For a closed term M, if there's some V such that $M \Downarrow V$ then we say that M *converges* and write $M \Downarrow$; if there is no V such that $M \Downarrow V$ then we say that M diverges and write $M \Uparrow$. As defined, PCF is a *deterministic* language. This has two effects: firstly, programs (*i.e.* closed terms of base type) can evaluate to at most one canonical form (*i.e.* numeral); secondly, convergence and divergence really are complementary: $M \Downarrow$ if, and only if, $M \not\Uparrow$.

We assume the usual notion of (typed) context. Given $M, N : T$, we define the ***contextual preorder*** by:

$$M \lesssim N \text{ iff } \forall C[-] : \mathtt{Nat}.\ C[M]\Downarrow\ \Rightarrow\ C[N]\Downarrow .$$

This relation gives rise to an evident notion of *contextual equivalence*, written $\simeq$. The (equational) full abstraction problem is precisely to give a "semantic" characterization of contextual equivalence.

2.1.2 Idealized Algol

Rather like PCF, Idealized Algol has become a generic name applied to a range of imperative metalanguages. While there are many variations possible—*e.g.* differing types, active vs. passive expressions, *etc.*—any Idealized Algol will exemplify a block-structured, higher-order programming language. We define a language IA as the following extension of PCF. We add a new base type `Com` of *commands, i.e.* operations that can change the state of a machine, and another base type `Var` of *variables, i.e.* "entities" whose state can vary over time:

$$T \quad ::= \quad \cdots \mid \mathtt{Com} \mid \mathtt{Var}$$

We now have to add some new terms.

$$\begin{aligned} M \quad ::= \quad & \cdots \mid \mathtt{skip} \mid \mathtt{assign}\ M\ M \mid \mathtt{deref}\ M \mid \\ & \mathtt{seq}_T\ M\ M \mid \mathtt{new}_T\ M \mid \mathtt{mkvar}\ M\ M \end{aligned}$$

`Skip` has type `Com` and plays a rôle akin to the numerals; operationally it is a command that always terminates, leaving the state unchanged. `Assign`, `deref` and `seq` are self-explanatory; in informal discussions, we'll often write them in the more familiar $\mathtt{x} := 8$, $!\mathtt{v}$ and $\mathtt{C}_1 ; \mathtt{C}_2$ styles. We extend `if0` by: if terms M, C_1 and C_2 respectively have types `Nat`, `Com` and `Com` in context Γ then $\Gamma \vdash \mathtt{if0}\ M\ C_1\ C_2 : \mathtt{Com}$. `New` and `mkvar` are more easily explained after presenting their typing rules.

variables

$$\frac{\Gamma \vdash M : \mathtt{Var} \quad \Gamma \vdash N : \mathtt{Nat}}{\Gamma \vdash \mathtt{assign}\ M\ N : \mathtt{Com}} \qquad \frac{\Gamma \vdash M : \mathtt{Var}}{\Gamma \vdash \mathtt{deref}\ M : \mathtt{Nat}}$$

sequencing

$$\frac{\Gamma \vdash M : \mathtt{Com} \quad \Gamma \vdash N : T}{\Gamma \vdash \mathtt{seq}_T\ M\ N : T}\ T \in \{\mathtt{Nat}, \mathtt{Com}\}$$

new & mkvar

$$\frac{\Gamma, v : \mathtt{Var} \vdash M : T}{\Gamma \vdash \mathtt{new}_T\, \lambda v.\, M : T}\; T \in \{\mathtt{Nat}, \mathtt{Com}\} \qquad \frac{\Gamma \vdash M : \mathtt{Nat} \to \mathtt{Com} \quad \Gamma \vdash N : \mathtt{Nat}}{\Gamma \vdash \mathtt{mkvar}\, M\, N : \mathtt{Var}}$$

The `new` constructor allocates a new variable. For example, if we have

$$\Gamma, \mathtt{v} : \mathtt{Var} \vdash \mathtt{v} := \mathtt{v} + 1\ ;\ !\mathtt{v}$$

then the intuitive "term" '`new v in v := v+1; !v`' is represented by:

$$\Gamma \vdash \mathtt{new}(\lambda \mathtt{v}.\, \mathtt{v} := \mathtt{v} + 1\ ;\ !\mathtt{v}).$$

The λ-abstraction of `v` reflects the binding of the newly allocated variable within its lexical scope, in this case '`v := v+1; !v`'. We assume that a newly allocated variable initially contains `0`. This design decision is reflected in the operational semantics presented below.

The final constructor is more mysterious. It appears as the consequence of an identification of `Var` with the "product" of an "acceptor" type and a "value" type. This can be thought of as an object-oriented account of a storage variable: it can be written to, via a *write method* or *acceptor* and it can be read from, via a *read method* or *dereferencing* operation. The writing method has type $\mathtt{Nat} \to \mathtt{Com}$, *i.e.* it takes a number and writes it into the variable; the reading method just has type `Nat`, it simply returns the current contents of the variable.

Semantically, `Var` will be considered to be the product of two objects. It therefore seems natural to include a "pairing" operation in the language; this is `mkvar`. The presence of this kind of term means that not everything of type `Var` actually behaves as a storage variable "should", *i.e.* we have so-called *bad variables.*

Operational semantics While the syntax and typing rules of IA are a relatively straightforward extension of those of PCF, the operational interpretation requires more of an adjustment. To take account of state, we have to consider not only the canonical form to which a term evaluates, but also any changes to the state that this evaluation may effect.

To formalize this, we define a notion of *store*. A `Var`-***context*** is a context where all the variables have type `Var`. Given a `Var`-context $\Gamma = x_1 : \mathtt{Var}, \ldots, x_n : \mathtt{Var}$, a Γ-***store*** s is a (partial) function from $\{x_1, \ldots, x_n\}$ to the natural numbers. If x_i is mapped to some number by s, we write $s(x_i)\!\downarrow$.

Given a Γ-store s, we denote the state change where variable v assumes value n by $\langle s \mid v \mapsto n\rangle$. It's not necessary that $v \in \mathtt{dom}(s)$. If it is, the state change irreversibly updates the "contents" of v. Otherwise, $\langle s \mid v \mapsto n\rangle$ is either a Γ-store or a $\{\Gamma, v\}$-store where v is "initialized" to n, depending on whether v occurs in Γ or not.

`Skip` is the only canonical form of type `Com`. The canonical forms of type `Var` are variables and terms of the form '`mkvar` $M\ N$'. We reserve ν as a metavariable ranging over variables of type `Var`.

$$V \quad ::= \quad \cdots \mid \texttt{skip} \mid \nu \mid \texttt{mkvar}\ M\ N$$

The operational semantics is now specified by relating store-term pairs

$$\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle,$$

where $\Gamma \vdash M : T$ and s, s' are Γ-stores. This can be read as "M evaluates in state s to canonical form V, transforming s to s' in the process". Note that all of M's free variables have type `Var`, *i.e.* M is closed, except for some "global variables". We amend the rules for typed λ-calculus in the obvious way.

canonical form & lambda

$$\frac{}{\langle s, V\rangle \Downarrow_\Gamma \langle s, V\rangle} \qquad \frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', \lambda x.\ M'\rangle \quad \langle s', M'[N/x]\rangle \Downarrow_\Gamma \langle s'', V\rangle}{\langle s, MN\rangle \Downarrow_\Gamma \langle s'', V\rangle}$$

We also need to redefine the operational semantics of PCF's term constructors to take stores into account. This is largely straightforward and is illustrated with a couple of examples.

PCF

$$\frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', \mathtt{n}\rangle}{\langle s, \mathtt{succ}\ M\rangle \Downarrow_\Gamma \langle s', \mathtt{n}+1\rangle} \qquad \frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', 0\rangle \quad \langle s', N\rangle \Downarrow_\Gamma \langle s'', V\rangle}{\langle s, \mathtt{if0}\ \ M\ \ N\ \ L\rangle \Downarrow_\Gamma \langle s'', V\rangle}$$

We now turn to the IA-specific terms. This is where the interesting manipulation and reference to stores occurs. Note the initialization of newly allocated variables to `0` in the rule for `new`.

variables

$$\frac{\langle s, N\rangle \Downarrow_\Gamma \langle s', \mathtt{n}\rangle \quad \langle s', M\rangle \Downarrow_\Gamma \langle s'', \nu\rangle}{\langle s, \mathtt{assign}\ M\ N\rangle \Downarrow_\Gamma \langle (s'' \mid \nu \mapsto n), \mathtt{skip}\rangle} \qquad \frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', \nu\rangle}{\langle s, \mathtt{deref}\ M\rangle \Downarrow_\Gamma \langle s', \mathtt{n}\rangle}\ s'(\nu) = n$$

sequencing

$$\frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', \mathtt{skip}\rangle \quad \langle s', N\rangle \Downarrow_\Gamma \langle s'', V\rangle}{\langle s, \mathtt{seq}_T\ M\ N\rangle \Downarrow_\Gamma \langle s'', V\rangle}$$

new

$$\frac{\langle\langle s \mid v \mapsto 0\rangle, M\rangle \Downarrow_{\Gamma,v} \langle\langle s' \mid v \mapsto n\rangle, V\rangle}{\langle s, \mathtt{new}_T\ \lambda v.\ M\rangle \Downarrow_\Gamma \langle s', V\rangle}$$

mkvar

$$\frac{\langle s, N\rangle \Downarrow_\Gamma \langle s', \mathtt{n}\rangle \quad \langle s', M\rangle \Downarrow_\Gamma \langle s'', \mathtt{mkvar}\ L\ L'\rangle \quad \langle s'', L\mathtt{n}\rangle \Downarrow_\Gamma \langle s''', \mathtt{skip}\rangle}{\langle s, \mathtt{assign}\ M\ N\rangle \Downarrow_\Gamma \langle s''', \mathtt{skip}\rangle}$$

$$\frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', \mathtt{mkvar}\ N\ N'\rangle \quad \langle s', N'\rangle \Downarrow_\Gamma \langle s'', \mathtt{n}\rangle}{\langle s, \mathtt{deref}\ M\rangle \Downarrow_\Gamma \langle s'', \mathtt{n}\rangle}$$

If M is a closed term then we abbreviate $\langle\varnothing, M\rangle \Downarrow_\varnothing \langle\varnothing, V\rangle$ with $M \Downarrow V$. An IA ***program*** is a closed term of type `Com`. Contextual equivalence is defined as for PCF, but observing with contexts of type `Com` rather than `Nat`.

Active vs. passive expressions In his original work [84] on Algol, Reynolds drew a sharp distinction between *expressions* and *commands*, the former being forbidden from changing the state (although they could "passively" read from it). In the language IA defined above, expressions are terms of type `Nat`, but the presence of term constructors such as $\mathtt{seq}_{\mathtt{Nat}}$ means that the evaluation of an expression *can* update a variable, *i.e.* we have *active*, rather than *passive*, expressions.

It is shown by O'Hearn [73] that Idealized Algol with passive expressions (*i.e.* dropping $\mathtt{seq}_{\mathtt{Nat}}$ and $\mathtt{new}_{\mathtt{Nat}}$) is a conservative extension of PCF. This is not the case for IA, as is illustrated in the following example.

Let f be the (PCF) term '$\lambda x^{\mathtt{Nat}}.\ \mathtt{if0}\ x\ (\mathtt{if0}\ (\mathtt{pred}\, x)\ 2\ \Omega)\ \Omega$' and consider placing this in the context

$$C[-] \;=\; \mathtt{new}\ v\ \mathtt{in}\,(-)(v := !v + 1; !v - 1)$$

where Ω is some divergent term like $\mathbf{Y}(\lambda x.\ x)$. As f interrogates its argument twice, the variable v gets incremented twice, so $!v$ evaluates to `1` the first time and to `2` the second time. Therefore $C[f] \Downarrow 2$ but, in PCF, $f \simeq \lambda x.\ \Omega$.

By means of a contrast, consider the term '$f(\mathtt{new}\ v\ \mathtt{in}\, v := !v + 1; !v - 1)$'. The first time f interrogates its argument, a new variable is allocated, initialized to 0, incremented and the value 0 is returned. The second time f interrogates its argument, exactly the same thing happens again. So this term must diverge, illustrating the crucial difference between copying and sharing.

2.2 Maybe, maybe not

2.2.1 Erratic choice

Erratic nondeterminism can be explained intuitively as "coin tossing": if, in the course of evaluating some expression, we were to encounter a choice between two possible continuations, we simply toss a coin to decide which way to go. The essential feature which distinguishes erratic choice from ambiguous choice is the absence of (explicit or implicit) backtracking: once the decision has been made, there's no turning back—even if it turns out that we've made a bad choice.

However, the analogy with coin tossing should not be taken to mean that we have any assumption of *fairness*: a process that repeatedly "makes a choice" between two alternatives could be implemented by an algorithm that always happens to pick the second choice. In other words, the coin can be (very) biased.

We can formalize erratic nondeterminism by extending IA with a new term constructor `or` for erratic choice. We call the extended language ***erratic* IA**, written EIA. (Obviously, there is a sublanguage, ***erratic* PCF**.) The types are no different to before so it really is just a matter of adding one constructor:

$$M \;::=\; \cdots \mid M \texttt{ or } M.$$

The typing rule for `or` is given below. Notice that we restrict nondeterminism to the base type `Nat`; we'll see later that this results in no loss of expressive power.

choice

$$\frac{\Gamma \vdash M : \texttt{Nat} \qquad \Gamma \vdash N : \texttt{Nat}}{\Gamma \vdash M \texttt{ or } N : \texttt{Nat}}$$

Given the above discussion, it's not too hard to see how to extend the operational semantics to take account of `or`.

choice

$$\frac{\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle}{\langle s, M \texttt{ or } N\rangle \Downarrow_\Gamma \langle s', V\rangle} \qquad \frac{\langle s, N\rangle \Downarrow_\Gamma \langle s', V\rangle}{\langle s, M \texttt{ or } N\rangle \Downarrow_\Gamma \langle s', V\rangle}$$

2.2.2 Contextual equivalences

In erratic IA or PCF, it's no longer the case that a term M of type `Nat` evaluates to at most one numeral. This means that, if $M \Downarrow V$ then the best we can say is that M *may* evaluate to canonical form V.

We've already seen that IA is not a conservative extension of PCF. It's also the case that PCF extended with `or` isn't a conservative extension of PCF. We start with the same term f, ***i.e.*** '$\lambda x^{\mathtt{Nat}}.\,\mathtt{if0}\ x\ (\mathtt{if0}\ (\mathtt{pred}\,x)\ 2\ \Omega)\ \Omega$'. In PCF, this is contextually equivalent to '$\lambda x.\,\Omega$' but, in the presence of observable nondeterminism, we can apply this term to '`0 or 1`' with a *possibility* of convergence to '2'. However, this example also illustrates the other phenomenom associated with erratic choice: while '`f(0 or 1)`' may converge, it also *may diverge*.

This potential unreliability vastly complicates reasoning about nondeterministic processes. It has the effect of destroying the complementary nature of convergence and divergence that we observed in PCF and IA, leading us to adopt two distinct notions of equivalence: the coarser ***may*** contextual equivalence and the more discriminating ***may/must*** contextual equivalence.

May contextual equivalence Given an EIA program M, we say that "M may converge" iff $M \Downarrow \mathtt{skip}$ and write $M \Downarrow^{\mathrm{may}}$ to denote this. We then define the ***may*** contextual preorder by:

$$M \lesssim_{\mathrm{may}} N \ \text{ iff } \forall C[-] : \mathtt{Com}.\ C[M] \Downarrow^{\mathrm{may}} \Rightarrow C[N] \Downarrow^{\mathrm{may}} .$$

We complete the definition by saying that M and N are ***may*** contextually equivalent iff $M \lesssim_{\mathrm{may}} N$ and $N \lesssim_{\mathrm{may}} M$, denoting this relation by $\simeq_{\mathrm{may}}$.

This notion of equivalence completely disregards potential unreliability of the kind we saw in the above example. So, for example,

$$(\lambda x^{\mathtt{Nat}}.\,\mathtt{if0}\ x\ (\mathtt{if0}\ (\mathtt{pred}\,x)\ 2\ \Omega)\ \Omega)(0\ \mathtt{or}\ 1)\ \simeq_{\mathrm{may}}\ 2.$$

Must-convergence in PCF As the previous example suggests, may contextual equivalence is not entirely satisfactory for reasoning about nondeterministic programs. Ideally we would also like to be able to take into account the question of whether (or not) a term is reliable. Such a notion of equivalence would clearly not relate the two terms above.

We first of all define a predicate on closed PCF terms, that of ***must convergence***. (We could have worked with a predicate of *may divergence* but it's technically a little simpler to work with must convergence and, since the two predicates are complementary, it makes no difference whatsoever.) The predicate is inductively defined according to the following rules.

canonical form

$$\frac{}{V \Downarrow^{\mathrm{must}}}$$

In order for an application 'MN' to be ensured of convergence, it must be the case that the "function" M must converge and that, for anything that it may evaluate to, substituting in the "argument" N must also converge.

lambda

$$\frac{M \Downarrow^{\text{must}} \qquad M \Downarrow \lambda x.\, M' \;\Rightarrow\; M'[N/x] \Downarrow^{\text{must}}}{MN \Downarrow^{\text{must}}}$$

The rules for arithmetic, conditionals and recursion are straightforward.

arithmetic & recursion

$$\frac{M \Downarrow^{\text{must}}}{\texttt{succ}\ M \Downarrow^{\text{must}}} \qquad \frac{M \Downarrow^{\text{must}}}{\texttt{pred}\ M \Downarrow^{\text{must}}} \qquad \frac{M(\mathbf{Y}\, M) \Downarrow^{\text{must}}}{\mathbf{Y}\, M \Downarrow^{\text{must}}}$$

conditional

$$\frac{M \Downarrow^{\text{must}} \qquad M \Downarrow \texttt{0} \;\Rightarrow\; N \Downarrow^{\text{must}} \qquad M \Downarrow \texttt{n} + \texttt{1} \;\Rightarrow\; L \Downarrow^{\text{must}}}{\texttt{if0}\ M\ N\ L \Downarrow^{\text{must}}}$$

The crucial rule is for erratic choice; in order to guarantee convergence of a term 'M `or` N' it must be the case that *both* M and N must converge.

choice

$$\frac{M \Downarrow^{\text{must}} \qquad N \Downarrow^{\text{must}}}{M\ \texttt{or}\ N \Downarrow^{\text{must}}}$$

Recursion & infinite branching One pragmatic consequence of adding erratic choice to PCF is that the fixpoint combinator becomes non-trivial at ground types. For example, the term '$\mathbf{Y}_{\texttt{Nat}}(\lambda x.\, \texttt{0}\ \texttt{or}\ \texttt{succ}\ x)$' may converge to *any* numeral. However, it's also the case that recursion doesn't provide a *reliable* route to countable nondeterminism: the above term may also diverge since—at least without any notion of "fairness"—there's no reason why the left-hand side base case should ever be chosen.

This result can be proved syntactically; see, for example, [45]. In chapter 4, we'll use a category of "finitely branching" strategies to model erratic PCF and IA; it will then follow from the fact that this provides sound models that no reliable countable nondeterminism can be programmed with the binary erratic choice operator.

M&M contextual equivalence We now extend the must-convergence predicate to EIA. It is inductively defined on store-term pairs $\langle s, M\rangle$ where Γ is a Var-context, s is a Γ-store and $\Gamma \vdash M : T$.

The rules for PCF extend in the obvious way; for example, here's application:

lambda

$$\frac{\langle s, M\rangle \Downarrow_{\Gamma}^{\text{must}} \qquad \langle s, M\rangle \Downarrow_{\Gamma} \langle s', \lambda x.\, M'\rangle \Rightarrow \langle s', M'[N/x]\rangle \Downarrow_{\Gamma}^{\text{must}}}{\langle s, MN\rangle \Downarrow_{\Gamma}^{\text{must}}}$$

The rules for the novel terms of IA are as follows.

sequencing

$$\frac{\langle s, M\rangle \Downarrow_{\Gamma}^{\text{must}} \qquad \langle s, M\rangle \Downarrow_{\Gamma} \langle s', \mathtt{skip}\rangle \Rightarrow \langle s', N\rangle \Downarrow_{\Gamma}^{\text{must}}}{\langle s, \mathtt{seq}\ M\ N\rangle \Downarrow_{\Gamma}^{\text{must}}}$$

assignment

$$\frac{\langle s, N\rangle \Downarrow_{\Gamma}^{\text{must}} \quad \langle s, N\rangle \Downarrow \langle s', \mathtt{n}\rangle \Rightarrow (\langle s', M\rangle \Downarrow_{\Gamma}^{\text{must}} \wedge (\langle s', M\rangle \Downarrow \langle s'', \mathtt{mkvar}\ L\ P\rangle \Rightarrow \langle s'', L\mathtt{n}\rangle \Downarrow_{\Gamma}^{\text{must}}))}{\langle s, \mathtt{assign}\ M\ N\rangle \Downarrow_{\Gamma}^{\text{must}}}$$

dereferencing

$$\frac{\langle s, M\rangle \Downarrow_{\Gamma}^{\text{must}} \quad \langle s, M\rangle \Downarrow \langle s', v\rangle \Rightarrow s'(v)\downarrow \quad \langle s, M\rangle \Downarrow \langle s', \mathtt{mkvar}\ L\ P\rangle \Rightarrow \langle s', P\rangle \Downarrow_{\Gamma}^{\text{must}}}{\langle s, \mathtt{deref}\ M\rangle \Downarrow_{\Gamma}^{\text{must}}}$$

new

$$\frac{\langle \langle s \mid v \mapsto 0\rangle, M\rangle \Downarrow_{\Gamma, v}^{\text{must}}}{\langle s, \mathtt{new}\ \lambda v.\, M\rangle \Downarrow_{\Gamma}^{\text{must}}}$$

We define the ***M&M*** contextual preorder by:

$$M \lesssim_{\text{m\&m}} N \ \text{ iff } \ \forall C[-] : \mathtt{Com}.\ (C[M] \Downarrow^{\text{may}} \Rightarrow C[N] \Downarrow^{\text{may}} \ \wedge \ C[M] \Downarrow^{\text{must}} \Rightarrow C[N] \Downarrow^{\text{must}})$$

We complete the definition of contextual equivalence by setting $M \simeq_{\text{m\&m}} N$ iff $M \lesssim_{\text{m\&m}} N$ and $N \lesssim_{\text{m\&m}} M$.

2.3 Some syntactic results

First of all, we note a couple of lemmas showing that the higher types in EIA are "logical" in the sense that no "computation", *i.e.* δ-rule-like reduction, is going on.

Suppose that $\Gamma \vdash M : T$ where Γ is a `Var`-context and T is either `Var` or of the form $T_1 \to T_2$.

Lemma 2.3.1 If $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$ then $s = s'$.

Proof A simple induction on the derivation that $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$. ■

Lemma 2.3.2 If $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$ then this V is unique and $\langle s, M\rangle \Downarrow_\Gamma^{\text{must}}$.

Proof Uniqueness of V is proved by an easy induction on the derivation that $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$. A similar induction yields the second part; we illustrate this with the case where M is an application:

$$\frac{\langle s, M_1\rangle \Downarrow_\Gamma \langle s', \lambda x.\, M_1'\rangle \quad \langle s', M_1'[M_2/x]\rangle \Downarrow_\Gamma \langle s'', V\rangle}{\langle s, M_1M_2\rangle \Downarrow_\Gamma \langle s'', V\rangle.}$$

By the inductive hypothesis, $\langle s, M_1\rangle \Downarrow_\Gamma^{\text{must}}$ and $\lambda x.\, M_1'$ is the only canonical form to which $\langle s, M_1\rangle$ can evaluate.

Also by the inductive hypothesis, $\langle s', M_1'[M_2/x]\rangle \Downarrow_\Gamma^{\text{must}}$ and we deduce:

$$\frac{\langle s, M_1\rangle \Downarrow_\Gamma^{\text{must}} \quad \langle s', M_1'[M_2/x]\rangle \Downarrow_\Gamma^{\text{must}}}{\langle s, M_1M_2\rangle \Downarrow_{\Gamma.}^{\text{must}}}$$

■

In other words, there's neither "stateful behaviour" nor "nondeterministic effects" at higher types.

Syntactic approximation If $\Gamma \vdash M : T \to T$ then we inductively define, for all $k \in \mathbf{N}_0$, a term $\Gamma \vdash \mathbf{Y}_T^k(M) : T$ as follows.

$$\begin{array}{rcl} \mathbf{Y}_T^0(M) & = & \Gamma \vdash \Omega_T : T \\ \mathbf{Y}_T^{k+1}(M) & = & \Gamma \vdash M(\mathbf{Y}_T^k(M)) : T \end{array}$$

In order to manipulate this intuitive notion of a "finite unwinding" of a recursion, we define the relation of *syntactic approximation* between terms $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$, denoted $M \prec_{\Gamma;T} N$. This is the least reflexive relation satisfying

$$\frac{M \prec_{\Gamma;T\to T} N}{\mathbf{Y}_T^k(M) \prec_{\Gamma;T} \mathbf{Y}_T(N)}\ k \in \mathbf{N}_0$$

plus additional rules expressing the (pre)congruence of $\prec_{\Gamma;T}$ with respect to our term forming constructions, *e.g.* if $M \prec_{\Gamma;\texttt{Nat}} N$ then $\texttt{succ}\ M \prec_{\Gamma;\texttt{Nat}} \texttt{succ}\ N$.

Lemma 2.3.3 If $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$ are terms of EIA such that $M \prec_{\Gamma;T} N$ and $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$ then $\langle s, N\rangle \Downarrow_\Gamma \langle s', W\rangle$ for some W such that $V \prec_{\Gamma;T} W$.

Proof A straightforward induction on the derivation that $\langle s, M\rangle \Downarrow \langle s', V\rangle$.

For example, suppose that $M = \mathbf{Y}^k(M')$ so that $N = \mathbf{Y}(N')$. If $\langle s, M\rangle \Downarrow \langle s', V\rangle$ then, by definition, $\langle s, M'(\mathbf{Y}^{k-1}(M'))\rangle \Downarrow \langle s', V\rangle$ and, by the inductive hypothesis, $\langle s, N(\mathbf{Y}(N))\rangle \Downarrow \langle s', W\rangle$ for some W such that $V \prec W$. But then, $\langle s, \mathbf{Y}(N)\rangle \Downarrow \langle s', W\rangle$ and we're done. ■

Lemma 2.3.4 If $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$ are terms of EIA such that $M \prec_{\Gamma;T} N$ and $\langle s, M\rangle \Downarrow^{\text{must}}_\Gamma$ then $\langle s, N\rangle \Downarrow^{\text{must}}_\Gamma$.

Proof By induction on the derivation that $\langle s, M\rangle \Downarrow^{\text{must}}$. We consider a couple of the key cases.

If $M = M_1 M_2$ so that $N = N_1 N_2$ and $M_1 \prec N_1$ and $M_2 \prec N_2$ then we have the following rule.

$$\frac{\langle s, M_1\rangle \Downarrow^{\text{must}} \qquad \langle s, M_1\rangle \Downarrow \langle s', \lambda x.\, M_1'\rangle \Rightarrow M_1'[M_2/x] \Downarrow^{\text{must}}}{\langle s, M_1 M_2\rangle \Downarrow^{\text{must}}}$$

So, by the inductive hypothesis, $\langle s, N_1\rangle \Downarrow^{\text{must}}$ and, by the previous lemmas, there's a unique $\lambda x.\, N_1'$ such that $\langle s, N_1\rangle \Downarrow \langle s, \lambda x.\, N_1'\rangle$ and $M_1' \prec N_1'$. By the inductive hypothesis, $\langle s, N_1'[N_2/x]\rangle \Downarrow^{\text{must}}$ so $\langle s, N\rangle \Downarrow^{\text{must}}$.

If $M = M_1 \texttt{ or } M_2$ so that $N = N_1 \texttt{ or } N_2$ and $M_1 \prec N_1$ and $M_2 \prec N_2$ then, by the inductive hypothesis, $\langle s, M_1\rangle \Downarrow^{\text{must}}$ and $\langle s, M_2\rangle \Downarrow^{\text{must}}$ and so $\langle s, N_1\rangle \Downarrow^{\text{must}}$ and $\langle s, N_2\rangle \Downarrow^{\text{must}}$ and hence $\langle s, N\rangle \Downarrow^{\text{must}}$. ■

Chapter 3

HO-style game semantics

We now proceed to formalize our intuitions of game semantics; see also [2, 49] for introductory accounts. The games presented in this chapter are commonly known as HO-style games (after Hyland-Ong). We proceed by defining the basic notions of "game", a "play" of a game and a "strategy" on a game. This leads to a general categorical framework, within which we construct an adequate model of an Algol-like language extended with a binary "erratic choice" operator, *cf.* [45].

We then establish a full abstraction result, yielding a characterization of may contextual equivalence, by proving a *definability* theorem: all compact strategies in the model denote some term of the language. This result relies on a *factorization* theorem which says, roughly, that any nondeterministic behaviour can be simulated by a deterministic process that has access to a random number generator. In fact, for all the processes of interest, it suffices that they have a "coin-tossing" capability (*i.e.* finite nondeterminism suffices).

A few remarks are probably in order. Although the presentation of HO-style game semantics given here has been much influenced by that of [63], there are also many differences. This reflects a general evolution of the basic definitions of game semantics over the last two or three years, a process that has been driven by concrete developments in presenting games models for a variety of different languages. This progress has continually emphasized that, whenever possible, restrictions should be placed on strategies rather than games. This has led to a general migration of conditions, such as *visibility* and *bracketing*, from being constraints on plays to being *properties* of plays that can be used to restrict the behaviours of strategies. This approach allows a very general development of our basic theory of games; more restricted settings can be introduced as we please, simply by constraining strategies in various ways. If these constraints can be shown to be orthogonal to one another, this provides a very flexible framework in which to analyse differing aspects of computational processes. The formulation of games that we give has been strongly influenced by these observations and can hopefully be more readily digested as a result.

3.1 Arenas & strategies

3.1.1 Arenas

The basic idea of HO-style game semantics is that a "game" is played in an *arena* which can be thought of as a general "playing area" setting out certain basic rules and conventions. Our formal definition is as follows.

Definition An ***arena*** A is a triple $\langle M_A, \lambda_A, \vdash_A \rangle$ where:

- M_A is a countable set of ***moves***. A move is best thought of as a "blob" with no internal structure; it is only recognizable as a move (of A) because it lives in M_A.

- $\lambda_A : M_A \to \{\forall, \exists\} \times \{?, !\}$ is a ***labelling*** function, where $\forall$, $\exists$, ? and ! are four distinguishable dummy symbols. We write $\lambda_A^{\forall\exists}$ for λ_A composed with first projection and $\lambda_A^{?!}$ for composition with second projection. So, considering $\lambda_A^{\forall\exists}$ and $\lambda_A^{?!}$ as set-theoretic functions, $\lambda_A = \langle \lambda_A^{\forall\exists}, \lambda_A^{?!} \rangle$, the pairing in **Set**.

  Let $O_A = \{m \in M_A \mid \lambda_A^{\forall\exists}(m) = \forall\}$ and $P_A = \{m \in M_A \mid \lambda_A^{\forall\exists}(m) = \exists\}$. Since λ_A is a total function, $\{O_A, P_A\}$ is a partition of M_A. The sets O_A and P_A *define* the "protagonists" ***Opponent*** and ***Player*** in the arena A. Accordingly, if $m \in O_A$ we say that it's an O-move (of A); otherwise it's a P-move (of A).

  We can use $\lambda_A^{?!}$ to define another partition of M_A, written as $\{Q_A, A_A\}$, by setting $Q_A = \{m \in M_A \mid \lambda_A^{?!}(m) = ?\}$ and $A_A = \{m \in M_A \mid \lambda_A^{?!}(m) = !\}$. If $m \in Q_A$ we say it's a ***Question***; otherwise it's an ***Answer***.

  We denote by $\overline{\lambda}_A$ the alternative labelling that "swaps" Opponent and Player, *i.e.* $\lambda_A^{\forall\exists}(m) = \forall$ if, and only if, $\overline{\lambda}_A^{\forall\exists}(m) = \exists$, but which leaves Questions and Answers unchanged.

- $\vdash_A$ is an ***enabling*** relation picking out a subset of $M_A + (M_A \times M_A)$. More formally, it's a binary relation on M_A satisfying

  (**e1**) if $n \vdash_A m$ and $n \neq m$ then $\lambda_A^{\forall\exists}(n) \neq \lambda_A^{\forall\exists}(m)$;

  (**e2**) if $m \vdash_A m$ then ($\lambda_A(m) = \forall ?$ and if $n \neq m$ then $n \not\vdash_A m$);

  (**e3**) if $n \vdash_A m$ and $\lambda_A^{?!}(m) = !$ then $\lambda_A^{?!}(n) = ?$.

The enabling relation introduces *causality* into an arena. When a move is played, this (potentially) enables other moves to subsequently be made. Indeed, most moves can't be played unless such an enabling move has already been made. The exceptions to this rule are the *self-enabling*, or ***initial***, moves such that $m \vdash_A m$.

Axiom (**e1**) says that if one move enables some other move then one is an O-move and the other a P-move. Axiom (**e2**) amounts to the requirement that Opponent can be partitioned into a set of *initial* moves and a set of *non-initial* moves. (**e3**) says that Answers are always enabled by Questions, but doesn't rule out Questions being enabled by Answers.

Examples of arenas The simplest arena is $\langle\varnothing,\varnothing,\varnothing\rangle$ which we denote by $\mathbf{1}$. The next simplest arena has one initial O-move q and is denoted by $\perp$.

Given a countable set X, we build the ***flat arena*** over X with a single Opponent Question q, representing a request for an element of X, and one Player Answer x for each $x \in X$. The O-move q is initial and enables all the P-moves.

We'll be particularly interested in three flat arenas, being those generated by a one-point, a two-point and a countably infinite set. These arenas are denoted by $\mathbf{C}$, $\mathbf{B}$ and $\mathbf{N}$ respectively; we conventionally fix their generating sets to be $\{a\}$, $\{\mathrm{tt}, \mathrm{ff}\}$ and the set $\mathbf{N}_0 = \{0, 1, 2, \ldots\}$ of non-negative integers. The arena $\perp$ can be thought of as the flat arena over $\varnothing$.

A little notation Let Σ be some alphabet; then we denote by $\Sigma^\star$ the set of all finite strings over Σ. Given two such strings s and t, we write $s \preceq t$ for the subsequence ordering and $s \sqsubseteq t$ for the (reflexive) prefix ordering; the least element of the latter, namely the unique string of length 0, is written as ε. The concatenation of s and t is written as $s \cdot t$ or st whenever we can get away with it; we also tend to identify a symbol $a \in \Sigma$ with the string a of length 1. The length of a finite string s is denoted by $|s|$ and its ith symbol (starting from 1) is written as s_i.

Given some subset $\Sigma' \subseteq \Sigma$, the restriction of s to Σ'—*i.e.* the subsequence of s obtained by removing any symbols from s that aren't in Σ'—is denoted by $s \upharpoonright \Sigma'$. The complementary notion, where we remove all symbols of s that *are* in Σ', is denoted by $s \restriction \Sigma'$. If a is some occurrence of a symbol in s then we write $a \in s$ and denote by $s_{\leqslant a}$ the prefix of s up to and including a; the prefix of s up to, but not including, a is denoted by $s_{<a}$.

3.1.2 Justified strings & legal plays

We represent the "history of play" in a game with sequences of moves subject to certain conditions. First of all, we define the basic notion of *justified string*; these strings are then further constrained by the *alternation condition*, giving rise to the class of *legal plays*.

Definition A ***justified string*** s in an arena A is a finite string over the alphabet M_A together with a pointer from each (occurrence of a) non-initial move $m \in s$ to an (occurrence of an) earlier move $n \in s_{<m}$ such that $n \vdash_A m$. We say that m is ***justified*** by n or that n ***justifies*** m and write J_A for the set of all justified strings of A. It's easy to see that the first move in any justified string must be an initial move.

It's important to realize that justification refers to particular instances of moves: given an occurrence of a move m occurring in a justified string s, there may be several occurrences of moves in $s_{<m}$ that *enable* m but the unique *justifying* move is given by the pointer structure. This observation is important for the following crucial definition.

Definition The ***Player view*** (or just the P***-view***) $\ulcorner s \urcorner$ of a non-empty string $s \in J_A$ is defined inductively as follows.

$$\begin{aligned}
\ulcorner sm \urcorner &= m, && \text{if } m \text{ is initial;}\\
\ulcorner sntm \urcorner &= \ulcorner s \urcorner nm, && \text{if } m \text{ is an O-move and}\\
& && n \text{ justifies } m;\\
\ulcorner sm \urcorner &= \ulcorner s \urcorner m, && \text{if } m \text{ is a P-move.}
\end{aligned}$$

Suppose that $sm \in J_A$ where m is a P-move. We say that sm satisfies ***Player visibility at*** m (usually we'll just say that sm is P***-vis at*** m) iff the justifier of m occurs in $\ulcorner s \urcorner$. If $s \in J_A$ then s satisfies ***Player visibility*** (usually we'll just say that s satisfies P***-visibility*** or even just that it's P***-vis***) iff s is P-vis at m for each (occurrence of a) P-move $m \in s$. If $s \in J_A$ satisfies Player visibility then its Player view is itself a justified string. But, beware! The converse isn't true: it's quite possible for the P-view of a justified string s to be justified even if s violates Player visibility.

So the P-view is determined by repeatedly "stepping over" P-moves and "chasing back" pointers from O-moves until an initial move is reached. We can invert this idea so that we step over O-moves and chase back pointers from P-moves until we run out of moves. This gives the following notion of the ***Opponent view*** (or just the O***-view***) $\llcorner s \lrcorner$ of a justified string $s \in J_A$.

$$\begin{aligned}
\llcorner \varepsilon \lrcorner &= \varepsilon\\
\llcorner sntm \lrcorner &= \llcorner s \lrcorner nm, && \text{if } m \text{ is a P-move and}\\
& && n \text{ justifies } m;\\
\llcorner sm \lrcorner &= \llcorner s \lrcorner m, && \text{if } m \text{ is an O-move.}
\end{aligned}$$

Again, suppose that $sm \in J_A$ but this time let m be a non-initial O-move of A; sm satisfies ***Opponent visibility at*** m (or is O***-vis at*** m) iff the justifier of m occurs in $\llcorner s \lrcorner$ and $s \in J_A$ satisfies ***Opponent visibility*** (or is O-vis) iff it's O-vis at m for each (occurrence of an) O-move $m \in s$. Note that the empty string ε is vacuously O-vis. Opponent visibility suffices to guarantee that the O-view of a justified string is also justified.

These notions of Player and Opponent view are closely related to branches in a Böhm tree. They can also be seen as a convenient abstraction of certain coded information (that remains implicit) in the AJM-style model of PCF [26].

Aside There is an apparent asymmetry between the notions of P-view and O-view in that a P-view contains exactly one occurrence of an initial move while an O-view can contain many occurrences of initial moves. This is a direct consequence of axiom (**e2**) which insists that all initial moves are O-moves so that the process of "tracing back" a view can only terminate by reaching an initial move if we're "stepping over" P-moves. If Player was allowed to have initial moves, a P-view could contain many initial P-moves while an O-view could only have one.

Legal plays A justified string s in arena A is a ***legal play*** (or just a ***play***) of A iff Opponent and Player strictly alternate, *i.e.* $s_i \in O_A$ if, and only if, i is odd. We write L_A for the set of all legal plays in arena A and L_A^{even} and L_A^{odd} for the even- and odd-length plays respectively. From now on, we'll only concern ourselves with legal plays.

For example, there is only one legal play in the arena $\mathbf{1}$, namely ε. In the flat arena over X, (even-length) legal plays have the form $\mathsf{q}x_1\mathsf{q}x_2\ldots\mathsf{q}x_n$ where each $x_i \in X$.

Suppose that $s \in L_A$ and let m be some (occurrence of a) move in s. We say that m is ***hereditarily justified*** by an (occurrence of an) initial move n iff following the justification pointers back from m leads us to n. We denote by $s \upharpoonright n$ the subsequence of s consisting of all those (occurrences of) moves hereditarily justified by n. This not(at)ion evidently generalizes to $s \upharpoonright I$ where I is a set of (occurrences of) initial moves occuring in s.

Suppose that $sm \in L_A^{\mathsf{odd}}$ so that m is an O-move. The ***current thread*** $\lceil sm \rceil$ of sm is defined to be $s \upharpoonright n$ where n is the hereditary justifier of m. Now, consider $sm \in L_A^{\mathsf{even}}$. We say that sm is ***well-threaded at*** m iff the justifier of m occurs in $\lceil s \rceil$. In this case, we define $\lceil sm \rceil = \lceil s \rceil m$. A play $s \in L_A$ is ***well-threaded*** iff it's well-threaded at m for each (occurrence of a) P-move in s. The current thread of a play is always a justified string. The well-threading condition guarantees legality: if m_1 and m_2 are two (occurrences of) O-moves in $\lceil s \rceil$, we know that there must be an (occurrence of a) P-move n immediately following m_1 in s (assuming m_1 occurs before m_2) and well-threadedness says that n also occurs in $\lceil s \rceil$. A similar argument shows that between any two P-moves in $\lceil s \rceil$, there's an O-move, so we're done. This argument generalizes to show that $s \upharpoonright I$ is legal for a collection I of initial occurrences.

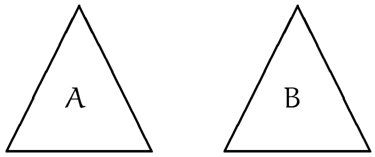
We also observe that if $s \in L_A$ is P-vis (and non-empty) then all the moves in $\ulcorner s \urcorner$ are hereditarily justified by the same initial occurrence. This is trivial if s ends with an initial occurrence m and follows immediately by P-visibility and the inductive hypothesis if s ends with a P-move. Otherwise, we have that $s = s_1 n s_2 m$ where n justifies m. So $\ulcorner s \urcorner = \ulcorner s_1 \urcorner n m$ and by P-vis, n's justifier occurs in $\ulcorner s_1 \urcorner$ and we conclude by invoking the inductive hypothesis. So the P-view is a subsequence of the current thread, a *potted history.*

3.1.3 Constructions on arenas

Product Given two arenas A and B, we can combine them, as follows, to form a new arena denoted by $A \times B$.

- $M_{A\times B} = M_A + M_B$;
- $\lambda_{A\times B} = [\lambda_A, \lambda_B]$, the copairing in **Set**;
- $n \vdash_{A\times B} m$ iff $n \vdash_A m$ or $n \vdash_B m$.

Intuitively, $A \times B$ is an arena consisting of A and B placed side-by-side, as in the picture below.

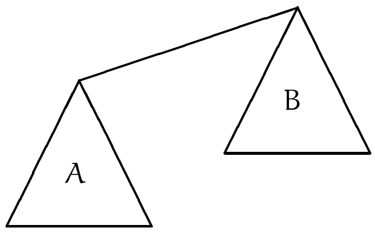


There is no "interaction" between the two ***constituent*** arenas: the initial moves of $A \times B$ are those of A and those of B and the enabling is likewise directly inherited. The "empty" arena **1** is a unit for this constructor.

Arrow An alternative way to combine two arenas makes one constituent subservient to the other: the initial moves of the combined arena are those of the privileged constituent; the new enabling is derived in the obvious way except that the initial moves of the subservient arena are now enabled by the initial moves of the privileged one.

- $M_{A\Rightarrow B} = M_A + M_B$;
- $\lambda_{A\Rightarrow B} = [\overline{\lambda}_A, \lambda_B]$;
- $n \vdash_{A\Rightarrow B} m$ iff $n \vdash_B m$ or ($n \neq m$ and $n \vdash_A m$) or ($n \vdash_B n$ and $m \vdash_A m$).

This can be visualized with the following picture where the line connecting A and B is intended to represent the enabling of the initial moves of A by the initial moves of B.

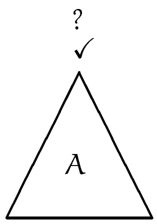


Note that, since initial moves of A are enabled by initial moves of B in $A \Rightarrow B$, an O-move of A must be considered to be a P-move of $A \Rightarrow B$; hence the difference between the labelling of $A \times B$ and $A \Rightarrow B$. Another way to think of this is to see Opponent in $A \Rightarrow B$ as being $P_A + O_B$.

Lifting Our final constructor is analogous to the familiar "lifting" operation on a preordered set. It operates on one arena, simply adding an initial "protocol" consisting of two new moves, ? and ✓, before evolving into the underlying arena. The formal definition is as follows.

- $M_{A_\perp} = \{?, \checkmark\} + M_A$;
- $\lambda_{A_\perp}(\mathsf{inl}(?)) = \forall ?$, $\lambda_{A_\perp}(\mathsf{inl}(\checkmark)) = \exists !$ and $\lambda_{A_\perp}(\mathsf{inr}(m)) = \lambda_A(m)$;
- $n \vdash_{A_\perp} m$ iff ($n = ?$ and ($m = ?$ or $m = \checkmark$)) or ($n = \checkmark$ and $m \vdash_A m$) or ($n \neq m$ and $n \vdash_A m$).

Once again, this is best viewed pictorially.



Notice that Player's Answer ✓ enables the (no longer) initial moves of A. This can be generalized to form the ***disjoint sum*** of an indexed collection of arenas; see [63] for more details. We'll make no use of this more general constructor so we stick with the unary case.

Aside Once we have the $\times$ and $\Rightarrow$ constructors, we can think of using them, in conjunction with the arena $\perp$, to build more complex arenas. For example, the arena **B** can be seen as being "rather like" $(\perp \times \perp) \Rightarrow \perp$. It's not the same though: any arena built by combining occurrences of $\perp$ with $\times$ and $\Rightarrow$ can only contain Questions—in other words, the Question/Answer aspect of moves is ignored. So, if we were interested in making a distinction between "questions" and "answers", we would have to reintroduce this somewhere else, perhaps by enriching the enabling relation. This would mean that the notions of "question" and "answer" would no longer be innate properties of moves; instead they would be reflected as certain *conventions* between the Opponent and Player of that particular arena. Note that lifting is *definable* in this setting: given an arena A, its lifting is $(A \Rightarrow \perp) \Rightarrow \perp$.

3.1.4 Strategies

A ***strategy*** σ on an arena A (usually written $\sigma : A$) is defined to be a set of even-length legal plays of A satisfying

(**s1**) $\varepsilon \in \sigma$;

(**s2**) if $sab \in \sigma$ then $s \in \sigma$.

Axiom (**s1**) is equivalent to requiring strategies to be non-empty. In fact, allowing the empty set would cause no problems; it would probably correspond to some kind of "stuck" or "deadlocked" process. But for the purposes of this thesis, we'll concentrate on processes that always *accept* an appropriately typed input, even if they'll never (visibly) respond to that stimulus. In this setting, the only possible source of non-termination is "infinite looping" and we make no (formal) distinction between "deadlock" and "livelock". Axiom (**s2**) is a convenient assumption to avoid redundancy in the model. Relaxing it leads to various (often bizarre) phenomena arising and we leave this direction open for future work.

The intuition behind the definition of strategy is that it's some kind of "crib-book" telling Player what options she has at any given point. For example, there is only one strategy on the arena **1**, namely $\{\varepsilon\}$. A more interesting example is the "one-off negation" strategy on $\mathbf{B} \Rightarrow \mathbf{B}$ with plays:

$$
\begin{array}{ccc}
\mathbf{B} & \Rightarrow & \mathbf{B} \\
 & & \mathsf{q} \\
\mathsf{q} & & \\
\mathsf{tt} & & \\
 & & \mathsf{ff}
\end{array}
\qquad
\begin{array}{ccc}
\mathbf{B} & \Rightarrow & \mathbf{B} \\
 & & \mathsf{q} \\
\mathsf{q} & & \\
\mathsf{ff} & & \\
 & & \mathsf{tt}
\end{array}
$$

We denote by $\mathsf{dom}(\sigma)$ the set $\{sa \mid sab \in \sigma\}$, known as the ***domain*** of σ. This is intended to represent those "contexts" in which the strategy σ will do something. We write $\mathsf{Str}(A)$ for the set of all strategies for arena A.

3.2 Composition of strategies

The notion of composition of strategies is unarguably the keystone of game semantics. It has its root in both the *geometry of interaction* [36–38] analysis of β-reduction in λ-calculus and in the (synchronous) parallel composition operators found in process calculi such as CCS [67] and CSP [46]. Unlike function composition then, it's an inherently dynamic process of interaction. This is best illustrated with a couple of examples.

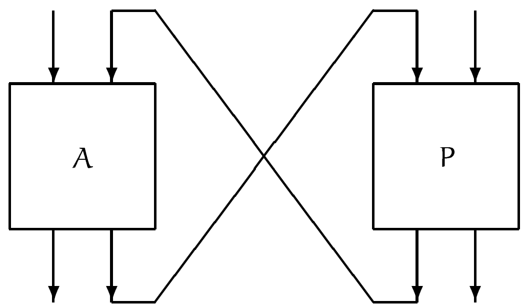
Suppose we have a negation "function" taking a Boolean input and returning its complement as a Boolean output. We'd like to "apply" this to an "argument" 'true'—expecting the result 'false'. We can think of both the "function" and its "argument" as *black boxes*; the "argument" is a box with a single output "wire" while the "function" has one input and one output "wire".

It's now intuitive to think of "composing" these two boxes by hooking up the output from the argument to the input of the function. It would also be in the spirit of a "black box methodology" to draw a new box around this composite process. This is clearly reminiscent of the two-stage process of "parallel composition and hiding" as found in CCS.

For a more complex example, consider a "process" (call it P), again consuming and producing a single Boolean value, but this time it repeatedly asks for inputs until it receives 'true'. In other words, if it gets an input 'false', it simply asks again for a new input; once it gets an input 'true', it outputs this—essentially it's a "filter" removing all occurrences of 'false' from an input data stream of Booleans. This behaviour can be represented by the following PCF term:

$$\mathbf{Y}_{\mathtt{Bool}\to\mathtt{Bool}}\,(\lambda f^{\,\mathtt{Bool}\to\mathtt{Bool}}.\,\lambda x^{\,\mathtt{Bool}}.\,\mathtt{if}\ x\ \mathtt{then}\ \mathtt{tt}\ \mathtt{else}\ f(x)\ \mathtt{fi}).$$

Hooking such a "process" up to an "argument" that always produces 'false' is clearly a mistake; this is a classical example of ***livelock***. But, consider the possible interactions with an "argument" (call it A) that nondeterministically produces either 'true' or 'false'. Once an output is demanded of P, it asks for input. This is provided by A. If A produces a 'true' input, P passes this on. Otherwise, P must ask again for another input—essentially it's back where it started. So P can "loop the loop" any finite number of times before A produces 'true'. This can be visualized by the following ***figure-of-eight*** diagram; see, for example, [4].



So we can view this notion of composition as a *synchronous* interaction: an internal response, such as A producing 'false', becomes an internal stimulus for P; in this case, P responds with a new stimulus for A... and so on until A produces 'true' whence P makes an external response 'true'. Note that in this example, it's also possible that a "malicious" execution takes place where A *never* produces 'true'. In this case, the interaction will continue like ∞ for as long as it likes...

Aside Composition of strategies is intimately linked to the notion of ***linear head reduction*** in λ-calculus as is shown in [26]. Linear head reduction was developed in the light of insights gained from studying ***proof net***, and subsequently ***pure net***, representations of λ-terms [25, 82]. The crucial observation is that any given λ-term T has a ***canonical form***, which takes the following general shape:

$$\lambda t_1 \dots \lambda t_m (\lambda e_1 \dots (\lambda e_n (t) A_1 \dots A_p) T_n \dots) T_1.$$

The λt_is are the ***head*** lambdas, the λe_js are the ***spine*** lambdas and the A_ks are the ***arguments*** of the ***head occurrence*** t. A canonical form (which is *not* generally unique) can be calculated by encoding T as a net, performing all available multiplicative reductions and reading off the canonical form from the ensuing net. We say that two λ-terms T and U are σ-equivalent iff they share a common canonical form. It's more usual to define this directly [83] as the least congruence on λ-terms containing

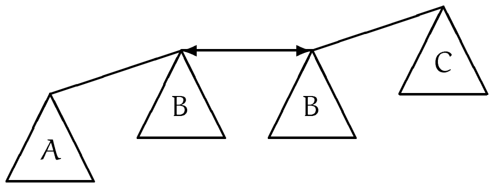
$$((\lambda x\, U)V_1)V_2 =_\sigma (\lambda x\, (U)V_2)V_1, \qquad \text{if } x \notin \mathtt{fv}(V_2)$$
$$(\lambda x_1\, \lambda x_2 U)V =_\sigma \lambda x_2\, (\lambda x_1\, U)V, \qquad \text{if } x_2 \notin \mathtt{fv}(V).$$

The first clause "extracts" the redex $(\lambda x\, U)V_1$ from being buried inside the application to V_2; note how the scope of x expands, hence the side condition. The second clause pushes the "potential" redex $(\lambda x_1\, U)V$ underneath the (essentially redundant) λx_2; again, note how the scope of x_2 expands.

Consider a canonical form T. It is a head normal form (*hnf*) if, and only if, $n = 0$. If the head occurrence t is free or bound by a head lambda, we say that T is a ***quasi*** head normal form (*qhnf*). Otherwise, suppose t is bound by λe_i. In this case, we say that T ***linear head reduces*** to $\lambda t_1 \dots \lambda t_m (\lambda e_1 \dots (\lambda e_n (T_i) A_1 \dots A_p) T_n \dots) T_1$. All that happens is that the subterm T_i corresponding to the binding λe_i is substituted for the head *occurrence* t; this is known as ***linear*** substitution. The linear head reduction of a term consists of repeatedly massaging it into a canonical form (multiplicative reduction) and performing this linear substitution (exponential firing) until a *qhnf* is reached; this is only well-defined upto σ-equivalence.

3.2.1 Legal interactions

We now return to the main thrust of this section by formalizing the discussion of the figure-of-eight mode of composition. Suppose we have strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ representing two processes that we wish to "compose" in this way. We can intuitively think of combining $A \Rightarrow B$ and $B \Rightarrow C$ into a "composite arena" $A \Rightarrow B\,; B \Rightarrow C$, looking something like this.



The figure-of-eight notion of composition means effectively that we consider the two occurrences of arena B to be connected by a synchronous buffer—a *cut link* of linear logic [35]. This means that we're only really interested in a restricted class of the legal plays of this proto-arena, the so-called *interactions*, where any Player move in (either occurrence of) B is immediately copied across to become an Opponent move in the other occurrence. This leads to a certain amount of redundancy in our representation (lots of repetition in B) which can be technically finessed with the following definition.

Definition Let u be a finite string of moves from arenas A, B and C together with "justification pointers" from all moves except those initial in C. We define $u \restriction B, C$ to be the subsequence of u consisting of all moves from B and C with their associated pointers, but where we delete any pointer to a move in A. We can similarly define $u \restriction A, B$, deleting any pointers to moves in C. We say that u is an ***interaction*** of A, B and C iff $u \restriction A, B \in J_{A \Rightarrow B}$ and $u \restriction B, C \in J_{B \Rightarrow C}$.

If u is an interaction of A, B and C then the "justification pointer" from any move of C must be to another move of C. But consider the case of a move a of A; *either* it points back to another move of A *or* it points to an *initial* move b of B. In this latter case, b itself points to an initial move c of C. We consequently define $u \restriction A, C$ to be the subsequence of u consisting of all moves from A and C but where, in the situation above, the pointers from a to b and from b to c are "tied together" to give a new pointer direct from a to c. All other pointers are inherited directly from u.

A ***legal interaction*** of arenas A, B and C is an interaction of A, B and C such that $u \restriction A, B \in L_{A \Rightarrow B}$, $u \restriction B, C \in L_{B \Rightarrow C}$ *and* $u \restriction A, C \in L_{A \Rightarrow C}$. We write $\mathsf{int}(A, B, C)$ for the set of all legal interactions of A, B and C.

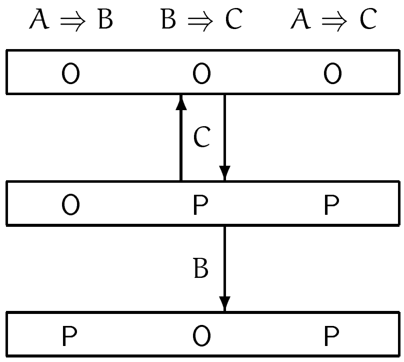
If $u \in \mathsf{int}(A, B, C)$ and $a \in u$ then we say that a has ***component*** $A \Rightarrow B$ iff it's a move of A or an O-move of B; "dually", a has component $B \Rightarrow C$ iff it's a move of C or a P-move of B. We use X and Y to range over components; if we're using X to denote one, Y is used to denote the other. If, for example, X denotes $A \Rightarrow B$, we write $u \restriction X$ for $u \restriction A, B$ and $u \restriction Y$ for $u \restriction B, C$.

A ***generalized*** O***-move*** (we should probably say "generalized O-occurrence") of u is (an occurrence of) a move in u which is either an O-move of $A \Rightarrow C$ or any move of B. The idea of this definition is that a move of u in B can be seen as an O-move in one or other of the components. Clearly there's a "dual" (but overlapping) notion of *generalized* P*-move* but we'll make no use of this.

State diagrams Many important properties in game semantics can be deduced with the aid of appropriate state diagrams. We'll encounter several in this thesis, beginning with the following one for legal interactions. The idea is that, by keeping track of whose turn it is to play in each of the internal components and also in the external restriction, we can see how the legality requirements constrain where the next move can be played.

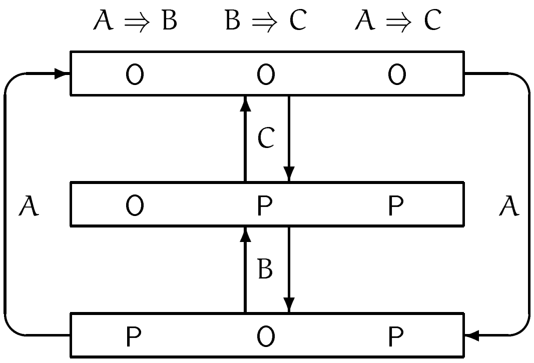
First of all, note that a non-empty $u \in \mathsf{int}(A, B, C)$ must begin with an initial move in C. After this, it's Player's turn in $A \Rightarrow C$ and $B \Rightarrow C$, but in $A \Rightarrow B$ nothing has happened yet so it's still Opponent's go. This constrains Player's options: she can move in B or in C but not in A. Why? If Player were to move in A, this would violate legality of $u \upharpoonright A, B$. If Player moves again in C, we're essentially back where we started; if she moves in B, this "toggles the state" of $A \Rightarrow B$ and $B \Rightarrow C$ so now it's Player's go in $A \Rightarrow B$ and Opponent's in $B \Rightarrow C$.

We can summarize this discussion with the following diagram where we keep track of whose turn it is to play in $A \Rightarrow B$, $B \Rightarrow C$ and $A \Rightarrow C$ respectively.



In the light of this, we can see in another way why we couldn't play in A in the "middle state" (OPP). In this state, it's Opponent to play in $A \Rightarrow B$ and Player to move in $A \Rightarrow C$. But no move of A in u can be simultaneously an O-move *and* a P-move. We *can* move in B though; this is because a move of B in u *is* both an O-move and a P-move—in this case, it's an O-move of $A \Rightarrow B$ and a P-move of $B \Rightarrow C$. This situation is reversed in the "bottom state" (POP). In this state, moves of C are ruled out for the same reasons as moves of A in OPP, but moves of A are perfectly acceptable.

We can now complete the ***state diagram*** for legal interactions.



The most important property of legal interactions can easily be deduced from this picture: given an O-move m in C, say, the next move n cannot be in A. In other words, Player can only "switch" locally; she cannot "leap across" from C to A.

3.2.2 Formalizing composition

We can now give a formal definition for the composition of two strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$. This is done in two stages; first of all, the ***parallel composition*** is defined to be

$$\sigma \parallel \tau =_{\text{df}} \{u \in \mathsf{int}(A, B, C) \mid u \restriction A, B \in \sigma \wedge u \restriction B, C \in \tau\}.$$

The idea is that σ and τ *synchronize* on their common ground, namely the arena B. If τ plays a P-move m (of $B \Rightarrow C$) in B, the effect is that "control is passed" to σ who can now respond to m in A or B. (Note that m looks like an O-move of $A \Rightarrow B$ to σ.) If σ were to respond in B, control would return to τ... This is the formal counterpart to the figure-of-eight intuition.

We complete the definition by "hiding" all the interesting stuff, *i.e.* the interaction between σ and τ in B.

$$\sigma ; \tau =_{\text{df}} \{u \restriction A, C \mid u \in \sigma \parallel \tau\}$$

We say that $\sigma;\tau$ is the ***composite*** of σ and τ. For this to make sense, we need to be sure that $\sigma ; \tau$ really is a strategy for $A \Rightarrow C$.

Well-defined composition We're aiming to prove that the composite $\sigma ; \tau$ of $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ is a well-defined strategy for $A \Rightarrow C$. We first prove a useful little "curtailing" lemma which allows us to "cut back" a legal interaction to demonstrate the existence of a witness to some earlier play.

Lemma 3.2.1 If $u \in \mathsf{int}(A, B, C)$ and $a \in u$ is an O-move of $A \Rightarrow C$ then $u_{<a} \restriction A, B \in L^{\mathsf{even}}_{A \Rightarrow B}$ and $u_{<a} \restriction B, C \in L^{\mathsf{even}}_{B \Rightarrow C}$.

Proof By the state diagram, a must have been played when we were in state OOO, taking us to one of the other states. Therefore, when a was played, it was Opponent's turn in both $A \Rightarrow B$ and $B \Rightarrow C$ and so $u_{<a} \restriction A, B$ and $u_{<a} \restriction B, C$ must both have even length. ■

This lemma essentially tells us that, for $u \in \mathsf{int}(A, B, C)$, $u \restriction A, C$ has even length if, and only if, $u \restriction A, B$ and $u \restriction B, C$ both have even length. In other words, a legal interaction u can witness an even-length play if, and only if, it's in the "top" (OOO) state of our diagram.

Proposition 3.2.2 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ then $\sigma ; \tau$ is a well-defined strategy for $A \Rightarrow C$.

Proof We need to check that $\sigma ; \tau$ contains only legal plays, that they all have even length, that it also contains ε and that it's even-length prefix-closed. The first of these is immediate from the definition of legal interaction. The second follows since any $u \in \sigma \parallel \tau$ is either even-length in A, B *and* C or odd-length in A, B *and* C. The third is obvious since $\varepsilon \in \sigma$ and $\varepsilon \in \tau$ so $\varepsilon \in \sigma \parallel \tau$.

For the final part, if $sab \in \sigma ; \tau$ then there must be some witness for this, *i.e.* some $u \in \sigma \parallel \tau$ such that $sab = u \upharpoonright A, C$. So consider $u_{<a}$. Since a is an O-move in A or C, the state diagram tells us that the proceeding move of u was a P-move, also in A or C. So $u_{<a} \upharpoonright A, C = s$ and $u_{<a}$ ends in the outside. Now $u_{<a}$ is a legal interaction since for any arena A, L_A is prefix-closed. So all we need to check is that $u_{<a} \upharpoonright A, B$ and $u_{<a} \upharpoonright B, C$ both have even length so that, by (**s2**), they're in σ and τ respectively. But this follows immediately from the above lemma and we're done. ■

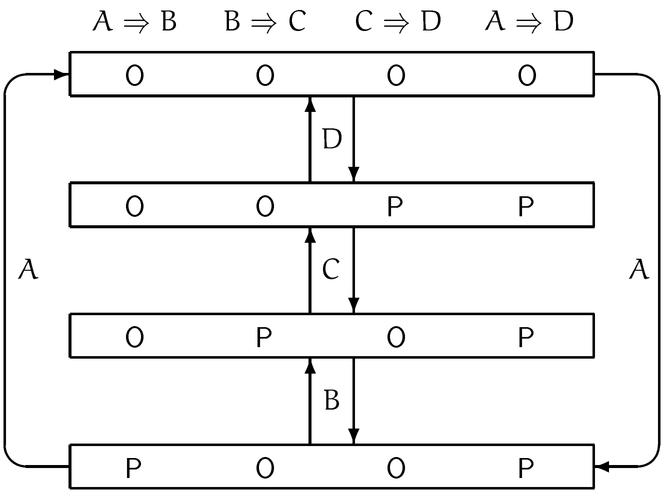
We've now covered the basic trinity of HO-style game semantics: the definition of *justified string*; the regular structure of *legal interactions*, as codifed in the state diagram; and the resulting theorem that composition of *strategies* is well-defined.

3.2.3 Composition is associative

Suppose we have three strategies $\sigma : A \Rightarrow B$, $\tau : B \Rightarrow C$ and $\upsilon : C \Rightarrow D$. We may wish to join them up into a "chain", yielding a new "strategy" $\sigma ; \tau ; \upsilon : A \Rightarrow D$. For this to make sense, it mustn't make any difference whether we first compose σ and τ and then the result with υ or if we do it the other way around. Formally, we want the composition operation on strategies to be associative.

We need to extend the notion of a legal interaction (of three arenas) to one of four arenas. It's clear how the definition of an interaction can be generalized; an interaction u of arenas A, B, C and D is ***legal***, written $u \in \mathsf{int}(A, B, C, D)$, iff the restrictions to A, B, to B, C, to C, D *and* to A, D are all legal. An equivalent way to state this is that $u \upharpoonright A, B, C \in \mathsf{int}(A, B, C)$ and $u \upharpoonright B, C, D \in \mathsf{int}(B, C, D)$.

Legal interactions of four arenas have a regular structure that's similar to that of their counterparts on three arenas; we have the following state diagram which is a simple extension of the three arena version.



The zipping lemma When we prove associativity of composition in this trace-based formulation of game semantics, we proceed by "uncovering", in two stages, how a play in one association was witnessed: if we start with a play in $(\sigma\,;\tau)\,;\upsilon$, we first find an interaction between $\sigma\,;\tau$ and υ; but the contribution of $\sigma\,;\tau$ must itself have come from another interaction, this time between σ and τ. So, in order to get the "full history" of interaction that has given rise to our original play in $(\sigma\,;\tau)\,;\upsilon$, we need to somehow combine these two stages into one.

Suppose, then, that we have an interaction $u \in \mathsf{int}(A, C, D)$ and another interaction $v \in \mathsf{int}(A, B, C)$ such that $u \restriction A, C = v \restriction A, C$. We'd like to combine these into some $w \in \mathsf{int}(A, B, C, D)$ such that $u = w \restriction A, C, D$ and $v = w \restriction A, B, C$. We'd also like this w to be uniquely determined by u and v. If we can do this then the question raised above, of combining two interactions into one, can be solved as a special case.

Consider two consecutive occurrences, a and b, in $u \restriction A, C$. It's clear from the state diagram for legal interactions (of four arenas) that any intervening moves of B and D must come solely from B or solely from D—it's impossible to "switch" from B to D, or *vice versa*, without going via C or A. This suggests that we can combine u and v into the desired w by "zipping" them together, "synchronizing" via the common ground of $u \restriction A, C$.

Lemma 3.2.3 (zipping) If we have $u \in \mathsf{int}(A, C, D)$ and $v \in \mathsf{int}(A, B, C)$ such that $u \restriction A, C = v \restriction A, C$ then there's a unique $w \in \mathsf{int}(A, B, C, D)$ such that $w \restriction A, C, D = u$ and $w \restriction A, B, C = v$.

Proof By induction on $|u \restriction A, C|$. The base case is easy. For the inductive step, consider the last occurrence of $u \restriction A, C$; call it a. The above observation implies that a is also the last occurrence of one or other (maybe even both) of u and v. Suppose it's u so that $u = u'a$ and $v = v'av''$ where v'' consists entirely of moves from B. Clearly u' and v' are legal interactions and $u' \restriction A, C = (u \restriction A, C)_{<a} = (v \restriction A, C)_{<a} = v' \restriction A, C$ so we can apply the inductive hypothesis to find a unique $w' \in \mathsf{int}(A, B, C, D)$ such that $u' = w' \restriction A, C, D$ and $v' = w' \restriction A, B, C$. Let $w = w'av''$. So $w \restriction A, C, D = w' \restriction A, C, D \cdot a = u$ and $w \restriction A, B, C = w' \restriction A, B, C \cdot av'' = v$ and this is clearly the unique such w. The other case is similar. ∎

Clearly this lemma is just one of a family of related results, the most obvious one being the "mirror image" obtained by starting with $u \in \mathsf{int}(A, B, D)$ and $v \in \mathsf{int}(B, C, D)$. We've chosen to state and prove the Zipping lemma for this one case since this suffices to illustrate the general argument which is always the same. In the sequel, we'll sometimes refer to "zipping", meaning not necessarily exactly the lemma proved here but something "very much like it".

The first time such a result was proved was in [11] where it was used to give an alternative (indeed, more general) way of defining composition of strategies. Our use of the lemma is quite different but the proof is almost exactly the same.

We can now demonstrate that composition is indeed associative.

Proposition 3.2.4 If $\sigma : A \Rightarrow B$, $\tau : B \Rightarrow C$ and $\upsilon : C \Rightarrow D$ then $(\sigma\,;\tau)\,;\upsilon = \sigma\,;(\tau\,;\upsilon)$.

Proof We prove the left-to-right inclusion; the other direction is similar.

Suppose $s \in (\sigma\,;\tau)\,;\upsilon$. This must have a witness $u \in (\sigma\,;\tau) \parallel \upsilon$ such that $s = u \restriction A, D$. By definition, $u \restriction A, C \in \sigma\,;\tau$ so there must be another witness $v \in \sigma \parallel \tau$ such that $u \restriction A, C = v \restriction A, C$. We apply lemma 3.2.3 to "zip up" u and v, yielding $w \in \mathsf{int}(A, B, C, D)$ such that $w \restriction A, D = s$, $w \restriction A, B, C = v$ and $w \restriction A, C, D = u$. Now $w \restriction B, C = v \restriction B, C \in \tau$ and $w \restriction C, D = u \restriction C, D \in \upsilon$ so, provided $w \restriction B, C, D \in \mathsf{int}(B, C, D)$, we can "hide" C to obtain $w \restriction A, B, D \in \sigma \parallel (\tau\,;\upsilon)$ and hence, $s = w \restriction A, D \in \sigma\,;(\tau\,;\upsilon)$.

But is $w \restriction B, C, D \in \mathsf{int}(B, C, D)$? It's certainly an interaction; we also know that the restrictions to B, C and to C, D are legal, so all we need to check is alternation in B, D. But this is immediate from the above state diagram. ■

Aside The definition of a legal interaction $u \in \mathsf{int}(A, B, C)$ insisted not only on the internal restrictions ($u \restriction A, B$ and $u \restriction B, C$) being legal but also the external restriction ($u \restriction A, C$). To see why this should be, consider the following interaction of $\perp$, $\perp$ and $\perp$.

$$\begin{array}{ccccc} \perp & \Rightarrow & \perp & \Rightarrow & \perp \\ & & & & q \\ & & q & & \\ & & & & q \\ q & & & & \end{array}$$

The internal restrictions are both legal and yet this interaction somehow violates our figure-of-eight intuition: with the first move, Opponent initiates an "interaction"; but before this has had a chance to finish, Opponent initiates another interaction. The consequence is that the resulting external restriction violates legality and although, in some sense, Player can "catch up"...

$$\begin{array}{ccccc} \perp & \Rightarrow & \perp & \Rightarrow & \perp \\ & & & & q \\ & & q & & \\ & & & & q \\ q & & & & \\ & & q & & \\ q & & & & \end{array}$$

...the damage has already been done as far as legality is concerned. Insisting on legality of the external restriction *enforces* the passing of control from one strategy to the other which should occur when a move is played in B—this is precisely the content of the state diagram for legal interactions.

3.2.4 Copycat strategies

We now introduce an important class of strategies. Given any arena A, the strategy $\mathsf{id}_A : A \Rightarrow A$ is defined to be

$$\{s \in L^{\mathsf{even}}_{A \Rightarrow A} \mid \forall s' \sqsubseteq^{\mathsf{even}} s.\ s' \upharpoonright A_\ell = s' \upharpoonright A_r\},$$

where we use the ℓ and r tags to distinguish between the two occurrences of A and $s' \sqsubseteq^{\mathsf{even}} s$ means that s' is an even-length prefix of s. The import of this definition is that a move by Opponent in either occurrence of A is *immediately* copied (by Player) to the other occurrence—hence the term ***copycat*** strategy.

If we have some strategy $\sigma : A \Rightarrow B$, we want $\mathsf{id}_A ; \sigma = \sigma = \sigma ; \mathsf{id}_B$.

Lemma 3.2.5 If we have a strategy $\sigma : A \Rightarrow B$ such that $u \in \mathsf{id}_A \parallel \sigma \subseteq \mathsf{int}(A_\ell, A_r, B)$ then $u \upharpoonright A_\ell, B = u \upharpoonright A_r, B$.

Proof We can see from the state diagram that any $u \in \mathsf{int}(A_\ell, A_r, B)$ consists of a sequence of "segments" of the form $am_1 \cdots m_k b$ where a is an O-move of "the outside", *i.e.* A_ℓ or B, b is a P-move of the outside and the m_is are moves "in the middle", *i.e.* of A_r.

The proof is by induction on the length of u. The base case is trivial. For the inductive step, we write u as $u'am_1 \cdots m_k b$ and, by the inductive hypothesis, $u' \upharpoonright A_\ell, B = u' \upharpoonright A_r, B$. There are four cases to consider, depending on whether a occurs in A_ℓ or B and how σ responds to a. For example, if a occurs in A_ℓ and σ responds with some b in B, we have an interaction:

$$\begin{array}{ccccc} A_\ell & \Rightarrow & A_r & \Rightarrow & B \\ & & \vdots & & \\ a & & & & \\ & & a & & \\ & & & & b \end{array}$$

From this, it's easy to see that $u \upharpoonright A_\ell, B = u \upharpoonright A_r, B$; the other cases are similar. ■

If $s \in \mathsf{id}_A ; \sigma$ then it has a witness $u \in \mathsf{id}_A \parallel \sigma$ and, by the above lemma, $s = u \upharpoonright A_r, B \in \sigma$. For the converse, if $s \in \sigma$ then it's easy to build some $u \in \mathsf{id}_A \parallel \sigma$ witnessing $s \in \mathsf{id}_A ; \sigma$. The "mirror image" of this argument establishes $\sigma = \sigma ; \mathsf{id}_B$.

Another useful example of a copycat strategy is the ***diagonal*** strategy for an arena A, written $\Delta_A : A \Rightarrow A \times A$, defined by

$$\{s \in L^{\mathsf{even}}_{A \Rightarrow (A_\ell \times A_r)} \mid \forall s' \sqsubseteq^{\mathsf{even}} s.\ (s' \upharpoonright \ell \in \mathsf{id}_A \wedge s' \upharpoonright r \in \mathsf{id}_A)\}.$$

This essentially copycats between the two occurrences (we use the ℓ and r subscripts to distinguish them) of A on the right hand side (RHS) of the arrow and the single occurrence on the left hand side (LHS), the net effect being an interleaving in the LHS of the play in the RHS.

3.3 A category of arenas & strategies

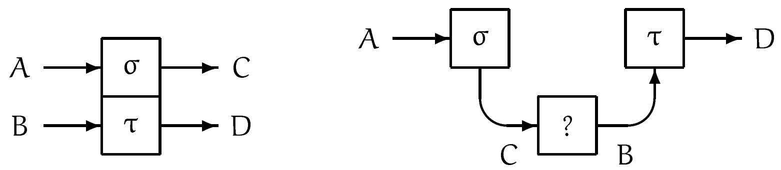
It is clear that we now have enough to build a basic category where objects are arenas and an arrow $f : A \to B$ is a strategy $\sigma_f : A \Rightarrow B$. (We'll generally blur the distinction between an arrow and the strategy that gives rise to it.) We compose two arrows by composing the associated strategies; we know from the previous section that this is well-defined and associative. The copycat strategy id_A is the identity arrow for the object A. We denote this category by **G**.

3.3.1 Symmetric monoidal closed structure

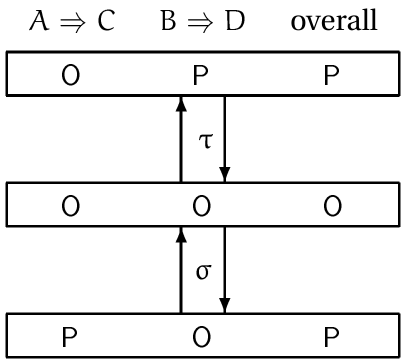
This basic category already has quite a lot of useful structure. We extend the product and arrow operations to strategies as follows: given $\sigma : A \Rightarrow C$ and $\tau : B \Rightarrow D$, we define $\sigma \times \tau : (A \times B) \Rightarrow (C \times D)$ and $\sigma \Rightarrow \tau : (C \Rightarrow B) \Rightarrow (A \Rightarrow D)$ by:

$$\sigma \times \tau =_{\mathrm{df}} \{s \in L_{(A\times B)\Rightarrow(C\times D)} \mid s \restriction A, C \in \sigma \wedge s \restriction B, D \in \tau\}$$
$$\sigma \Rightarrow \tau =_{\mathrm{df}} \{s \in L_{(C\Rightarrow B)\Rightarrow(A\Rightarrow D)} \mid s \restriction A, C \in \sigma \wedge s \restriction B, D \in \tau\}.$$

Despite a superficial similarity in these definitions, the resulting strategies show markedly different behaviours. The product operation simply places strategies "side by side" whereas the arrow sets up a more complex dependency. This can be viewed pictorially:



We need to show that $\sigma \times \tau$ and $\sigma \Rightarrow \tau$ are well-defined. For $\sigma \times \tau$, note that, if Opponent starts in C (*i.e.* in σ) then it's now Player's go in $A \Rightarrow C$ while it's still Opponent's turn in $B \Rightarrow D$. This means that Player is forced to move in $A \Rightarrow C$, *i.e.* respond as σ. These kinds of properties have become known as *switching conditions*. We can summarize this one in the following state diagram.



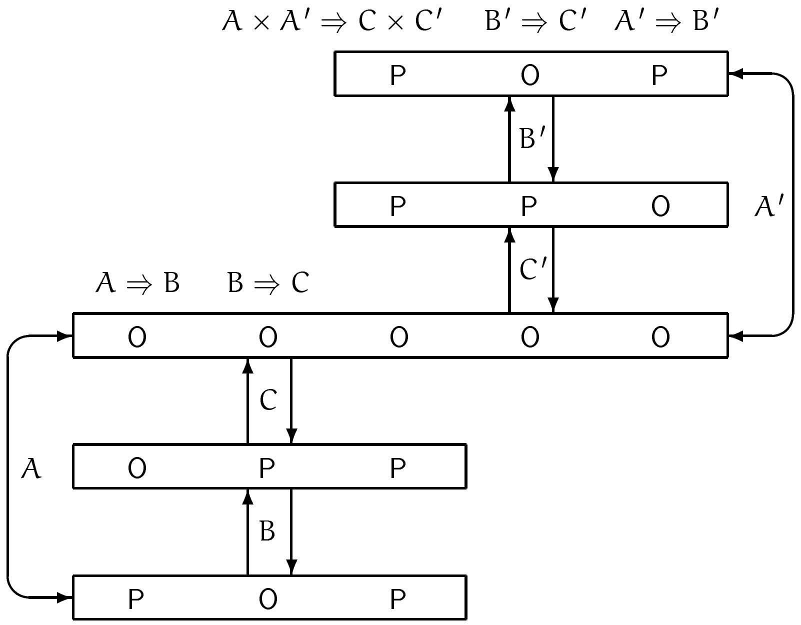
Any trace of $\sigma \times \tau$ is in the "middle" state (OOO). An immediate corollary of this is that $\sigma \times \tau$ is well-defined: if $sab \in \sigma \times \tau$ then the ab must come entirely from either σ or τ and, since σ and τ are both well-defined strategies, $s \in \sigma \times \tau$. The same state diagram and argument shows that $\sigma \Rightarrow \tau$ is well-defined.

Product & arrow as bifunctors The essential point about the switching condition for $\times$ is that, if we "prod" σ then this will not provoke a reaction from τ: they may be side-by-side but they don't interact with one another.

Suppose now that we have four strategies: $\sigma : A \Rightarrow B$, $\sigma' : A' \Rightarrow B'$, $\tau : B \Rightarrow C$ and $\tau' : B' \Rightarrow C'$. Consider $u \in (\sigma \times \sigma') \parallel (\tau \times \tau')$. Given the above discussion, we might hope that $u \upharpoonright A, B, C \in \sigma \parallel \tau$ and $u \upharpoonright A', B', C' \in \sigma' \parallel \tau'$.

Lemma 3.3.1 If $u \in \mathsf{int}(A \times A', B \times B', C \times C')$ such that $u \upharpoonright A, B$, $u \upharpoonright A', B'$, $u \upharpoonright B, C$ and $u \upharpoonright B', C'$ are all legal (in the appropriate arenas) then $u \upharpoonright A, B, C \in \mathsf{int}(A, B, C)$ and $u \upharpoonright A', B', C' \in \mathsf{int}(A', B', C')$.

Proof The proof is by state diagram. We combine the earlier diagrams for legal interactions and for $\times$, obtaining the following.



From this, it's easy to see that restricting to either A, B, C or to A', B', C' results in a legal interaction in that restriction: there is no "tangling". ■

We have a kind of "inverse" to this which allows us to reconstruct an interaction $u \in \mathsf{int}(A \times A', B \times B', C \times C')$ by zipping together something from $\mathsf{int}(A, B, C)$ with something from $\mathsf{int}(A', B', C')$.

Lemma 3.3.2 Suppose we have $s \in L_{(A\times A')\Rightarrow(C\times C')}$, $u \in \mathsf{int}(A,B,C)$ and $u' \in \mathsf{int}(A',B',C')$ such that $s \upharpoonright A,C = u \upharpoonright A,C$ and $s \upharpoonright A',C' = u' \upharpoonright A',C'$. Then there exists a unique $v \in \mathsf{int}(A\times A', B\times B', C\times C')$ such that $s = v \upharpoonright A\times A', C\times C'$, $u = v \upharpoonright A,B,C$ and $u' = v \upharpoonright A',B',C'$.

Proof The usual kind of "zipping" argument, going by induction on $|s|$. ■

It's clear that $\mathsf{id}_A \times \mathsf{id}_B = \mathsf{id}_{A\times B}$ and $\mathsf{id}_A \Rightarrow \mathsf{id}_B = \mathsf{id}_{A\Rightarrow B}$. We now verify the other functor law for $\times$. The proof for $\Rightarrow$ is very similar.

Lemma 3.3.3 If we have $\sigma : A \Rightarrow B$, $\sigma' : A' \Rightarrow B'$, $\tau : B \Rightarrow C$ and $\tau' : B' \Rightarrow C'$ then $(\sigma\,;\tau)\times(\sigma'\,;\tau') = (\sigma\times\sigma')\,;(\tau\times\tau')$.

Proof We show the left-to-right inclusion first. If $s \in (\sigma\,;\tau)\times(\sigma'\,;\tau')$ then, by definition, we have $s \upharpoonright A,C \in \sigma\,;\tau$ and $s \upharpoonright A',C' \in \sigma'\,;\tau'$. This means that we must have $u \in \mathsf{int}(A,B,C)$ witnessing $s \upharpoonright A,C$ and $u' \in \mathsf{int}(A',B',C')$ witnessing $s \upharpoonright A',C'$. By lemma 3.3.2, we can find a unique $v \in \mathsf{int}(A\times A', B\times B', C\times C')$ such that $s = v \upharpoonright A\times A', C\times C'$, $u = v \upharpoonright A,B,C$ and $u' = v \upharpoonright A',B',C'$. So $v \upharpoonright A,B = u \upharpoonright A,B \in \sigma$ and $v \upharpoonright A',B' = u \upharpoonright A',B' \in \sigma'$ and hence $v \upharpoonright A\times A', B\times B' \in \sigma\times\sigma'$. For similar reasons, $v \upharpoonright B\times B', C\times C' \in \tau\times\tau'$ so that $s = v \upharpoonright A\times A', C\times C' \in (\sigma\times\sigma')\,;(\tau\times\tau')$. For the other way, suppose we have $s \in (\sigma\times\sigma')\,;(\tau\times\tau')$. This must have a witness $u \in (\sigma\times\sigma') \parallel (\tau\times\tau')$. So $u \upharpoonright A,B \in \sigma$ and $u \upharpoonright B,C \in \tau$. By lemma 3.3.1, $u \upharpoonright A,B,C \in \mathsf{int}(A,B,C)$ so $u \upharpoonright A,C \in \sigma\,;\tau$. Similarly we have $u \upharpoonright A',C' \in \sigma'\,;\tau'$ and so $s = u \upharpoonright A\times A', C\times C' \in (\sigma\,;\tau)\times(\sigma'\,;\tau')$. ■

Symmetric monoidal structure Recall that a symmetric monoidal category is a 7-tuple $\langle \mathbf{C}, \otimes, I, \alpha, \lambda, \rho, \gamma\rangle$ where $\mathbf{C}$ is a category, $-\otimes- : \mathbf{C}\times\mathbf{C} \to \mathbf{C}$ is a bifunctor, I is some object of $\mathbf{C}$ and the others are natural isomorphisms...

$$\begin{array}{l} \alpha_{A,B,C} : (A\otimes B)\otimes C \cong A\otimes(B\otimes C) \\ \gamma_{A,B} : A\otimes B \cong B\otimes A \\ \lambda_A : I\otimes A \cong A,\ \rho_A : A\otimes I \cong A \end{array}$$

...required to satisfy the well-known Kelly-Mac Lane coherence conditions. The standard reference for this is Mac Lane [59] but we'd also recommend Chapter 2 of [63] for a brief but lucid account.

In our basic category, the monoidal structure is given by the bifunctor $\times$ which, we admit, is notationally a bit confusing—but in §3.5 we'll introduce an important subcategory where $\times$ really is a categorical product so perhaps we're justified in this choice. Anyway, the required natural isomorphisms are just the obvious copycat strategies induced by the natural bijections (in **Set**) between the move sets, *e.g.* for associativity we have:

$$(M_A + M_B) + M_C \cong M_A + (M_B + M_C).$$

It's then a straightforward enterprise to verify the equations demanded by the coherence conditions. The "tensor unit" I is the empty arena $\mathbf{1}$.

Closed structure A symmetric monoidal closed category (SMCC) is a symmetric monoidal category where every functor $A \otimes -$ has a specified right adjoint $A \multimap -$ giving rise to the familiar ***currying*** natural isomorphism.

$$\mathbf{C}(A \otimes B, C) \cong \mathbf{C}(B, A \multimap C)$$

Our candidate right adjoint is $A \Rightarrow -$ and, since the only difference between the arenas $(A \times B) \Rightarrow C$ and $B \Rightarrow (A \Rightarrow C)$ lies in the tagging of the moves in the appropriate disjoint unions, we have

$$\begin{array}{rcl} \mathbf{G}(A \times B, C) & = & \{\sigma \in \mathsf{Str}((A \times B) \Rightarrow C)\} \\ & \cong & \{\sigma \in \mathsf{Str}(B \Rightarrow (A \Rightarrow C))\} \\ & = & \mathbf{G}(B, A \Rightarrow C) \end{array}$$

Denoting this isomorphism by $\Lambda_A(-)$, we define $\mathsf{eval}_{A,B} : (A \times (A \Rightarrow B)) \Rightarrow B$ by $\Lambda_A^{-1}(\mathsf{id}_{A \Rightarrow B})$ and a routine verification establishes that, for $\sigma : A \times B \Rightarrow C$, we have

$$\sigma = (\mathsf{id}_A \times \Lambda_A(\sigma)) \,;\mathsf{eval}_{A,C}.$$

Proposition 3.3.4 Our basic category **G** of arenas and strategies is an SMCC.

Finally we note that, for any arena A, we have a unique strategy, written $!_A$, on $A \Rightarrow \mathbf{1}$, *i.e.* $\mathbf{1}$ is a terminal object of **G**.

3.3.2 Ordering strategies

We assume the basic definitions of order theory such as partial orders, suprema and infima, directed and consistent sets, (complete) lattices and CPOs, and their appropriate homomorphisms. Our basic reference for all this is Davey & Priestley [30], whose notation and terminology we also adopt.

A partial ordering on strategies The obvious idea to order strategies, namely subset inclusion, turns out to be exactly right. If σ and τ are strategies on an arena A, we define

$$\sigma \leqslant_A \tau \text{ iff } \sigma \subseteq \tau.$$

This is patently a partial order on the set of strategies for A. There's always a least element, denoted by $\perp_A$, namely $\{\varepsilon\}$.

Given two strategies σ and τ on some arena A, we define $\sigma \vee \tau =_{\mathrm{df}} \sigma \cup \tau$ and $\sigma \wedge \tau =_{\mathrm{df}} \sigma \cap \tau$. It's easy to check that $\sigma \vee \tau$ and $\sigma \wedge \tau$ are well-defined strategies. For example, if $sab \in \sigma \wedge \tau$ then it must be in σ and τ, so s is also in σ and τ and hence in $\sigma \wedge \tau$. It's also obvious that $\sigma \vee \tau$ and $\sigma \wedge \tau$ are, respectively, the *least* upper bound (lub) and *greatest* lower bound (glb) of σ and τ. These lubs and glbs generalize to arbitrary sets of strategies in an evident way.

Although we'll only generally be interested in lubs of directed sets, we may as well state the stronger result.

Lemma 3.3.5 Given an arena A, the set of all strategies on A ordered by $\leqslant$ is a complete lattice.

It's also easy to check that composition of strategies is monotone with respect to this ordering. For example, if σ and τ are composable strategies and $\tau \leqslant \tau'$ then any $u \in \sigma \parallel \tau$ is also in $\sigma \parallel \tau'$.

Lemma 3.3.6 If $\sigma, \sigma' : A \Rightarrow B$ and $\tau, \tau' : B \Rightarrow C$ are strategies such that $\sigma \leqslant \sigma'$ and $\tau \leqslant \tau'$ then $\sigma ; \tau \leqslant \sigma' ; \tau'$.

Now, let Δ be a directed set of strategies for $B \Rightarrow C$ and let σ and υ be strategies for $A \Rightarrow B$ and $C \Rightarrow D$ respectively. We write $\sigma ; \Delta$ for the (directed) set $\{\sigma ; \tau \mid \tau \in \Delta\}$ and similarly write $\Delta ; \upsilon$ for the obvious set. It's now straightforward to verify that composition is continuous, *i.e.* our category "is" CPO-enriched.

Lemma 3.3.7 If Δ is a directed set of strategies for $B \Rightarrow C$ and $\sigma : A \Rightarrow B$ and $\upsilon : C \Rightarrow D$ are strategies then $\sigma ; (\bigsqcup \Delta) = \bigsqcup(\sigma ; \Delta)$ and $(\bigsqcup \Delta) ; \upsilon = \bigsqcup(\Delta ; \upsilon)$.

Proof If $s \in \sigma ; (\bigsqcup \Delta)$ then $s \in \sigma ; \tau$ for some $\tau \in \Delta$ and so $s \in \bigsqcup(\sigma ; \Delta)$. The other half follows from monotonicity. The other equality is basically the same. ■

Compact strategies & algebraicity An element $x \in P$ of a CPO is ***compact*** iff for all directed $\Delta \subseteq P$, if $x \leqslant \bigsqcup \Delta$ then $x \leqslant d$ for some $d \in \Delta$, the idea being that any "computation" of $\bigsqcup \Delta$ yields x at a "finite stage". The set of compact elements of P is denoted by $\mathcal{K}(P)$.

Now it's clear that any strategy with finite cardinality must be compact: any finite subset $\{\sigma_1, \ldots , \sigma_n\} \subseteq^{\mathsf{fin}} \Delta$ of a directed set Δ has an upper bound in Δ.

In fact, we also have the "converse" result that any compact strategy necessarily has finite cardinality. To see this, consider any strategy σ on arena A and let

$$\Delta_\sigma = \{\sigma' \in \mathsf{Str}(A) \mid \sigma' \subseteq^{\mathsf{fin}} \sigma\}.$$

It's clear that Δ_σ is directed and that σ is an upper bound, *i.e.* $\bigsqcup \Delta_\sigma \subseteq \sigma$. Now, for each $s \in \sigma$, we have a strategy $\sigma_s = \{s' \in L_A^{\mathsf{even}} \mid s' \sqsubseteq s\}$ which clearly has finite cardinality. So each $\sigma_s \in \Delta_\sigma$ and hence $s \in \bigsqcup \Delta_\sigma$, *i.e.* $\sigma \subseteq \bigsqcup \Delta_\sigma$. So $\sigma = \bigsqcup \Delta_\sigma$. This means that any strategy is the lub of a directed set of strategies of finite cardinality and so no strategy with infinite cardinality can be compact.

Recall that P is an algebraic CPO iff it's a CPO where, for all $x \in P$, the set of all $x' \in \mathcal{K}(P)$ such that $x' \leqslant x$ is directed and, moreover, x is the lub of this set, *i.e.*

$$x = \bigsqcup \{x' \in \mathcal{K}(P) \mid x' \leqslant x\}.$$

This is equivalent to the definition in terms of the "way below" relation. Clearly, an immediate consequence of the above reasoning is:

Lemma 3.3.8 For any arena A, the set $\mathsf{Str}(A)$ ordered by $\leqslant_A$ is an algebraic CPO.

3.4 Constraining strategies

The category that we built in the previous section is fairly "wild" in the sense that strategies are subject to no constraints. In this section, we introduce a number of restrictions that can be placed on strategies, yielding various subcategories of interest.

There are essentially two ways in which we can constrain behaviours. The first is to use properties of legal plays, such as Player visibility, to induce constraints on strategies; *e.g.* a strategy can only contain plays satisfying P-visibility. The second way is concerned more with "global" properties of strategies, rather than those of the individual plays themselves. It subdivides into two basic cases: we can have *coherence* conditions saying that all the plays in a strategy are, in some sense, compatible with one another; and we can have *liveness* conditions that insist on certain plays being in a strategy, given that certain others already are.

We begin by examining the *visibility* condition and the *bracketing* condition, both of which are induced by properties of legal plays and, thus, they exemplify the first style of constraint mentioned above.

3.4.1 The visibility condition

First of all, we extend the notion of P-view to a (non-empty) legal interaction $u \in \mathsf{int}(A, B, C)$ as follows.

$$\begin{array}{rlr} \ulcorner ua \urcorner = a, & & \text{if } a \text{ is initial in } C \\ \ulcorner ubva \urcorner = \ulcorner u \urcorner ba, & & \text{if } a \text{ is an } \mathsf{O}\text{-move of } A \Rightarrow C \\ & & \text{and } b \text{ justifies } a \\ \ulcorner ua \urcorner = \ulcorner u \urcorner a, & & \text{if } a \text{ is a } \mathsf{P}\text{-move of } A \Rightarrow C \\ & & \text{or is a move of } B. \end{array}$$

This was known as the ***core*** of u, written $\overline{u}$, in [63] but, as this really is just a kind of "generalized P-view", we prefer to avoid introducing new notation and terminology; instead, we'll stick to the convention that s and t range over (bits of) legal *plays* while u and v range over (bits of) legal *interactions*.

In words, the P-view of u is the subsequence of u obtained by removing all segments $a \ldots b$ where a is an O-move of $A \Rightarrow C$, b is a P-move of $A \Rightarrow C$, all the intervening moves are in B and neither a nor b appears in $\ulcorner u \upharpoonright A, C \urcorner$. So it's just the part of u that's relevant to determining the P-view of its outer restriction.

Lemma 3.4.1 If $u \in \mathsf{int}(A, B, C)$ and $m \in \ulcorner u \urcorner$ is a generalized O-move of u then $\ulcorner u_{\leqslant m} \urcorner \upharpoonright A, C = \ulcorner u_{\leqslant m} \upharpoonright A, C \urcorner$.

Proof By induction on $|u_{\leqslant m}|$. If m is initial in C, it's trivial; this encompasses the base case. If m is a move of B, let $u_{\leqslant m} = vm$ so that $\ulcorner u_{\leqslant m} \urcorner = \ulcorner v \urcorner \cdot m$. From the state diagram for legal interactions, the last move of v is a generalized O-move of u so, by the inductive hypothesis, $\ulcorner u_{\leqslant m} \urcorner \upharpoonright A, C = \ulcorner v \urcorner \upharpoonright A, C = \ulcorner v \upharpoonright A, C \urcorner = \ulcorner u_{\leqslant m} \upharpoonright A, C \urcorner$. If m is a move of $A \Rightarrow C$, let $u_{\leqslant m} = vnv'm$ where n justifies m. So $\ulcorner u_{\leqslant m} \urcorner \upharpoonright A, C = \ulcorner v \urcorner \upharpoonright A, C \cdot nm$. Since n is a P-move of $A \Rightarrow C$, the last move of v is a generalized O-move of u so $u_{\leqslant m} \upharpoonright A, C = \ulcorner v \upharpoonright A, C \urcorner \cdot nm = \ulcorner u_{\leqslant m} \upharpoonright A, C \urcorner$ and we're done. ■

The following crucial result is essentially lemma 3.2.3 of [63].

Lemma 3.4.2 If $u \in \mathsf{int}(A, B, C)$ where $u \upharpoonright A, B$ and $u \upharpoonright B, C$ both satisfy Player visibility and m is a generalized O-move of u with component X then $\ulcorner u_{\leqslant m} \upharpoonright X \urcorner = \ulcorner \ulcorner u_{\leqslant m} \urcorner \upharpoonright X \urcorner$.

Proof By induction on $|u_{\leqslant m}|$. If m is initial in X, it's trivial; once again, this encompasses the base case. Otherwise, we must first show that the justifier of m (call it n) occurs in $\ulcorner u_{\leqslant m} \urcorner$. If m is an O-move of $A \Rightarrow C$, it obviously occurs in $\ulcorner u_{\leqslant m} \urcorner \upharpoonright A, C = \ulcorner u_{\leqslant m} \upharpoonright A, C \urcorner$ by the previous lemma. By the definition of P-view, the justifier of m occurs in $\ulcorner u_{\leqslant m} \upharpoonright A, C \urcorner$. If m is a move of B, it's a P-move in component Y so its justifier occurs in $\ulcorner u_{<m} \upharpoonright Y \urcorner$. The last move of $u_{<m}$ is a generalized O-move with component Y so, by the inductive hypothesis, $\ulcorner u_{<m} \upharpoonright Y \urcorner = \ulcorner \ulcorner u_{<m} \urcorner \upharpoonright Y \urcorner$. So $n \in \ulcorner u_{<m} \urcorner$ and hence in $\ulcorner u_{\leqslant m} \urcorner$ (since m is in B).

Now, let $u_{\leqslant m} = vnv'm$ so that $\ulcorner u_{\leqslant m} \upharpoonright X \urcorner = \ulcorner v \upharpoonright X \urcorner \cdot nm$. From the state diagram, we know that the last move of v is a generalized O-move with component X so, by the inductive hypothesis, we have $\ulcorner u_{\leqslant m} \upharpoonright X \urcorner = \ulcorner \ulcorner v \urcorner \upharpoonright X \urcorner \cdot nm$ and, since $n \in \ulcorner u_{\leqslant m} \urcorner$, this equals $\ulcorner \ulcorner u_{\leqslant m} \urcorner \upharpoonright X \urcorner$ and we're done. ■

Player visibility A strategy σ satisfies ***Player visibility*** (we might also say that it satisfies P***-visibility*** or just that it's P***-vis***) iff

$$\text{for all } sab \in \sigma, \text{ the justifier of } b \text{ occurs in } \ulcorner sa \urcorner.$$

Note that, since strategies are closed under even-length prefixes, every play in a P-vis strategy must satisfy the Player visibility condition for legal plays. Also, because any P-view contains exactly one initial occurrence, the P-visibility condition prevents Player from "switching context"; whatever thread Opponent has just played in, Player is obliged to continue in. The following is therefore a simple example of a violation of P-visibility...

$$\begin{array}{ccc}
\mathbf{C} & \Rightarrow & \mathbf{C} \\
 & & \mathbf{q} \\
\mathsf{q} & & \\
 & & \mathsf{q} \\
\mathsf{q} & &
\end{array}$$

...where both the Questions in the left-hand copy of **C** are justified by the bold initial occurrence.

It is shown in [7] that strategies subject to Player visibility suffice to model ground type references, *i.e.* where we can store data such as Boolean values or integers. An indication of the strength of this constraint is that strategies not subject to P-vis are necessary to model *general* references [3].

Robustness of P-visibility If we only wish to consider those strategies satisfying some constraint—such as P-visibility—then we must be sure that the class of such strategies is closed under composition. We will sometimes refer to this property as the *robustness* of said constraint.

Proposition 3.4.3 If $u \in \mathsf{int}(A, B, C)$ such that both $u \restriction A, B$ and $u \restriction B, C$ satisfy Player visibility then so does $u \restriction A, C$.

Proof Let $m \in u$ be (an occurrence of) a P-move (of $A \Rightarrow C$) in component X. Since u is P-vis in both components, the justifier of m must occur in $\ulcorner u_{<m} \restriction X \urcorner$. Now, the state diagram for legal interactions tells us that the last occurrence in $u_{<m}$ is a generalized O-move with component X so, by lemma 3.4.2, $\ulcorner u_{<m} \restriction X \urcorner = \ulcorner \ulcorner u_{<m} \urcorner \restriction X \urcorner \preceq \ulcorner u_{<m} \urcorner \restriction X$. So, provided m isn't initial in A, its justifier occurs in $\ulcorner u_{<m} \urcorner \restriction A, C = \ulcorner u_{<m} \restriction A, C \urcorner$ (by lemma 3.4.1) as desired.

But, what if m is initial in A? In this case, the justifying move n is an initial occurrence in B which is, in turn, justified by some initial occurrence n' in C. The above reasoning tells us that $n \in \ulcorner \ulcorner u_{<m} \urcorner \restriction A, B \urcorner$. But since n is a P-move of $B \Rightarrow C$, applying the same argument again shows that $n' \in \ulcorner \ulcorner u_{<n} \urcorner \restriction B, C \urcorner$. Since $n \in \ulcorner u_{<m} \urcorner$, we must have $n' \in \ulcorner u_{<m} \urcorner \restriction A, C = \ulcorner u_{<m} \restriction A, C \urcorner$ and we're done. ■

An immediate corollary of this result is that, since P-vis strategies only contain P-vis plays, composing two P-vis strategies must result in another P-vis strategy.

Corollary 3.4.4 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ both satisfy Player visibility then so does their composite $\sigma ; \tau$.

3.4.2 The bracketing condition

The bracketing condition is typically used to enforce a *stack discipline*: when an Answer is played, it must be answering the *pending* Question, that is, the most recently asked but as yet unanswered Question. This has two effects: firstly, we can't skip any Questions; and secondly, a Question can be answered at most once. Our treatment of bracketing is rather cursory: the formulation of the condition (and the proof that it's robust) closely mirrors that of the visibility condition; indeed, it's slightly simpler.

Aside The bracketing condition is comprehensively studied in Jim Laird's PhD thesis [56] where he considers a hierarchy of notions of bracketing and makes connections to both computation and (classical) logic.

Specifically, he shows that relaxing the bracketing condition leads to fully abstract models of languages with control operators such as `call/cc` and Felleisen's $\mathcal{C}$ operator. On the logical side, the definability results that lead to full abstraction are related to Peirce's law, $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$, and to "double negation" translations of classical logic into intuitionistic logic.

Player bracketing We must now formalize what we mean by "answering" an "as yet unanswered" Question. We say that an Answer a ***answers*** a Question q in a justified string $s \in J_A$ iff a is justified by q in s. Given a string $s \in M_A^\star$, we define the ***pending question*** of s to be the rightmost occurrence of s that's an Opponent Question but which isn't answered in s.

A strategy σ on an arena A satisfies ***Player bracketing*** (we'll usually shorten this to P-***bracketing***) iff for all $sab \in \sigma$,

$$\text{if } b \in A_A \text{ then } b \text{ answers the pending question of } \ulcorner sa \urcorner.$$

Notice that we *don't* insist that b answers the pending question of sa. To see why, consider the following play of $\mathsf{id}_\mathbf{C}$...

$$\begin{array}{ccc}
\mathbf{C} & \Rightarrow & \mathbf{C} \\
 & & \mathbf{q} \\
\mathbf{q} & & \\
 & & q \\
q & & \\
\mathbf{a} & & \\
 & & \mathbf{a}
\end{array}$$

...where the moves in bold make up one of the (two) threads. This play is *necessarily* in $\mathsf{id}_\mathbf{C}$ but, if we were to insist that all Player Answers answer the pending question of the entire history, Player's last move here would manifestly violate the bracketing condition. The crucial point is that, with her last move, Opponent has "selected" one of the two active threads to continue in; in order to satisfy the bracketing condition at this point, Player must now answer the pending question of *this* thread, not the pending question of the overall play (which could, as here, be in another thread entirely).

Robustness of P-bracketing It's not too hard to adapt the proof of robustness of P-visibility to show that P-bracketing is also a robust constraint: we proceed by defining a "truncated P-view" where we stop once an Opponent Question is reached; we then extend this notion of view to legal interactions and prove the analogues of lemmas 3.4.1 and 3.4.2. Robustness is then an easy corollary.

3.4.3 Determinism

We now turn to an example of a coherence condition on strategies. A strategy is ***deterministic*** iff it satisfies

$$(sab \in \sigma \wedge sac \in \sigma) \Rightarrow sab = sac.$$

The force of this is that, at each point, if the strategy has a response then it consists of a unique choice of move *and* justification pointer.

Given an arena A, we can define a binary relation on L_A^{even} by:

$$s \frown t \text{ iff the longest common prefix of } s \text{ and } t \text{ has even length.}$$

This relation is reflexive and symmetric (but not at all transitive) and is known as ***coherence***. Clearly, a strategy σ is deterministic if, and only if, for all $s, t \in \sigma$, s and t are coherent, *i.e.* σ is a clique.

It's easy to see that the composite of two (composable) deterministic strategies is itself deterministic. All previous games models, including those for imperative features such as references and continuations, have imposed determinism as a global constraint on strategies. It's this condition that we're interested in relaxing.

3.4.4 Innocence

The notion of P-view described earlier was originally designed to describe the "currently relevant" part of a play, in the sense that a "good" strategy should base its decision on how to continue solely on the P-view of the play to date. This idea of strategies operating on partial information was motivated by the search for a denotational characterization of the PCF-definable functionals: if a strategy could "see" the entire history of play then its behaviour could potentially depend on *intensional* factors such as the order in which a "function" had evaluated its arguments. Such things cannot be expressed in PCF so it was necessary to restrict attention to the class of so-called *innocent* strategies.

Definition First of all, suppose we have $sab \in L_A^{\mathsf{even}}$ which is P-vis at b and $ta \in L_A^{\mathsf{odd}}$ such that $\ulcorner sa \urcorner = \ulcorner ta \urcorner$. There must be a unique way to extend ta with the move b such that $\ulcorner sab \urcorner = \ulcorner tab \urcorner$; we denote this extension by $\mathsf{match}(sab, ta)$. (All this is really saying is that the justifier of b in tab is the same as the justifier of b in sab; this is okay because the latter occurs in $\ulcorner sa \urcorner$, by P-visibility, and $\ulcorner ta \urcorner = \ulcorner sa \urcorner$, by assumption.)

Now, if σ is a strategy for an arena A satisfying determinism and P-visibility then we say that σ is ***innocent*** iff

$$(sab \in \sigma \wedge t \in \sigma \wedge ta \in L_A \wedge \ulcorner sa \urcorner = \ulcorner ta \urcorner) \Rightarrow \mathsf{match}(sab, ta) \in \sigma.$$

This is our first example of a liveness constraint. It is perhaps best understood by considering a simple example of a strategy that violates the condition. One such strategy is the following one for the arena $(\mathbf{B} \Rightarrow \mathbf{B}) \Rightarrow \mathbf{B}$:

$$
\begin{array}{ccccc}
(\mathbf{B} & \Rightarrow & \mathbf{B}) & \Rightarrow & \mathbf{B} \\
 & & & & \mathsf{q} \\
 & & \mathsf{q} & & \\
\mathsf{q} & & & & \\
\mathsf{tt} & & & & \\
 & & \mathsf{ff} & & \\
 & & & & \mathsf{tt}
\end{array}
$$

Intuitively, it ought to represent a "process" that takes, as its input, a "function" from Booleans to Booleans, applies this input to 'true' and returns 'true' if the result of this application is 'false'. The problem (as far as innocence is concerned) with this strategy is that it's sensitive to whether or not its input is "strict" so that, if we supply a constant function as input, it will fail to produce any output:

$$
\begin{array}{ccccccccccc}
\mathbf{1} & \Rightarrow & (\mathbf{B} & \Rightarrow & \mathbf{B}) & ; & (\mathbf{B} & \Rightarrow & \mathbf{B}) & \stackrel{\sigma}{\Rightarrow} & \mathbf{B} \\
 & & & & & & & & & & \mathsf{q} \\
 & & & & & & & & \mathsf{q} & & \\
 & & & & \mathsf{q} & & & & & & \\
 & & & & \mathsf{ff} & & & & & & \\
 & & & & & & & & \mathsf{ff} & &
\end{array}
$$

At this point, σ is completely stuck as it has no response to qqff. We can now see where the liveness condition bites: we have $\mathsf{qqqttfftt} \in \sigma$, $\mathsf{qq} \in \sigma$ and $\ulcorner \mathsf{qqqttff} \urcorner = \mathsf{qqff} = \ulcorner \mathsf{qqff} \urcorner$, but $\mathsf{qqfftt} = \mathsf{match}(\mathsf{qqqttfftt}, \mathsf{qqff}) \notin \sigma$.

The subcategory of innocent strategies It's not too hard to see that, for any arena A, id_A is an innocent strategy. More generally, any "essentially copycatting" strategy—such as Δ_A or $\mathsf{assoc}_{A,B,C}$—is innocent. (In fact, all these strategies are *history-free* in the sense that Player's choice of move and justification pointer is entirely determined by the last move that Opponent played. We will speak more of history-free strategies in chapter 5.)

We already know that deterministic strategies and P-vis strategies are (independently) closed under composition. So, to show that the innocent strategies form a subcategory of $\mathbf{G}$, we just have to check that the liveness condition is preserved by composition. We postpone the proof of this fact until §3.7 where we'll see how the assumption of determinism is vital.

In the meantime, we note that, since all the required natural isomorphisms are induced by innocent strategies, the category of innocent strategies is a sub-SMCC of $\mathbf{G}$.

3.5 A Cartesian closed category

3.5.1 Single-threaded strategies

In general, a legal play in a strategy can contain several initial occurrences, giving rise to several distinct threads of play. We might reasonably expect the behaviour of a strategy to be independent of the particular thread it's currently playing in; we call such strategies *single-threaded* or *thread-independent*. Before we examine this class of strategies, we first give a couple of examples of how a strategy can violate this condition.

Our first example is the "counter" (or "clock") strategy on the arena **N** given by $\{\varepsilon, q1, q1q2, q1q2q3, \ldots\}$ that returns the number of times the initial Question has been asked so far. The behaviour of this strategy in a given thread is clearly dependent on its behaviour in other threads. But usually we think of a "program" of natural number type as representing a fixed value and no program implementing the counter strategy can possibly have this property. (However, we should note that by "fixed value" we don't necessarily mean deterministic: the strategy $\{\varepsilon, q2, q3, q2q2, q2q3, q3q2, q3q3, \ldots\}$ which, at every stage, chooses randomly between outputting 'two' or outputting 'three' could be considered to be a value since it does at least behave uniformly, if a little unpredictably.)

Our second example is a "one-off" strategy such as $\{\varepsilon, q5\}$ again for the arena **N**. This responds only to the first (or, more generally, we could have a strategy that responds to the first n) initial Question(s). Even if the strategy's behaviour is uniform while it lasts, in the end we must regard it as non-uniform simply because it won't continue indefinitely.

Why should we disallow such strategies? One way to think of it is that different threads in a strategy correspond to different occurrences of a term and so they really ought to all have the same (potential) behaviours. This leads us to consider the following class of single-threaded strategies whose behaviour depends only on one thread at a time.

Definition The definition of single-threading comes in two parts. First of all, we have a sanity condition which resembles P-visibility. Formally, we say that a strategy σ on an arena A is ***well-threaded*** iff

$$\text{for all } sab \in \sigma, \text{ the justifier of } b \text{ occurs in } \lceil sa \rceil.$$

Now, given $sab \in L_A^{\mathsf{even}}$ which is well-threaded at b and $ta \in L_A^{\mathsf{odd}}$ such that $\lceil sa \rceil = \lceil ta \rceil$, we denote by $\mathsf{Match}(sab, ta)$ the unique $tab \in L_A$ which is well-threaded at b satisfying $\lceil sab \rceil = \lceil tab \rceil$.

If σ is well-threaded, we say that it's a ***single-threaded*** strategy iff

$$(sab \in \sigma \wedge t \in \sigma \wedge ta \in L_A \wedge \lceil sa \rceil = \lceil ta \rceil) \Rightarrow \mathsf{Match}(sab, ta) \in \sigma.$$

The similarity between the definitions of single-threading and innocence is not superficial. Recall that the P-view $\ulcorner sa \urcorner$ of a justified string $sa \in J_A^{odd}$ satisfying P-visibility is a subsequence of the current thread $\lceil sa \rceil$. This means that any P-vis strategy is automatically well-threaded. Of course, this doesn't imply that such a strategy is single-threaded. After all, both of the above examples satisfy P-visibility. However, it is the case that every innocent strategy is single-threaded. This is a consequence of the fact that, if $\lceil sa \rceil = \lceil ta \rceil$ then $\ulcorner sa \urcorner = \ulcorner ta \urcorner$.

Aside Note that both of the above examples fail to be single-threaded (although both *are* well-threaded) since they have a *temporal* element to their behaviour: the output of the counter is clearly dependent on the "time" at which output is requested; in the case of the second strategy, the question is not *what* to output but *whether* to output at all. Otherwise put, a single-threaded strategy represents a process with no internal time: different threads of the strategy are *independent* copies of the same underlying single thread of possibilities.

3.5.2 Comonoid homomorphisms

In the above informal discussion, we alluded—on a number of occasions—to the idea that single-threaded strategies behave "uniformly" across different threads or that different threads are "independent". We now make this idea precise by showing that the single-threaded strategies correspond exactly to the *comonoid homomorphisms*, *i.e.* those arrows $f : A \to B$ that satisfy the following equations.

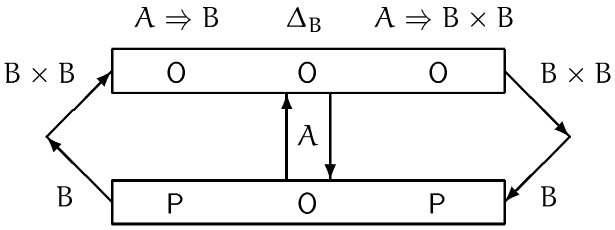
$$\begin{tikzcd} A \arrow[r, "\Delta_A"] \arrow[d, "f"'] & A \times A \arrow[d, "f \times f"] \\ B \arrow[r, "\Delta_B"'] & B \times B \end{tikzcd} \qquad \begin{tikzcd} A \arrow[d, "f"'] \arrow[dr, "!_A"] & \\ B \arrow[r, "!_B"'] & \mathbf{1} \end{tikzcd}$$

In fact, since $\mathbf{1}$ is a terminal object in our basic category $\mathbf{G}$, the triangular diagram is satisfied by *all* strategies $\sigma : A \Rightarrow B$. So we only need to check that a strategy is single-threaded if, and only if, it makes the square diagram commute.

Lemma 3.5.1 If σ is a single-threaded strategy on an arena A, $s \in \sigma$ and I is a set of initial occurrences of s then $s \upharpoonright I \in \sigma$.

Proof By induction on $|s|$. The base case is trivial. If $sab \in \sigma$ then we have $s \in \sigma$ and, by the inductive hypothesis, $s \upharpoonright I \in \sigma$. Since σ is well-threaded, we either have $sab \upharpoonright I = s \upharpoonright I \cdot ab$ or $sab \upharpoonright I = s \upharpoonright I$. In the latter case, there's nothing more to show. In the former case, we have $sab \in \sigma$, $s \upharpoonright I \in \sigma$ and $\lceil sa \rceil = \lceil s \upharpoonright I \cdot a \rceil$ and as σ is single-threaded, we have $s \upharpoonright I \cdot ab \in \sigma$. ■

We now embark on a sequence of technical lemmas that characterize the kinds of plays that can arise in $\sigma\,;\Delta_B$ and $\Delta_A\,;(\sigma \times \sigma)$ for single-threaded $\sigma : A \Rightarrow B$. Given a legal interaction $u \in \mathsf{int}(A, B, B \times B)$ such that $u \upharpoonright B, B \times B \in \Delta_B$, we will sometimes refer to the left hand copy of B in $B \times B$ as B_ℓ and to the right hand copy as B_r. Legal interactions of this form can be described by a specialized state diagram:

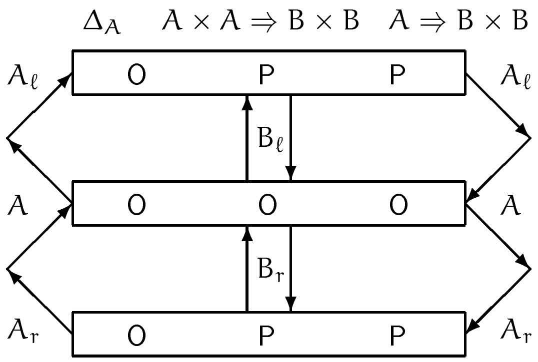


We define a partition $\{L, R\}$ of the initial occurrences of $u \upharpoonright A, B$ by setting $m \in L$ iff m is justified in u by an initial occurrence in B_ℓ; otherwise $m \in R$. Let $s = u \upharpoonright A, B \times B$. The initial occurrences of s in B_ℓ are denoted by ℓ; similarly define r to be those coming from B_r. The legal play s is easily seen to be related to the "innards" of u by: $s \upharpoonright \ell = (u \upharpoonright A, B) \upharpoonright L$ and $s \upharpoonright r = (u \upharpoonright A, B) \upharpoonright R$.

Lemma 3.5.2 If $\sigma : A \Rightarrow B$ is single-threaded and $s \in L_{A \Rightarrow (B \times B)}$ then $s \in \sigma\,;\Delta_B$ if, and only if, $s \upharpoonright \ell \in \sigma$ and $s \upharpoonright r \in \sigma$.

Proof Suppose, firstly, that $s \in \sigma\,;\Delta_B$. Let $u \in \sigma \parallel \Delta_B$ such that $u \upharpoonright A, B \times B = s$. With the partition $\{L, R\}$ defined above, we apply lemma 3.5.1 to get $(u \upharpoonright A, B) \upharpoonright L \in \sigma$ and so $s \upharpoonright \ell \in \sigma$. The case for $s \upharpoonright r$ is similar. For the converse, we proceed by induction on the length of s. The base case is trivial. So suppose that $sab \in L_{A \Rightarrow (B \times B)}$ such that $sab \upharpoonright \ell \in \sigma$ and $sab \upharpoonright r \in \sigma$. Since σ is single-threaded, the occurrences a and b are hereditarily justified by the same initial occurrence m so, WLOG, suppose $sab \upharpoonright \ell = (s \upharpoonright \ell)ab$ and $sab \upharpoonright r = s \upharpoonright r$. By (**s2**) we have $s \upharpoonright \ell \in \sigma$ and, by the inductive hypothesis, $s \in \sigma\,;\Delta_B$. Let $u \in \sigma \parallel \Delta_B$ be a witness for this. We now know that $u \upharpoonright A, B \in \sigma$ and, since $s \upharpoonright \ell = (u \upharpoonright A, B) \upharpoonright L$, we have $(u \upharpoonright A, B)a \in L_{A \Rightarrow B}$ where the final occurrence a is justified as in $(s \upharpoonright \ell)a$. Hence $\lceil (u \upharpoonright A, B)a \rceil = \lceil (s \upharpoonright \ell)a \rceil$ and, by single-threading of σ, we have $\mathsf{Match}((s \upharpoonright \ell)ab, (u \upharpoonright A, B)a) \in \sigma$. Thus we can extend u to witness $sab \in \sigma\,;\Delta_B$. For example, if a is a move of B and b is a move of A (!) then $v = uaab \in \sigma \parallel \Delta$, where $v \upharpoonright A, B = \mathsf{Match}((s \upharpoonright \ell)ab, (u \upharpoonright A, B)a)$, *i.e.* the move a is copied by Δ into "the middle" to which σ responds by playing b in "the outside". ∎

Let $u \in \mathsf{int}(A, A \times A, B \times B)$ be a legal interaction such that $u \upharpoonright A, A \times A \in \Delta_A$. We extend the notation previously introduced, denoting by A_ℓ the left hand copy of $A \times A$, *etc.* If, additionally, we have that $u \upharpoonright A_\ell, B_\ell \in L_{A \Rightarrow B}$ and $u \upharpoonright A_r, B_r \in L_{A \Rightarrow B}$ then the form of u is substantially constrained: the switching condition in the $(A \times A) \Rightarrow (B \times B)$ component combines with the enforced switching caused by Δ_A. This is best summarized in yet another state diagram.



Lemma 3.5.3 If $u \in \mathsf{int}(A, A \times A, B \times B)$ such that $u \upharpoonright A, A \times A \in \Delta_A$, $u \upharpoonright A_\ell, B_\ell \in L_{A \Rightarrow B}$ and $u \upharpoonright A_r, B_r \in L_{A \Rightarrow B}$ then $(u \upharpoonright A, B \times B) \upharpoonright \ell = u \upharpoonright A_\ell, B_\ell$ and $(u \upharpoonright A, B \times B) \upharpoonright r = u \upharpoonright A_r, B_r$.

Proof By induction on $|u|$. The base case is trivial. Otherwise, we proceed by cases according to which arena the last occurrence of u comes from. First of all, let $u = u'm$ where m is from $B \times B$ and, WLOG, suppose m is played in B_ℓ. So $u \upharpoonright A_\ell, B_\ell = (u' \upharpoonright A_\ell, B_\ell)m = ((u' \upharpoonright A, B \times B) \upharpoonright \ell)m = (u \upharpoonright A, B \times B) \upharpoonright \ell$, where the penultimate step follows by the inductive hypothesis. The other restriction is trivial: $u \upharpoonright A_r, B_r = u' \upharpoonright A_r, B_r = (u' \upharpoonright A, B \times B) \upharpoonright r = (u \upharpoonright A, B \times B) \upharpoonright r$. The other case is where m occurs in the $A \Rightarrow (A \times A)$ component. In this case, $u = u'mm$ where one occurrence of m is in A and the other is in $A \times A$. We assume, WLOG, that m occurs in A_ℓ. So we have $u \upharpoonright A_\ell, B_\ell = (u' \upharpoonright A_\ell, B_\ell)m = ((u' \upharpoonright A, B \times B) \upharpoonright \ell)m$, by the inductive hypothesis. Since $u \upharpoonright A, A \times A \in \Delta_A$, we have $((u' \upharpoonright A, B \times B) \upharpoonright \ell)m = (u \upharpoonright A, B \times B) \upharpoonright \ell$ and we're done. The case of the other restriction is similar and easier. ■

Lemma 3.5.4 If $\sigma : A \Rightarrow B$ then $s \in \Delta_A ; (\sigma \times \sigma)$ if, and only if, $s \in L_{A \Rightarrow (B \times B)}$, $s \upharpoonright \ell \in \sigma$ and $s \upharpoonright r \in \sigma$.

Proof Suppose first that $s \in \Delta_A ; (\sigma \times \sigma)$. Therefore $s \in L_{A \Rightarrow (B \times B)}$. We must have a witness $u \in \Delta_A \parallel (\sigma \times \sigma)$ for s and, by definition, $u \upharpoonright A_\ell, B_\ell \in \sigma$ and $u \upharpoonright A_r, B_r \in \sigma$. Clearly u satisfies the hypotheses of lemma 3.5.3 so we have $s \upharpoonright \ell = u \upharpoonright A_\ell, B_\ell$ and $s \upharpoonright r = u \upharpoonright A_r, B_r$ and we're done.

We prove the converse by induction on $|s|$. The base case is trivial so suppose we have some $sab \in L_{A \Rightarrow (B \times B)}$ such that $sab \upharpoonright \ell \in \sigma$ and $sab \upharpoonright r \in \sigma$. It must be the case that $sab \upharpoonright \ell$ and $sab \upharpoonright r$ both have even length so sab must also have even length and the occurrences a and b are hereditarily justified either both in ℓ or both in r. WLOG, let's assume that $sab \upharpoonright \ell = (s \upharpoonright \ell)ab$ and $sab \upharpoonright r = s \upharpoonright r$. By (**s2**) we have $s \upharpoonright \ell \in \sigma$ and invoking the inductive hypothesis yields $s \in \Delta_A ; (\sigma \times \sigma)$. This must have a witness $u \in \Delta_A \parallel (\sigma \times \sigma)$. We'd like to extend this to some $v \in \Delta_A \parallel (\sigma \times \sigma)$ witnessing sab in $\Delta_A ; (\sigma \times \sigma)$. Since $sab \upharpoonright \ell = (u \upharpoonright A_\ell, B_\ell)ab$ we can straightforwardly do this by cases on where a and b occur. For example, if they both occur in A then $uaabb \in \Delta_A \parallel (\sigma \times \sigma)$ does the job. ■

And so, at last...

Proposition 3.5.5 A strategy $\sigma : A \Rightarrow B$ is single-threaded if, and only if, it's a comonoid homomorphism.

Proof The "only if" part follows immediately from lemmas 3.5.2 and 3.5.4. For the "if" half, let $\sigma : A \Rightarrow B$ be a comonoid homomorphism and suppose that $sab \in \sigma$. First of all, we want to show that the justifier of b occurs in $\lceil sa \rceil$, *i.e.* that σ is well-threaded. To this end, consider the legal interaction $u \in \mathsf{int}(A, B, B \times B)$ such that $u \upharpoonright A, B = sab$ and $u \upharpoonright B, B \times B \in \Delta_B$ where (the Opponent in) Δ_B plays the hereditary justifier of a (call it m) in B_ℓ and all other initial occurrences in B_r. So $u \upharpoonright A, B \times B \in \sigma ; \Delta_B = \Delta_A ; (\sigma \times \sigma)$. This in turn has a witness $v \in \mathsf{int}(A, A \times A, B \times B)$. This v satisfies the hypotheses of lemma 3.5.3 so $(v \upharpoonright A, B \times B) \upharpoonright \ell = v \upharpoonright A_\ell, B_\ell \in \sigma$. But $(v \upharpoonright A, B \times B) \upharpoonright \ell = (u \upharpoonright A, B \times B) \upharpoonright \ell = sab \upharpoonright m$. So $sab \upharpoonright m \in \sigma$ and, since $a \in sab \upharpoonright m$, we must therefore have $b \in sab \upharpoonright m$ and hence the justifier of b occurs in $\lceil sa \rceil$.

Suppose now that we also have some $t \in \sigma$ and $ta \in L_A$ such that $\lceil sa \rceil = \lceil ta \rceil$. We know that $\mathsf{Match}(sab, ta)$ is the unique well-threaded extension of ta such that $\lceil sab \rceil = \lceil tab \rceil$. So $\mathsf{Match}(sab, ta) \upharpoonright m = sab \upharpoonright m$ and $\mathsf{Match}(sab, ta) \upharpoonleft m = t \upharpoonleft m$. Consider the legal interaction $u' \in \mathsf{int}(A, B, B \times B)$ such that $u' \upharpoonright A, B = t$ and $u' \upharpoonright B, B \times B \in \Delta_B$ where (the Opponent in) Δ_B plays all the initial occurrences of t except m in B_r; if $m \in t$ then this is played in B_ℓ. By the same argument as before, we have some $v' \in \mathsf{int}(A, A \times A, B \times B)$ witnessing $u' \upharpoonright A, B \times B$ in $\Delta_A ; (\sigma \times \sigma)$. By lemma 3.5.3, we have $v' \upharpoonright A_r, B_r = (u' \upharpoonright A, B \times B) \upharpoonright r = t \upharpoonleft m$, so $t \upharpoonleft m \in \sigma$. Similarly, we have $s \upharpoonright m = t \upharpoonright m = v' \upharpoonright A_\ell, B_\ell \in \sigma$. We now extend v' with the final segment $a \cdots b$ of v yielding $v'' \in \mathsf{int}(A, A \times A, B \times B)$ such that $v'' \upharpoonright A_r, B_r = v' \upharpoonright A_r, B_r$ and $v'' \upharpoonright A_\ell, B_\ell = sab \upharpoonright m$. So $(v'' \upharpoonright A, B \times B) \upharpoonright \ell \in \sigma$ and $(v'' \upharpoonright A, B \times B) \upharpoonright r \in \sigma$ and, by lemma 3.5.4, $v'' \upharpoonright A, B \times B \in \Delta_A ; (\sigma \times \sigma) = \sigma ; \Delta_B$. Finally, this means that we have $u'' \in \mathsf{int}(A, B, B \times B)$ witnessing $v'' \upharpoonright A, B \times B$ in $\sigma ; \Delta_B$. So $\mathsf{Match}(sab, ta) = u'' \upharpoonright A, B \in \sigma$. ∎

Proposition 3.5.6 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are single-threaded strategies then so is $\sigma ; \tau$.

Proof We paste our diagrams together...

$$\begin{tikzcd}
A \arrow[r, "\sigma"] \arrow[d, "\Delta_A"] & B \arrow[r, "\tau"] \arrow[d, "\Delta_B"] & C \arrow[d, "\Delta_C"] \\
A \times A \arrow[r, "\sigma \times \sigma"'] & B \times B \arrow[r, "\tau \times \tau"'] & C \times C
\end{tikzcd}$$

...and applying functoriality of $\times$ gives $\Delta_A ; (\sigma ; \tau) \times (\sigma ; \tau) = \Delta_A ; (\sigma \times \sigma) ; (\tau \times \tau) = (\sigma ; \tau) ; \Delta_C$ as required. ∎

3.5.3 Thread relations

The very nature of the liveness condition for single-threaded strategies makes it inevitable that most strategies that satisfy it are both very big and contain lots of redundant information. For example, the single-threaded strategy for ‘true’ on **B** is the infinite set $\{\varepsilon, \mathsf{q}\mathsf{tt}, \mathsf{q}\mathsf{tt}\mathsf{q}\mathsf{tt}, \ldots\}$—and yet it’s clear that all possible behaviours of this strategy can be succinctly summarized as “whenever Opponent plays q, Player answers it with tt”. We formalize this idea with the notion of *thread relation* which derives from the *view functions* of [63].

Given a well-threaded strategy σ, we define $\mathsf{Rel}(\sigma)$ by:

$$\mathsf{Rel}(\sigma) = \{\varepsilon\} \cup \{sab \mid \exists tab \in \sigma.\ \lceil tab \rceil = sab\}.$$

Note that, if $sab \in \mathsf{Rel}(\sigma)$ then sa is a ***well-opened***, odd-length legal play, *i.e.* it has exactly one initial occurrence. As defined, $\mathsf{Rel}(\sigma)$ isn’t actually a relation; but we think of it as relating sa to b. The reason for setting it up this way is that we need to know which occurrence of sa justifies b in sab.

A ***thread relation*** is simply a set R of well-opened, even-length legal plays of an arena A such that $\varepsilon \in R$. Given such a thread relation, we define:

$$\begin{aligned}
T_0(R) &= \{\varepsilon\} \\
T_{n+1}(R) &= \{sab \in L_A \mid s \in T_n(R) \wedge \exists tab \in R.\ \lceil sa \rceil \cdot b = tab\} \\
\mathsf{Traces}(R) &= \bigcup_{n \in \mathbf{N}_0} T_n(R)
\end{aligned}$$

So $\mathsf{Traces}(R)$ is the biggest well-threaded strategy that can be “generated” from R. We say that R is ***saturated*** iff

$$sab \in R \text{ if, and only if, } \exists tab \in \mathsf{Traces}(R).\ \lceil tab \rceil = sab.$$

It’s easy to see that, if σ and τ are well-threaded strategies for arena A such that $\sigma \subseteq \tau$ then $\mathsf{Rel}(\sigma) \subseteq \mathsf{Rel}(\tau)$. Similarly, a straightforward induction yields, for thread relations R and S for arena A, if $R \subseteq S$ then $\mathsf{Traces}(R) \subseteq \mathsf{Traces}(S)$.

Lemma 3.5.7 If σ and R are, respectively, a well-threaded strategy and a thread relation for an arena A then $\mathsf{Traces}(\mathsf{Rel}(\sigma))$ is single-threaded, $\mathsf{Rel}(\mathsf{Traces}(R))$ is a saturated thread relation and

$$\begin{aligned}
\sigma &\subseteq \mathsf{Traces}(\mathsf{Rel}(\sigma)) \\
R &\supseteq \mathsf{Rel}(\mathsf{Traces}(R)).
\end{aligned}$$

Proof A simple induction gives us $\sigma \subseteq \mathsf{Traces}(\mathsf{Rel}(\sigma))$. As for single-threading, suppose $sab, t \in \mathsf{Traces}(\mathsf{Rel}(\sigma))$, $ta \in L_A$ and $\lceil sa \rceil = \lceil ta \rceil$. By definition, there are $n, m \in \mathbf{N}_0$ such that $sab \in T_{n+1}(\mathsf{Rel}(\sigma))$ and $t \in T_m(\mathsf{Rel}(\sigma))$. So $\lceil sa \rceil \cdot b \in \mathsf{Rel}(\sigma)$ and hence $\lceil ta \rceil \cdot b \in \mathsf{Rel}(\sigma)$ which means that $\mathsf{Match}(sab, ta) \in T_{m+1}(\mathsf{Rel}(\sigma))$. Therefore $\mathsf{Match}(sab, ta) \in \mathsf{Traces}(\mathsf{Rel}(\sigma))$ as required.

It’s easy to see that $\mathsf{Rel}(\mathsf{Traces}(R)) \subseteq R$ and, by definition, $sab \in \mathsf{Rel}(\mathsf{Traces}(R))$ if, and only if, $\exists tab \in \mathsf{Traces}(R).\ \lceil tab \rceil = sab$, so $\mathsf{Rel}(\mathsf{Traces}(R))$ is saturated. ■

A standard corollary of this lemma is that $\mathsf{Traces}(\mathsf{Rel}(-))$ is a closure operator on the well-threaded strategies. So, given a well-threaded strategy σ, we can find a single-threaded strategy $\mathsf{Traces}(\mathsf{Rel}(\sigma))$ containing it and, moreover, $\mathsf{Rel}(\sigma) = \mathsf{Rel}(\mathsf{Traces}(\mathsf{Rel}(\sigma)))$.

We now show that $\mathsf{Traces}(\mathsf{Rel}(\sigma))$ is the *least* single-threaded strategy containing σ. This yields a useful characterization of the class of single-threaded strategies.

Lemma 3.5.8 If σ and τ are, respectively, a well- and a single-threaded strategy for arena A such that $\sigma \subseteq \tau$ then $\mathsf{Traces}(\mathsf{Rel}(\sigma)) \subseteq \tau$.

Proof By induction on the length of $s \in \mathsf{Traces}(\mathsf{Rel}(\sigma))$. The base case is trivial. If $sab \in \mathsf{Traces}(\mathsf{Rel}(\sigma))$ then $s \in \mathsf{Traces}(\mathsf{Rel}(\sigma))$ and, by the inductive hypothesis, $s \in \tau$. By the definition of $\mathsf{Traces}(-)$ we have $tab \in \mathsf{Rel}(\sigma)$ such that $tab = \lceil sa \rceil \cdot b$. By the definition of $\mathsf{Rel}(-)$ we have $t'ab \in \sigma$ such that $\lceil t'ab \rceil = tab$. So $t'ab \in \tau$ and $\lceil sa \rceil = ta = \lceil t'a \rceil$ and, by single-threading, $sab = \mathsf{Match}(t'ab, sa) \in \tau$. ■

This means that, if σ is a single-threaded strategy then $\mathsf{Traces}(\mathsf{Rel}(\sigma)) \subseteq \sigma$. It's also easy to see that, for a saturated thread relation R, $R \subseteq \mathsf{Rel}(\mathsf{Traces}(R))$. Therefore we have a one-2-one correspondence between saturated thread relations and single-threaded strategies:

$$\begin{aligned} \sigma &= \mathsf{Traces}(\mathsf{Rel}(\sigma)) \\ R &= \mathsf{Rel}(\mathsf{Traces}(R)). \end{aligned}$$

Well-opened strategies We say that a strategy σ is ***well-opened*** iff every $s \in \sigma$ has exactly one initial occurrence. It's therefore clear that any well-opened strategy σ can be thought of as a thread relation. In fact, it must be a saturated thread relation since if $s \in \sigma$ then $s \in \mathsf{Traces}(\sigma)$. (This is proved by a simple induction on the length of s.)

The converse of this also holds: every saturated thread relation R is a well-opened strategy. To see this, consider $sab \in R$. We must have some $tab \in \mathsf{Traces}(R)$ such that $sab = \lceil tab \rceil$. But then there must be a prefix $t' \sqsubseteq tab$ such that $\lceil t' \rceil = s$ and, since $t' \in \mathsf{Traces}(R)$, we have $s \in R$.

We now have a one-to-one correspondence between well-opened strategies and single-threaded strategies. As the following lemma shows, this correspondence preserves the various constraints of interest.

Lemma 3.5.9 Let $\sigma : A$ be a single-threaded strategy.

1. σ is deterministic if, and only if, $\mathsf{Rel}(\sigma)$ is deterministic;

2. σ is P-vis if, and only if, $\mathsf{Rel}(\sigma)$ is P-vis;

3. σ is P-brk if, and only if, $\mathsf{Rel}(\sigma)$ is P-brk.

Proof Suppose that σ is deterministic and let $sab \in \mathsf{Rel}(\sigma)$ and $sac \in \mathsf{Rel}(\sigma)$. So we have $tab \in \sigma$ and $t'ac \in \sigma$ such that $\lceil tab \rceil = sab$ and $\lceil t'ac \rceil = sac$. Therefore $\lceil ta \rceil = \lceil t'a \rceil$. Since σ is single-threaded, we have $\mathsf{Match}(tab, t'a) \in \sigma$ and, since σ is deterministic, $\mathsf{Match}(tab, t'a) = t'ac$. This means that $\lceil t'ac \rceil = \lceil tab \rceil$ and so $sac = \lceil t'ac \rceil = \lceil tab \rceil = sab$. The reverse is trivial.

The other two cases are very similar; we show only that of the visibility condition. Suppose that σ satisfies Player visibility and $sab \in \mathsf{Rel}(\sigma)$. So we have some $tab \in \sigma$ such that $sab = \lceil tab \rceil$. Since σ is P-vis, we know that the justifier of b occurs in $\ulcorner ta \urcorner = \ulcorner \lceil ta \rceil \urcorner = \ulcorner sa \urcorner$. The other direction is trivial. ■

Note that we really make use of the fact that σ is single-threaded in the case of determinism. To see why, recall that the counter strategy on **N** is deterministic—but its thread relation is anything but.

3.5.4 Product as product

We now honour the promise made in §3.3 to introduce a subcategory of **G** where the $\times$ constructor is a genuine categorical product. The subcategory in question is that generated by the class of single-threaded strategies. We know from proposition 3.5.6 that these are closed under composition and, since all innocent strategies are single-threaded and the identities are innocent, we can deduce that this is indeed a subcategory; we'll denote it by **C**. This subcategory is also a sub-SMCC of **G** since all the necessary structure is provided by innocent strategies and, therefore, **C** will be a Cartesian closed category (CCC).

To see that $\times$ restricts to be a genuine categorical product in **C**, first of all note that we have evident projection strategies $\mathsf{fst}_{A,B} : A \times B \Rightarrow A$ and $\mathsf{snd}_{A,B} : A \times B \Rightarrow B$ which copycat between the RHS and the appropriate part of the LHS. Of course, these arrows exist for more general reasons: in any monoidal category where the tensor unit **I** is terminal, we can define the first projection $\pi_{A,B}$ by $(1_A \times !_B) ; \rho_A$ and similarly for the second projection $\pi'_{A,B}$. See [53] for more.

If $s \in L_{A \Rightarrow (B \times C)}$ then we denote by ℓ the set of initial occurrences of s from B; similarly, r is those initial occurrences from C. Given $\sigma : A \Rightarrow B$ and $\tau : A \Rightarrow C$, we define the ***pairing*** of σ and τ as a strategy for $A \Rightarrow (B \times C)$ by:

$$\langle \sigma, \tau \rangle = \{ s \in L_{A \Rightarrow (B \times C)} \mid s \restriction b \in \sigma \wedge s \restriction c \in \tau \},$$

In other words, $\langle \sigma, \tau \rangle$ plays like σ if the current thread started in B and like τ if it started in C.

Lemma 3.5.10 Suppose that $\upsilon : A \Rightarrow (B \times C)$ is a single-threaded strategy and let $s \in L_{A \Rightarrow (B \times C)}$. We have $s \in \upsilon$ if, and only if, $s \restriction \ell \in \upsilon ; \mathsf{fst}_{B,C}$ and $s \restriction r \in \upsilon ; \mathsf{snd}_{B,C}$.

Proof The "only if" direction follows fairly immediately from lemma 3.5.1. The converse is an easy induction on the length of s. ■

We therefore have that $\langle\sigma,\tau\rangle;\mathsf{fst}_{B,C} = \sigma$ and $\langle\sigma,\tau\rangle;\mathsf{snd}_{B,C} = \tau$. Moreover, the above lemma says that $\langle\sigma,\tau\rangle$ is the unique single-threaded strategy that satisfies these equations, so we have a product diagram in **C**.

$$A \xleftarrow{\mathsf{fst}_{A,B}} A \times B \xrightarrow{\mathsf{snd}_{A,B}} B$$

3.5.5 Single-threaded strategies & lifting

Recall the counter strategy on **N** from our earlier discussion. Imagine we could somehow gather all of its threads into a bag; we could think of this bag as some kind of "super-thread" of **N**. There is now a sense in which the counter strategy is "single-super-threaded": we could have any number of bags; each would be a counter entirely independent of all the others.

Such a scheme seems fraught with technical complications. However, we can obtain exactly this effect simply by considering the counter as a strategy on $\mathbf{N}_\perp$: the effect of the initial protocol is to "package up" all the counter's threads into a single thread; we can get a new, independent counter by re-asking the initial Question of the lifting protocol. The thread relation of this strategy is given by:

$$\{\varepsilon, ?\checkmark, ?\checkmark q1, ?\checkmark q1q2, ?\checkmark q1q2q3, \ldots\}.$$

The intuition is that, "in between" the initial Question $?$ and the Answer $\checkmark$, the process allocates itself some private storage which it can subsequently utilize, *e.g.* our counter could have a single variable which it increments each time it's asked for output. So we can use lifting to recover "dynamic" (*i.e.* temporally dependent) behaviours in single-threaded strategies. This is suggestive of the use, pioneered by Moggi [68], of monads—such as lifting on a conventional category of (pre)domains—to distinguish (at the level of types, at least in the metalanguage) between *values* (static, timeless processes) and *computations* (dynamic processes such as our counter). We will see in the next subsection that lifting can be extended to a Kleisli triple on the category of single-threaded strategies, thus cementing this intuitive connection.

3.5.6 Lifting as a Kleisli triple

Kleisli triples The categorical concept of ***monad*** (see [59]) has an alternative presentation as a ***Kleisli triple***. Given a category $\mathcal{C}$, a Kleisli triple on $\mathcal{C}$ consists of three mappings:

- for each object A, an object $T(A)$, usually just written TA;
- for each object A, an arrow $\eta_A : A \to TA$;
- for each arrow $f : A \to TB$, an arrow $f^* : TA \to TB$.

We will sometimes refer to $\eta(-)$ and $(-)^*$ as the *unit* and *promotion* mappings. They are required to satisfy three axioms. Let $f : A \to TB$ and $g : B \to TC$ be arrows of $\mathcal{C}$.

(kt1) $f\,;\eta_A^* = f$;

(kt2) $\eta_A\,; f^* = f$;

(kt3) $f^*\,; g^* = (f\,; g^*)^*$.

We can now build the ***Kleisli category*** $\mathcal{C}_T$ with the same objects as $\mathcal{C}$ and where $\mathcal{C}_T(A, B) = \mathcal{C}(A, TB)$. If we have two arrows, $f : A \to B$ and $g : B \to C$, of $\mathcal{C}_T$, we define their composite $f \fatsemi g : A \to C$ by $f\,; g^*$. The identity for an object A is η_A. The axioms (**kt1-3**) are just sufficient to prove that the identities are the identities and that composition is associative.

Legal interactions revisited We extend the notions of hereditary justification and current thread to legal interactions as follows. Let $u \in \mathsf{int}(A, B, C)$. If $m \in u$ occuring in B or C, its hereditary justifier in u is the same as its hereditary justifier in $u \restriction B, C$. If m occurs in A, let n be its hereditary justifier in $u \restriction A, B$; then the hereditary justifier of m in u is the initial occurrence in C that justifies n in $u \restriction B, C$. If n is an initial occurrence in u, $u \restriction n$ denotes the subsequence of u consisting of all (occurrences of) moves hereditarily justified by n.

Lemma 3.5.11 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are single-threaded strategies then $\mathsf{Rel}(\sigma\,;\tau) = \sigma\,;\mathsf{Rel}(\tau)$.

Proof Suppose that $s \in \sigma\,;\mathsf{Rel}(\tau)$. It must have a witness $u \in \sigma \parallel \mathsf{Rel}(\tau)$ such that $u \restriction B, C \in \mathsf{Rel}(\tau)$. This means that $u \restriction B, C \in \tau$ and, hence, we have $u \in \sigma \parallel \tau$ and $s \in \sigma\,;\tau$. But since s is well-opened, this gives $s \in \mathsf{Rel}(\sigma\,;\tau)$.

For the converse, we know that, if $s \in \mathsf{Rel}(\sigma\,;\tau)$ then there's some $t \in \sigma\,;\tau$ such that $\lceil t \rceil = s$. In other words, $s = t \restriction m$ for some initial occurrence $m \in t$. The play t must have a witness $v \in \sigma \parallel \tau$. Consider $v \restriction m$. Applying lemma 3.5.1 we have $(v \restriction m) \restriction B, C = (v \restriction B, C) \restriction m \in \tau$ and $(v \restriction m) \restriction A, B = (v \restriction A, B) \restriction I \in \sigma$, where I is the set of initial occurrences of $v \restriction A, B$ that are justified by m in v. We also have $(v \restriction m) \restriction A, C = (v \restriction A, C) \restriction m = t \restriction m = s \in L_{A \Rightarrow C}$ so that $v \restriction m \in \mathsf{int}(A, B, C)$. Since $v \restriction m$ has only one initial occurrence, $v \restriction m \in \sigma \parallel \mathsf{Rel}(\tau)$. We now calculate $s = t \restriction m = (v \restriction A, C) \restriction m = (v \restriction m) \restriction A, C \in \sigma\,;\mathsf{Rel}(\tau)$. ■

Lifting We now return to the main theme of this subsection by defining a Kleisli triple for lifting. The mapping on objects is simply $A \mapsto A_\perp$. It therefore remains to specify the unit and promotion mappings. To this end, given an arena A, we define a well-opened strategy $\mathsf{up}_A = \{\varepsilon\} \cup \{?\checkmark s \mid s \in \mathsf{id}_A\}$ and, given a well-opened strategy $\sigma : A \Rightarrow B_\perp$, we define $\sigma^\star : A_\perp \Rightarrow B_\perp$ by $\sigma^\star = \{\varepsilon, ??\} \cup \{??\checkmark s \mid ?s \in \sigma\}$.

Suppose we have well-opened strategies $\sigma : A \Rightarrow B_\perp$ and $\tau : B \Rightarrow C_\perp$. We have the following useful properties:

- $\sigma\,;\mathsf{up}^\star_B = \sigma$;
- $\mathsf{up}_A\,;\sigma^\star = \sigma$;
- $\sigma^\star\,;\tau^\star = (\sigma\,;\tau^\star)^\star$.

The easiest way to see these equalities is by considering the following "schematic" presentations of the various composites:

$$
\begin{array}{ccccc}
 & \sigma & & \mathsf{up}^\star_B & \\
A & \Rightarrow & B_\perp & \Rightarrow & B_\perp \\
 & & & & ? \\
 & & ? & & \\
\vdots & & & & \\
 & & \checkmark & & \\
 & & & & \checkmark \\
 & \vdots & & \vdots &
\end{array}
\qquad
\begin{array}{ccccc}
 & \mathsf{up}_A & & \sigma^\star & \\
A & \Rightarrow & A_\perp & \Rightarrow & B_\perp \\
 & & & & ? \\
 & & ? & & \\
 & & \checkmark & & \\
 & \vdots & & \vdots &
\end{array}
\qquad
\begin{array}{ccccc}
 & \sigma^\star & & \tau^\star & \\
A_\perp & \Rightarrow & B_\perp & \Rightarrow & C_\perp \\
 & & & & ? \\
 & & ? & & \\
? & & & & \\
\checkmark & & & & \\
 & \vdots & & & \\
 & & \checkmark & & \\
 & & & \vdots & \\
 & & & & \checkmark \\
 & \vdots & & \vdots &
\end{array}
$$

We now exploit the one-to-one correspondence between well-opened and single-threaded strategies to define the unit and promotion mappings for our proto-Kleisli triple:

- for an arena A, $\eta_A = \mathsf{Traces}(\mathsf{up}_A)$;
- for a strategy $\sigma : A \Rightarrow B_\perp$, $\sigma^* = \mathsf{Traces}(\mathsf{Rel}(\sigma)^\star)$.

We now prove that we have a Kleisli triple. First of all, we prove a stronger form of lemma 3.5.11 in the case where σ and τ are "Kleisli arrows".

Lemma 3.5.12 If $\sigma : A \Rightarrow B_\perp$ and $\tau : B \Rightarrow C_\perp$ are single-threaded strategies then $\sigma\,;\mathsf{Rel}(\tau)^\star = \mathsf{Rel}(\sigma)\,;\mathsf{Rel}(\tau)^\star$.

Proof The right-to-left inclusion is trivial since $\mathsf{Rel}(\sigma) \leqslant \sigma$. For the converse, suppose that $s \in \sigma\,;\mathsf{Rel}(\tau)^\star$. It must have a witness $u \in \sigma \parallel \mathsf{Rel}(\tau)^\star$ such that $u \upharpoonright B_\perp, C \in \mathsf{Rel}(\tau)^\star$. But there can be at most one occurrence of the initial move ? in $B_\perp$ and hence $u \upharpoonright A, B_\perp \in \mathsf{Rel}(\sigma)$. Therefore $u \in \mathsf{Rel}(\sigma) \parallel \mathsf{Rel}(\tau)^\star$, witnessing $s \in \mathsf{Rel}(\sigma)\,;\mathsf{Rel}(\tau)^\star$. ■

It's now easy to see that (**kt1**) and (**kt2**) hold. Let $\sigma : A \Rightarrow B_\perp$ be a single-threaded strategy. By definition, $\sigma ; \eta_B^* = \sigma ; \mathsf{Traces}(\mathsf{Rel}(\eta_B)^\star) = \sigma ; \mathsf{Traces}(\mathsf{up}_B^\star)$, so that:

$$\begin{aligned}\mathsf{Rel}(\sigma ; \eta_B^*) &= \sigma ; \mathsf{up}_B^\star \\ &= \mathsf{Rel}(\sigma) ; \mathsf{up}_B^\star \\ &= \mathsf{Rel}(\sigma).\end{aligned}$$

Applying $\mathsf{Traces}(-)$ to each side gives the desired result; (**kt2**) is similar. For (**kt3**), we have $\sigma^* ; \tau^* = \mathsf{Traces}(\mathsf{Rel}(\sigma)^\star) ; \mathsf{Traces}(\mathsf{Rel}(\tau)^\star)$. We now calculate:

$$\begin{aligned}\mathsf{Rel}(\sigma^* ; \tau^*) &= \mathsf{Traces}(\mathsf{Rel}(\sigma)^\star) ; \mathsf{Rel}(\tau)^\star \\ &= \mathsf{Rel}(\sigma)^\star ; \mathsf{Rel}(\tau)^\star \\ &= (\mathsf{Rel}(\sigma) ; \mathsf{Rel}(\tau)^\star)^\star \\ &= (\sigma ; \mathsf{Rel}(\tau)^\star)^\star \\ &= \mathsf{Rel}(\sigma ; \mathsf{Traces}(\mathsf{Rel}(\tau)^\star))^\star\end{aligned}$$

Applying $\mathsf{Traces}(-)$ to either side of this yields $\sigma^* ; \tau^* = (\sigma ; \tau^*)^*$ and we're done.

In summary, we've shown that lifting is a Kleisli triple (and hence a monad) on the category of single-threaded strategies. It's not too hard to show that lifting is a ***strong*** monad, meaning that we have a natural transformation with components $\mathsf{st}_{A,B} : A \times B_\perp \to (A \times B)_\perp$, given by the evident copycat strategies, satisfying some equations relating it to the monoidal associativity and unit isomorphisms and the monadic unit and multiplication transformations; see [53] for more.

3.5.7 Order enrichment

Given an arena A, we write $\mathsf{STr}(A)$ for the set of single-threaded strategies on A. It's easy to see that, if Δ is a directed subset of $\mathsf{STr}(A)$ then $\bigsqcup \Delta$ is itself single-threaded. So $\mathsf{STr}(A)$ is a sub-CPO of $\mathsf{Str}(A)$.

Lemma 3.5.13 If σ and τ are single-threaded strategies on an arena A then $\sigma \subseteq \tau$ if, and only if, $\mathsf{Rel}(\sigma) \subseteq \mathsf{Rel}(\tau)$.

Proof We've already seen the "only if" direction. If $\mathsf{Rel}(\sigma) \subseteq \mathsf{Rel}(\tau)$ then applying lemma 3.5.8 gives $\sigma = \mathsf{Traces}(\mathsf{Rel}(\sigma)) \subseteq \mathsf{Traces}(\mathsf{Rel}(\tau)) = \tau$. ■

Armed with this, we can rework the argument of §3.3 to characterize the compact elements of $\mathsf{STr}(A)$. It's clear that any strategy σ with a finite thread relation $\mathsf{Rel}(\sigma)$ is compact. Now, for any strategy σ, we define

$$\Delta_\sigma = \{\sigma' \in \mathsf{STr}(A) \mid \mathsf{Rel}(\sigma') \subseteq^{\mathsf{fin}} \mathsf{Rel}(\sigma)\}.$$

This is a directed set with σ amongst its upper bounds. If we can show that σ is the lub of Δ_σ then we have that a strategy in $\mathsf{STr}(A)$ is compact if, and only if, its thread relation is finite.

To see that σ really is the lub, consider $s \in \sigma$. This induces a well-threaded strategy

$$\sigma_s = \{s' \in L_A^{\mathsf{even}} \mid s' \sqsubseteq s\}.$$

By lemma 3.5.7, $\overline{\sigma}_s = \mathsf{Traces}(\mathsf{Rel}(\sigma_s))$ is single-threaded and $\sigma_s \subseteq \overline{\sigma}_s$. It's clear that $\mathsf{Rel}(\sigma_s)$ is finite; but $\mathsf{Rel}(\sigma_s) = \mathsf{Rel}(\overline{\sigma}_s)$ so $\overline{\sigma}_s \in \Delta_\sigma$. Since $s \in \overline{\sigma}_s$, we have $s \in \bigsqcup \Delta_\sigma$ and hence $\sigma \subseteq \bigsqcup \Delta_\sigma$.

Lemma 3.5.14 The set $\mathsf{STr}(A)$ ordered by $\leqslant_A$ is an algebraic CPO.

3.6 Deterministic factorization

Nondeterminism, in this sense, is determinism, but with some missing parameter – its "oracle". David Park [78].

Preamble Consider the flat arena over a countable set X. There's an obvious strategy, denoted by $\top_X$, where *every* time Opponent asks the initial Question q, Player can choose *any* $x \in P_X$ with which to answer it. Such a strategy always satisfies Player visibility, Player bracketing and well-threading (in fact, these are all equivalent restrictions for legal plays on a flat arena) but obviously won't be deterministic if X has more than one element. It's also clearly single-threaded. Such strategies can be thought of as "canonical" nondeterministic strategies in a sense that will shortly be made precise.

3.6.1 Oracles & nondeterminism

Suppose we have a strategy σ on an arena A such that $sa \in \mathsf{dom}(\sigma)$. This means that Player has a choice from some non-empty set of moves: $\{b_i \in P_A \mid sab_i \in \sigma\}$. Suppose now that Player could somehow delegate the task of choosing that move to Opponent. In this situation, Opponent's response could be interpreted as *suggesting* a move for Player—so Player could behave entirely deterministically by following Opponent's advice. We could rephrase this as saying that Player "consults the Oracle" and acts according to this divination.

Even given this, we still have the question of how Opponent could possibly impart such advice. Well, since the set M_A of moves of A is, by definition, finite or countably infinite, we know that the set L_A^{even} of even-length legal plays of A is also countable. We can therefore think of "coding up" such legal plays as numbers with an injective function $\mathsf{code}_A : L_A^{\mathsf{even}} \rightarrowtail \mathbf{N}$. If we add a single P-move (representing a "request for advice") and a countable number of O-moves (representing each possible "piece of advice", *i.e.* a code for some play of A) to A, we can build a deterministic strategy $\mathsf{det}(\sigma)$ with behaviour based on the above discussion.

The easiest way to amend A in this way is simply to consider $\mathbf{N} \Rightarrow A$. When sa is "reached" in $\mathsf{det}(\sigma)$, play continues as below, where $n_i = \mathsf{code}_A(sab_i)$.

$$\begin{array}{ccc}
\mathbf{N} & \Rightarrow & A \\
 & \vdots & \\
 & & a \\
\mathsf{q} & & \\
\mathsf{n}_i & & \\
 & & b_i
\end{array}$$

Clearly this can be interpreted as saying that Player asks (by playing q) for advice and Opponent (by playing the code of b_i) suggests b_i which Player duly plays.

If we want to recover all possible behaviours of σ, we will have to "plug in" a suitably nondeterministic strategy for $\mathbf{1} \Rightarrow \mathbf{N}$. In this instance, it must be able to play (at the very least) each move n_i for each $n_i = \mathsf{code}_A(sab_i)$:

$$\begin{array}{ccccc}
\mathbf{1} & \Rightarrow & \mathbf{N} & \Rightarrow & A \\
 & & \vdots & & \\
 & & & & a \\
 & & \mathsf{q} & & \\
 & & \mathsf{n}_1, \mathsf{n}_2, \ldots & & \\
 & & & & b_1, b_2, \ldots
\end{array}$$

In fact, it turns out to be more convenient simply to define $\mathsf{det}(\sigma)$ in such a way that we just have to compose it with $\top_{\mathbf{N}}$—this will be our ***oracle*** strategy. In other words, Opponent can hand out bad advice, *i.e.* suggest a move not in the domain of the strategy at that point, which $\mathsf{det}(\sigma)$ will wisely ignore.

3.6.2 Deterministic factorization

Let us first of all fix, for each arena A, an injective function $\mathsf{code}_A : L_A^{\mathsf{even}} \rightarrowtail \mathbf{N}$. We now inductively define a mapping $\mathsf{det}(-)$ from L_A^{even} to $L_{\mathbf{N}\Rightarrow A}^{\mathsf{even}}$ by:

$$\begin{array}{rcl}
\mathsf{det}(\varepsilon) & = & \varepsilon \\
\mathsf{det}(sab) & = & \mathsf{det}(s) \cdot a \cdot \mathsf{q} \cdot \mathsf{code}_A(sab) \cdot b.
\end{array}$$

We need to specify the justification information too. The move q is justified by the hereditary justifier of a, $\mathsf{code}_A(sab)$ is justified by q and a and b are justified as they are in sab. (We're skimming over a few technical nasties here: strictly speaking, we should first define the underlying *string* of $\mathsf{det}(s)$ and then prove that we *can* attach justification pointers to it as stated above, thus arriving at a legal play. But it's absolutely obvious that s is a subsequence of $\mathsf{det}(s)$ (considering both as strings in $M_{\mathbf{N}\Rightarrow A}^{\star}$) so there's really nothing to prove.)

Finally, given a strategy σ for A, we define $\mathsf{det}(\sigma)$ to be:

$$\{\mathsf{det}(s) \mid s \in \sigma\} \cup \{\mathsf{det}(s) \cdot a\mathsf{q} \mid sa \in \mathsf{dom}(\sigma)\}.$$

Proposition 3.6.1 The strategy $\mathsf{det}(\sigma)$ is deterministic and $\sigma = \top_{\mathbf{N}} ; \mathsf{det}(\sigma)$.

Proof First of all, $\mathsf{det}(\sigma)$ is obviously deterministic: after an O-move in A, Player always plays q; and after an O-move in **N**, Player always plays the (unique) P-move of A associated with that number.

For the second part, note that $\top_{\mathbf{N}} ; \mathsf{det}(\sigma) = \{\mathsf{det}(s) \upharpoonright A \mid s \in \sigma\} = \sigma$. ■

This means that any strategy for A can be simulated by a deterministic strategy for $\mathbf{N} \Rightarrow A$, if we supply the "canonical" nondeterministic strategy $\top_{\mathbf{N}}$ as its "input". It's probably worth noting that a compact strategy can always be simulated by a deterministic strategy for $\mathbf{B} \Rightarrow A$ since, at any point, there can only be a finite amount of nondeterminism. In other words, "coin tossing" suffices.

3.6.3 Preservation of constraints

If we have some (possibly) nondeterministic strategy σ which satisfies, for example, P-visibility then we'd also like $\mathsf{det}(\sigma)$ to satisfy P-visibility. In other words, the deterministic factorization should *preserve* other (orthogonal) constraints. This is fairly straightforward in the cases of P-visibility and P-bracketing. First of all, we prove a useful little lemma; the desired result is then a more or less immediate consequence of this.

Lemma 3.6.2 If $sa \in L_A^{\mathsf{odd}}$ then $\ulcorner \mathsf{det}(s) \cdot a \urcorner \upharpoonright A = \ulcorner sa \urcorner$.

Proof By induction on the length of s. The case where a is initial is trivial and encompasses the base case. Otherwise, $sa = s'a'bta$ where b justifies a. In this case, $\ulcorner sa \urcorner = \ulcorner s'a' \urcorner ba = \ulcorner \mathsf{det}(s') \cdot a' \urcorner \upharpoonright A \cdot ba$, by the inductive hypothesis. Now, we have $\ulcorner \mathsf{det}(s) \cdot a \urcorner \upharpoonright A = \ulcorner \mathsf{det}(s') \cdot a'\mathsf{q} \cdot \mathsf{code}_A(s'a'b) \urcorner \cdot ba \upharpoonright A = \ulcorner \mathsf{det}(s') \cdot a' \urcorner \upharpoonright A \cdot ba$ and we're done. ■

Lemma 3.6.3 If σ satisfies P-visibility (resp. P-bracketing) then so does $\mathsf{det}(\sigma)$.

Proof For P-visibility, consider $sab \in \mathsf{det}(\sigma)$; we want the justifier of b to occur in $\ulcorner sa \urcorner$. If b is the Question in **N** then this is trivial; otherwise, b must be a move in A and must therefore be justified in A. So $sab = \mathsf{det}(s'a'b)$ for some $s'a'b \in \sigma$. So the justifier of b occurs in $\ulcorner s'a' \urcorner$ and $\ulcorner sa \urcorner = \ulcorner \mathsf{det}(s') \cdot a' \urcorner \cdot \mathsf{q}a$. Therefore we have $\ulcorner sa \urcorner \upharpoonright A = \ulcorner \mathsf{det}(s')a' \urcorner \upharpoonright A = \ulcorner s'a' \urcorner$, by the previous lemma, and we're done.

For P-bracketing, we consider $sab \in \mathsf{det}(\sigma)$ where b is an Answer. Since the only Player Answers in $\mathbf{N} \Rightarrow A$ are in A, we know that sab must be of the form $\mathsf{det}(s'a'b)$ for some $s'a'b \in \sigma$. This means that b is justified by the pending question of $\ulcorner s'a' \urcorner$. Now, since the only Opponent Questions of $\mathbf{N} \Rightarrow A$ are in A, the pending question of $\ulcorner sa \urcorner$ is the pending question of $\ulcorner sa \urcorner \upharpoonright A = \ulcorner s'a' \urcorner$ and we're done. ■

Single-threading Unfortunately, the situation is less satisfactory with single-threaded strategies: $\mathsf{det}(\sigma)$ is rarely single-threaded, even when σ is. To see why, consider the following typical play of $\mathsf{det}(\mathsf{id}_{\mathbf{C}})$.

$$\begin{array}{ccccc}
\mathbf{N} & \Rightarrow & (\mathbf{C} & \Rightarrow & \mathbf{C}) \\
 & & & & \mathsf{q} \\
\mathsf{q} & & & & \\
\mathsf{code}(\mathsf{qq}) & & & & \\
 & & \mathsf{q} & &
\end{array}$$

As defined, $\mathsf{det}(\mathsf{id}_{\mathbf{C}})$ makes no provision for the situation where Opponent, instead of playing $\mathsf{code}(\mathsf{q})$, starts an entirely new thread. In other words, the additional play in $\mathbf{N}$ provides Opponent with new opportunities to start a new thread (or, indeed, to resume an old one).

However, this is really just a consequence of the particular way in which $\mathsf{det}(-)$ has been defined and isn't a fundamental problem: if σ is a well-opened strategy then so is $\mathsf{det}(\sigma)$ and so, to factorize a single-threaded strategy, we simply factorize its well-opened representative:

$$\mathsf{Det}(\sigma) =_{\mathrm{df}} \mathsf{Traces}(\mathsf{det}(\mathsf{Rel}(\sigma))).$$

Lemmas 3.5.9 and 3.6.3 guarantee that we can shuffle between these different representations without messing up any of the other constraints.

Lemma 3.6.4 If σ is a single-threaded strategy then $\mathsf{Det}(\sigma)$ is deterministic and single-threaded and $\top_{\mathbf{N}} ; \mathsf{Det}(\sigma) = \sigma$.

Proof It's clear that $\mathsf{Det}(\sigma)$ is single-threaded and deterministic. For the second part, we calculate:

$$\begin{array}{rcl}
\mathsf{Rel}(\top_{\mathbf{N}} ; \mathsf{Det}(\sigma)) & = & \top_{\mathbf{N}} ; \mathsf{Rel}(\mathsf{Det}(\sigma)) \\
 & = & \top_{\mathbf{N}} ; \mathsf{det}(\mathsf{Rel}(\sigma)) \\
 & = & \mathsf{Rel}(\sigma).
\end{array}$$

Applying $\mathsf{Traces}(-)$ to either side yields $\top_{\mathbf{N}} ; \mathsf{Det}(\sigma) = \sigma$. ■

3.7 Innocent strategies

3.7.1 Deterministic innocence

As noted earlier, we must show that innocent strategies are closed under composition. Before doing this, we first give a little intuition for the nature of this constraint. As is commonly the case in game semantics, the best way to do this is to consider the simplest strategy that *violates* the condition of interest.

Consider the following strategy...

$$
\begin{array}{ccccc}
(\mathbf{C} & \Rightarrow & \mathbf{C}) & \Rightarrow & \mathbf{C} \\
 & & & & \mathsf{q} \\
 & & \mathsf{q} & & \\
\mathsf{q} & & & & \\
\mathsf{a} & & & & \\
 & & \mathsf{a} & & \\
 & & & & \mathsf{a}
\end{array}
$$

...which is $\sigma = \{\varepsilon, \mathsf{qq}, \mathsf{qqqa}, \mathsf{qqqaaa}\}$ when considered as a set of legal plays. This isn't innocent because $\mathsf{qqqaaa} \in \sigma$, $\mathsf{qq} \in \sigma$, $\mathsf{qqtt} \in L_{(\mathbf{C}\Rightarrow\mathbf{C})\Rightarrow\mathbf{C}}$ and $\ulcorner \mathsf{qqqttt} \urcorner = \mathsf{qqtt} = \ulcorner \mathsf{qqtt} \urcorner$, but $\mathsf{qqtttt} \notin \sigma$. (Of course, this strategy isn't single-threaded either and so couldn't possibly be innocent; but even the single-threaded strategy $\mathsf{Traces}(\sigma)$ isn't innocent, for precisely the above reason.)

A more operational explanation of this example can be given: the idea of (the process represented by) this strategy is that it takes an input f of type $\mathtt{Com} \to \mathtt{Com}$, "applies" f to `skip` and terminates if that application does. Now, it should make no difference to this whether or not f looks at its input but, as it stands, σ *is* sensitive to this. Consider "plugging in" the strategy $\{\varepsilon, \mathsf{qa}\}$ for $\mathbf{C} \Rightarrow \mathbf{C}$:

$$
\begin{array}{ccccccccccc}
\mathbf{1} & \Rightarrow & (\mathbf{C} & \Rightarrow & \mathbf{C}) & ; & (\mathbf{C} & \Rightarrow & \mathbf{C}) & \Rightarrow & \mathbf{C} \\
 & & & & & & & & & & \mathsf{q} \\
 & & & & & & & & \mathsf{q} & & \\
 & & & & \mathsf{q} & & & & & & \\
 & & & & \mathsf{a} & & & & & & \\
 & & & & & & & & \mathsf{a} & &
\end{array}
$$

It's now clear why innocence requires the play qqaa to be in σ: innocent strategies care about the answers that are produced, rather than the *means by which* they are produced. Of course, there's nothing wrong with a process that *is* sensitive to such "strictness" aspects of its inputs; but innocence is a property of strategies intended to make precise a notion of "applicative process" and such processes must be insensitive to such things.

View functions Innocent strategies can be characterized by a subclass of the well-opened strategies, known as *view functions* [63]. The details mirror those of the similar correspondence between single-threaded strategies and thread relations; we give just a brief sketch.

Given a deterministic, P-visible strategy σ, we define

$$\mathsf{fun}(\sigma) = \{\varepsilon\} \cup \{\ulcorner sab \urcorner \mid sab \in \sigma\}.$$

This is guaranteed to be a well-opened strategy.

Given a deterministic set of even-length P-views V, we define

$$\begin{aligned} T_0(V) &= \{\varepsilon\} \\ T_{n+1}(V) &= \{sab \in L_A \mid s \in T_n(V) \wedge \exists tab \in V. \ulcorner sa \urcorner \cdot b = tab\} \\ \mathsf{traces}(V) &= \bigcup_{n \in \mathbf{N}_0} T_n(V). \end{aligned}$$

This is the largest innocent strategy that can be built using V. As before, we say that V is ***saturated*** iff every entry is used, *i.e.*

$$sab \in V \Rightarrow \exists tab \in \mathsf{traces}(V). \ulcorner tab \urcorner = sab.$$

It's not too hard to rework the argument of §3.5, yielding a one-to-one correspondance between innocent strategies and saturated view functions.

Proposition 3.7.1 If σ is innocent and V is a saturated view function then

$$\begin{aligned} \mathsf{traces}(\mathsf{fun}(\sigma)) &= \sigma \\ V &= \mathsf{fun}(\mathsf{traces}(V)). \end{aligned}$$

We are now in a position to precisely identify the class of thread relations that correspond to innocent strategies. Given a saturated thread relation R, *i.e.* a well-opened strategy, we say that R is an ***innocent*** thread relation iff

- R is deterministic and P-visible;
- if $s \in \mathsf{Traces}(R)$ then: $\lceil sab \rceil \in R \iff \ulcorner sab \urcorner \in R$.

The following lemma exhibits a one-to-one correspondence between innocent thread relations and view functions.

Lemma 3.7.2 Let R be a saturated thread relation. R is innocent if, and only if, $\mathsf{Traces}(R)$ is an innocent strategy.

Proof This is immediate, by lemma 3.5.9, for determinism and P-visibility. We just need to check the liveness conditions. Firstly, suppose that R is innocent. Let $sab, t \in \mathsf{Traces}(R)$ such that $\ulcorner sa \urcorner = \ulcorner ta \urcorner$. By definition, we must have some $s'ab \in R$ such that $\lceil sab \rceil = s'ab$. Since R is innocent, we also have $\ulcorner s'ab \urcorner \in R$. So $\ulcorner ta \urcorner = \ulcorner sa \urcorner = \ulcorner \lceil sa \rceil \urcorner = \ulcorner s'a \urcorner$ from which we deduce $\ulcorner \mathsf{match}(sab, ta) \urcorner = \ulcorner s'ab \urcorner \in R$. Since $t \in \mathsf{Traces}(R)$ by hypothesis, we have $\lceil \mathsf{match}(sab, ta) \rceil \in R$ witnessing $\mathsf{match}(sab, ta) \in \mathsf{Traces}(R)$.

Conversely, if $\mathsf{Traces}(R)$ is an innocent strategy, suppose we have $s \in \mathsf{Traces}(R)$. If $\lceil sab \rceil \in R$ then $\lceil sab \rceil \in \mathsf{Traces}(R)$ and, since $\mathsf{Traces}(R)$ is innocent, $\ulcorner sab \urcorner = \ulcorner \lceil sab \rceil \urcorner \in \mathsf{Traces}(R)$ and hence $\ulcorner sab \urcorner \in R$. If $\ulcorner sab \urcorner \in R$ then $\ulcorner sab \urcorner \in \mathsf{Traces}(R)$ and, since $\mathsf{Traces}(R)$ is innocent, we have $sab \in \mathsf{Traces}(R)$ and hence $\lceil sab \rceil \in R$. ■

If V is a view function, $R_V = \mathsf{Rel}(\mathsf{traces}(V))$ is an innocent thread relation such that $\mathsf{traces}(V) = \mathsf{Traces}(R_V)$. We might reasonably ask then whether innocent thread relations are no more than those well-opened strategies that satisfy the familiar liveness condition for innocence (plus determinism and P-visibility of course), suitably modified to take account of well-openedness:

$$(sab \in \sigma \wedge t \in \sigma \wedge \ulcorner sa \urcorner) = \ulcorner ta \urcorner \Rightarrow \lceil \mathsf{match}(sab, ta) \rceil \in \sigma.$$

This is indeed the case, as the following lemma establishes.

Lemma 3.7.3 Let R be a thread relation. R is an innocent thread relation if, and only if, R is an innocent well-opened strategy.

Proof Again, we just check the liveness conditions. Firstly, suppose R is an innocent thread relation and let $sab, t \in R$ such that $\ulcorner sa \urcorner = \ulcorner ta \urcorner$. Since $s \in R$, we have $s \in \mathsf{Traces}(R)$ and so $\lceil sab \rceil \in R$ if, and only if, $\ulcorner sab \urcorner \in R$. Since $\lceil sab \rceil = sab \in R$, we have $\ulcorner sab \urcorner \in R$. As $\ulcorner \mathsf{match}(sab, ta) \urcorner = \ulcorner sab \urcorner$ and $t \in \mathsf{Traces}(R)$, we have $\lceil \mathsf{match}(sab, ta) \rceil \in R$.

Conversely, if R is an innocent well-opened strategy, suppose that $s \in \mathsf{Traces}(R)$ and $\ulcorner sab \urcorner \in R$. So $\lceil sab \rceil = \lceil s_{\leqslant b'} \rceil \cdot ab$ for some occurrence $b' \in s$. Since $s_{\leqslant b'} \in \mathsf{Traces}(R)$, we have $\lceil s_{\leqslant b'} \rceil \in R$ and $\ulcorner sa \urcorner = \ulcorner \lceil s_{\leqslant b'} \rceil \cdot a \urcorner$. But then, by innocence, $\lceil sab \rceil = \lceil s_{\leqslant b'} \rceil \cdot ab \in R$. It remains to show that, if $\lceil sab \rceil \in R$ then $\ulcorner sab \urcorner \in R$. We do this by induction on the length of s; the base case is trivial. Otherwise, $\ulcorner sab \urcorner = \ulcorner s_{\leqslant b'} \urcorner \cdot ab$ where b' justifies a. By the inductive hypothesis, since $\lceil s_{\leqslant b'} \rceil \in R$, we have $\ulcorner s_{\leqslant b'} \urcorner \in R$. So $\ulcorner \lceil sa \rceil \urcorner = \ulcorner \ulcorner s_{\leqslant b'} \urcorner \cdot a \urcorner$ and, by innocence, $\ulcorner sab \urcorner \in R$. ∎

Robustness In our current framework where strategies are just sets of traces, innocence is a little less robust than we'd like. The following lemma is the vital result—but it necessarily depends on determinism; it's worth comparing this with lemma 5.5.3 in AJM-games.

Lemma 3.7.4 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are innocent strategies, $s, t \in \sigma\,;\tau$ with (unique) witnesses $u, v \in \sigma \parallel \tau$ and $\ulcorner sa \urcorner = \ulcorner ta \urcorner$ then $\ulcorner ua \urcorner = \ulcorner va \urcorner$.

Proof By induction on $|ua|$. If a is initial—encompassing the base case—this is trivial. Otherwise, a is justified by some b' and ua has the form $u_{<a'} \cdot a' \cdot w_u \cdot b' \cdot u' \cdot a$ where a' immediately precedes b' in sa. Similarly, va has the form $v_{\leqslant a'} \cdot w_v \cdot b' \cdot v' \cdot a$ and, by the inductive hypothesis, $\ulcorner u_{\leqslant a'} \urcorner = \ulcorner v_{\leqslant a'} \urcorner$ so that $\ulcorner ua \urcorner = \ulcorner va \urcorner$ provided $w_u = w_v$. By lemma 3.4.2, $\ulcorner u_{\leqslant a'} \restriction A, B \urcorner = \ulcorner v_{\leqslant a'} \restriction A, B \urcorner$ and similarly for the B, C component. Let $w_u = m_1 \cdot w'_u$ and $w_v = m'_1 \cdot w'_v$. Assuming WLOG that a' occurs in component A, B we have, by innocence of σ, that $v_{<a'} \cdot a' m_1 \in \sigma$. But, as σ is deterministic, $v_{<a'} \cdot a' m_1 = v_{<a'} \cdot a' m'_1$ and iterating this process for as long as $|w_u|$ yields $w_u = w_v$. ∎

Proposition 3.7.5 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are innocent strategies then so is $\sigma ; \tau$.

Proof Suppose that $sab, t \in \sigma ; \tau$, $ta \in L_A$ and $\ulcorner sa \urcorner = \ulcorner ta \urcorner$. We must have witnesses $uam_1 \cdots m_k b, v \in \sigma \parallel \tau$ for sab and t. Applying the previous lemma, we get that $\ulcorner ua \urcorner = \ulcorner va \urcorner$. So, if a occurs in component X, we have $\ulcorner ua \restriction X \urcorner = \ulcorner va \restriction X \urcorner$ and invoking innocence in the appropriate strategy X yields $vam_1 \in X$. Again, we just iterate this argument to get a witness $vam_1 \cdots m_k b \in \sigma \parallel \tau$ for $tab \in \sigma ; \tau$. ■

3.7.2 Innocent nondeterminism

Unfortunately, this dependence on determinism is necessary: the argument just doesn't work in general. Why not? The essential problem arises when a given play in $\sigma ; \tau$ can be there for more than one reason, *i.e.* it has more than one witness. Consider the following strategies. Each is (the well-opened representative of) a deterministic, innocent strategy.

$$
\begin{array}{ccccc}
(\mathbf{C} & \Rightarrow & \mathbf{C}) & \overset{\sigma}{\Rightarrow} & \mathbf{C} \\
 & & & & \mathsf{q} \\
 & & \mathsf{q} & & \\
\mathsf{q} & & & & \\
\mathsf{a} & & & &
\end{array}
\qquad\qquad
\begin{array}{ccccc}
(\mathbf{C} & \Rightarrow & \mathbf{C}) & \overset{\tau}{\Rightarrow} & \mathbf{C} \\
 & & & & \mathsf{q} \\
 & & \mathsf{q} & & \\
 & & \mathsf{a} & & \\
 & & & & \mathsf{a}
\end{array}
$$

If we pair these up and compose with the obvious strategy on $(\mathbf{C} \times \mathbf{C}) \Rightarrow \mathbf{C}$ which nondeterministically copycats* between the RHS copy of $\mathbf{C}$ and either copy of $\mathbf{C}$ in the LHS, we get the following interactions.

$$
\begin{array}{ccccccccccccc}
(\mathbf{C} & \Rightarrow & \mathbf{C}) & \Rightarrow & (\mathbf{C} & \times & \mathbf{C}) & ; & (\mathbf{C} & \times & \mathbf{C}) & \Rightarrow & \mathbf{C} \\
 & & & & & & & & & & & & \mathsf{q} \\
 & & & & & & & & \mathsf{q} & & & & \\
 & & & & \mathsf{q} & & & & & & & & \\
 & & \mathsf{q} & & & & & & & & & & \\
\mathsf{q} & & & & & & & & & & & & \\
\mathsf{a} & & & & & & & & & & & & \\
\hline
 & & & & & & & & & & & & \mathsf{q} \\
 & & & & & & & & & & \mathsf{q} & & \\
 & & & & & & \mathsf{q} & & & & & & \\
 & & \mathsf{q} & & & & & & & & & & \\
 & & \mathsf{a} & & & & & & & & & & \\
 & & & & & & \mathsf{a} & & & & & & \\
 & & & & & & & & & & \mathsf{a} & & \\
 & & & & & & & & & & & & \mathsf{a}
\end{array}
$$

---

*This strategy, while not deterministic, does satisfy both P-visibility and the liveness condition for innocence.

We can see from this that the play qq has two distinct witnesses in the composite strategy. There are also two extensions, qqaa and qqqa, of qq in the composite. This is problematic because, according to the definition of innocence, we should then also be able to find qqqaaa—and yet there's patently no way this play could ever arise in this composition. This problem relates, perhaps surprisingly, to the classical question of "branching time distinctions" often found in process calculus, *e.g.* [67]: the two plays, qqaa and qqqa, should be considered as being in *separate branches* and, hence, the innocence condition shouldn't "bite" at this point.

An indirect definition of innocence We can make this intuition more precise with the following argument which is typical of the kind of "reverse engineering" often used, behind the scenes, in games research.

Suppose we have a Cartesian closed category $\mathcal{C}$; let $\mathcal{R}$ be an object of $\mathcal{C}$. We define, for any object X of $\mathcal{C}$, the "dereliction" arrow $d_X : \mathcal{R} \times X \to X$ by $d_X = \pi'_{\mathcal{R},X}$ and, for any arrow $f : \mathcal{R} \times X \to Y$, we define its "promotion" $f^{\mathcal{R}} : \mathcal{R} \times X \to \mathcal{R} \times Y$ by $f^{\mathcal{R}} = \langle \pi_{\mathcal{R},X}, f \rangle$. It's routine to check that, together with the object mapping $X \mapsto \mathcal{R} \times X$, this defines a comonad on $\mathcal{C}$. We're interested in the case where $\mathcal{C}$ is a Cartesian closed category of arenas and strategies, such as the one described earlier, and $\mathcal{R}$ is $\mathbf{N}$, the flat arena for the natural numbers.

Given an arrow $\sigma : A \to B$, we say that it's an ***innocent arrow*** iff there exists some *deterministic* innocent strategy $\mathsf{Det}(\sigma) : \mathbf{N} \times A \to B$ such that $\sigma = \mathsf{oracle}_A \,;\, \mathsf{Det}(\sigma)$, where oracle_A is defined as $\langle !_A \,;\, \top_{\mathbf{N}}, \mathsf{id}_A \rangle$. Let $\sigma : A \to B$ and $\tau : B \to C$ be innocent arrows. We calculate:

$$
\begin{aligned}
\sigma \,;\, \tau &= \mathsf{oracle}_A \,;\, \mathsf{Det}(\sigma) \,;\, \mathsf{oracle}_B \,;\, \mathsf{Det}(\tau) \\
&= \mathsf{oracle}_A \,;\, \langle !_{\mathbf{N}\times A} \,;\, \top_{\mathbf{N}}, \mathsf{Det}(\sigma) \rangle \,;\, \mathsf{Det}(\tau) \\
&= \langle \mathsf{oracle}_A \,;\, \pi_{\mathbf{N},A}, \mathsf{oracle}_A \,;\, \mathsf{Det}(\sigma) \rangle \,;\, \mathsf{Det}(\tau) \\
&= \mathsf{oracle}_A \,;\, \langle \pi_{\mathbf{N},A}, \mathsf{Det}(\sigma) \rangle \,;\, \mathsf{Det}(\tau) \\
&= \mathsf{oracle}_A \,;\, (\mathsf{Det}(\sigma) \fatsemi \mathsf{Det}(\tau)),
\end{aligned}
$$

where $\fatsemi$ denotes the Kleisli composition operation. Since we already know that deterministic innocent strategies are closed under composition, we can see that $\sigma \,;\, \tau$ is an innocent arrow. Moreover, it is just the result of supplying the oracle to the deterministic innocent strategy $\mathsf{Det}(\sigma) \fatsemi \mathsf{Det}(\tau)$ obtained by composition in the Kleisli category. It's also easy to see that this subcategory of innocent arrows is Cartesian closed.

It's worth noting that the class of innocent arrows *strictly* contains the class of (possibly) nondeterministic strategies satisfying the usual innocence condition: obviously, any deterministic innocent strategy is an innocent arrow; to see the strictness of the inclusion, let's return to the above example. One (of the many) possible ways of factorizing the composite strategy $\{\varepsilon, \mathsf{qq}, \mathsf{qqaa}, \mathsf{qqqa}\}$ as an innocent arrow is:

$$\begin{array}{ccccccc}
(\mathbf{N} & \times & (\mathbf{C} & \Rightarrow & \mathbf{C})) & \Rightarrow & \mathbf{C} \\
 & & & & & & \mathsf{q} \\
\mathsf{q} & & & & & & \\
2 & & & & & & \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
\hline
 & & & & & & \mathsf{q} \\
\mathsf{q} & & & & & & \\
3 & & & & & & \\
 & & & & \mathsf{q} & & \\
 & & & & \mathsf{a} & & \\
 & & & & & & \mathsf{a}.
\end{array}$$

The two ways in which qq can arise are distinguished by the number associated with the oracle; in this case, 2 for σ and 3 for τ. So $\ulcorner \mathsf{qq2qqaa} \urcorner \neq \ulcorner \mathsf{qq3qa} \urcorner$ and, hence, there's no need for innocence to bite. So, in some sense, those extra numbers attached to (certain) P-moves seem to encode a tree-like branching structure in the strategy.

Remark This indirect construction of a category of nondeterministic innocent strategies shows that the main result of the next section—*i.e.* full abstraction with respect to may testing for EIA—can be recast to provide a fully abstract model of PCF extended with erratic choice. However, a more "synthetic", *a priori* definition of what it is to be an innocent strategy would be preferable. The above analysis suggests that a full-blown reformulation of strategies as trees could be what is required. Rather than carrying out this programme, we'll return to the question of branching in chapter 5 in the context of AJM-games where the more detailed intensional information in the model can be exploited.

3.7.3 Innocent factorization

We now briefly sketch the innocent factorization theorem which shows that any deterministic, P-visible strategy can be written as the composite of a certain "cell" strategy with a deterministic innocent strategy. This result was first proved in [7].

Let A be an arena. We fix an encoding of the even-length legal plays of A, written $\mathsf{L}_A^{\mathsf{even}}$, as natural numbers, denoting this by $\mathsf{code}_A : \mathsf{L}_A^{\mathsf{even}} \rightarrowtail \mathbf{N}_0$. This can always be done since, by definition, A has a countable set of moves.

The idea of the factorization is to encode *history as state*, the current history of play being represented by its code which is stored in a memory cell. The factorization proceeds by first looking up the current history in the cell, deciding what to do next, updating the cell appropriately and making the chosen move.

A storage cell We must first define an appropriate arena for which our storage cell can be a strategy. Let $\mathbf{N}^\perp$ be the arena where

$$\begin{array}{rcl} M_{\mathbf{N}^\perp} & = & M_{\mathbf{N}} \\ \lambda_{\mathbf{N}^\perp}(\mathsf{q}) & = & \exists! \\ \lambda_{\mathbf{N}^\perp}(\mathsf{n}) & = & \forall? \\ \mathsf{n} & \vdash_{\mathbf{N}^\perp} & \mathsf{n}, \text{ for all } n \in \mathbf{N}_0 \\ \mathsf{n} & \vdash_{\mathbf{N}^\perp} & \mathsf{q}, \text{ for all } n \in \mathbf{N}_0 \end{array}$$

Intuitively, this is just the usual flat arena $\mathbf{N}$ turned "upside down". We think of the move n in $\mathbf{N}^\perp$ as "write the value n" and the unique answering move q as "OK", an acknowledgement.

We define the arena **Var** to be $\mathbf{N}^\perp \times \mathbf{N}$, except that we rename the moves to aid legibility so, strictly speaking, **Var** is only isomorphic to this product arena. We use $\mathsf{wr_n}$ and ok instead of n and q in $\mathbf{N}^\perp$ respectively; and rd instead of q in $\mathbf{N}$.

The strategy cell answers any move of the form $\mathsf{wr_n}$ with ok and any rd with the n such that the last "write move" was $\mathsf{wr_n}$. So, for example, we have:

$$\begin{array}{ccc} \mathbf{N}^\perp & \times & \mathbf{N} \\ \mathsf{wr_2} & & \\ \mathsf{ok} & & \\ & & \mathsf{rd} \\ & & \mathsf{2} \\ \mathsf{wr_5} & & \\ \mathsf{ok} & & \\ \mathsf{wr_3} & & \\ \mathsf{ok} & & \\ & & \mathsf{rd} \\ & & \mathsf{3}. \end{array}$$

In the event that a rd is played before any $\mathsf{wr_n}$, there is no response by cell. However, it can sometimes be useful to consider an *initialized* cell strategy; we denote by cell_n the strategy that, in this situation, answers the rd with n.

The factorization Let σ be a well-opened, deterministic, P-visible strategy for A. We define a strategy $\mathsf{inn}(\sigma)$ for $\mathbf{Var} \Rightarrow A$ with the following typical behaviour:

$$\begin{array}{ccc} \mathbf{Var} & \Rightarrow & A \\ & \vdots & \\ & & a \\ \mathsf{rd} & & \\ \mathsf{n} & & \\ \mathsf{wr_m} & & \\ \mathsf{ok} & & \\ & & b. \end{array}$$

The move n, when decoded, corresponds to some play $t \in \sigma$. If $ta \in \mathsf{dom}(\sigma)$, $\mathsf{inn}(\sigma)$ continues by writing the m such that $m = \mathsf{code}_A(tab)$ for the $tab \in \sigma$. Once this is acknowledged, $\mathsf{inn}(\sigma)$ finishes by playing the move b.

With a careful definition [7], $\mathsf{inn}(\sigma)$ is an innocent well-opened strategy. (It could equally be defined as a view function.) Moreover, if σ satisfies P-bracketing then so does $\mathsf{inn}(\sigma)$. The compactness argument for single-threaded strategies can be reworked to show that an innocent strategy is compact if, and only if, its view function is finite. If σ is finite, it's not too hard to see that $\mathsf{inn}(\sigma)$ is too, so its view function must also be finite and hence $\mathsf{inn}(\sigma)$ is compact.

It's clear that $\mathsf{cell}_e \,;\, \mathsf{inn}(\sigma) = \sigma$, where $e = \mathsf{code}_A(\varepsilon)$. This factorization can probably be generalized to avoid the assumption of determinism; but we feel that this is best left until a better understanding of nondeterministic innocence has emerged.

3.8 Full abstraction for may-equivalence

Having completed our development of HO-games, we now turn to the question of characterizing may-equivalence in EIA. We proceed by building up a model in stages, beginning with PCF.

3.8.1 The model of EIA

Typed λ-calculus We briefly review the interpretation of a typed λ-calculus in a Cartesian closed category. First of all, we must fix the semantics of the types. This requires specifying an object for each base type, in our case an arena: $[\![\mathtt{Nat}]\!] = \mathbf{N}$. We now complete the definition of the types by induction: $[\![T \to U]\!] = [\![T]\!] \Rightarrow [\![U]\!]$.

A well-typed (open) term $x_1 : T_1, \ldots, x_n : T_n \vdash M : T$ is modelled by an arrow $[\![M]\!] : [\![T_1]\!] \times \ldots \times [\![T_n]\!] \to [\![T]\!]$; if M is closed then $[\![M]\!] : \mathbf{1} \to [\![T]\!]$. We write $[\![\Gamma]\!]$ for $[\![T_1]\!] \times \ldots \times [\![T_n]\!]$ where Γ is a context $x_1 : T_1, \ldots, x_n : T_n$.

A λ-calculus term is either a variable, an abstraction or an application. Variables are modelled simply by "looking up" the appropriate part of the context. Let $\Gamma = x_1 : T_1, \ldots, x_n : T_n$ be a context. If $1 \le i \le n$ then $[\![x_i]\!]$ is the ith projection from $[\![\Gamma]\!]$.

$$[\![\Gamma \vdash x_i : T_i]\!] = \pi_i : [\![\Gamma]\!] \to [\![T_i]\!].$$

Abstractions are modelled using the *currying* natural isomorphism. If we have a term $\Gamma, x : T \vdash M : U$ then we define:

$$[\![\Gamma \vdash \lambda x.\, M : T \to U]\!] = \Lambda_T([\![\Gamma, x : T \vdash M]\!]) : [\![\Gamma]\!] \to [\![T]\!] \Rightarrow [\![U]\!].$$

The final case is that of application. This is interpreted with the *evaluation* map. If we have $\Gamma \vdash M : T \to U$ and $\Gamma \vdash N : T$ then

$$[\![\Gamma \vdash MN : U]\!] = \langle [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!] \rangle \,;\, \mathsf{eval}_{T,U} : [\![\Gamma]\!] \to [\![U]\!].$$

Arithmetic The numeral $\mathtt{n}$ is modelled by the strategy $\{\varepsilon, \mathsf{qn}, \mathsf{qnqn}, \ldots\}$. If we have a term $\Gamma \vdash M : \mathtt{Nat}$ then we define

$$\begin{array}{rcl} [\![\Gamma \vdash \mathtt{succ}\, M]\!] & = & [\![\Gamma \vdash M]\!] \,;\, \mathsf{succ} : [\![\Gamma]\!] \to \mathbf{N} \\ [\![\Gamma \vdash \mathtt{pred}\, M]\!] & = & [\![\Gamma \vdash M]\!] \,;\, \mathsf{pred} : [\![\Gamma]\!] \to \mathbf{N} \end{array}$$

where the strategies $\mathsf{succ}, \mathsf{pred} : \mathbf{N} \Rightarrow \mathbf{N}$ are specified as follows.

$$\begin{array}{ccc} & \mathsf{succ} & \\ \mathbf{N} & \Rightarrow & \mathbf{N} \\ & & \mathsf{q} \\ \mathsf{q} & & \\ \mathsf{n} & & \\ & & \mathsf{n}+1 \end{array} \qquad \begin{array}{ccc} & \mathsf{pred} & \\ \mathbf{N} & \Rightarrow & \mathbf{N} \\ & & \mathsf{q} \\ \mathsf{q} & & \\ \mathsf{m}+1 & & \\ & & \mathsf{m} \end{array} \qquad \begin{array}{ccc} & \mathsf{pred} & \\ \mathbf{N} & \Rightarrow & \mathbf{N} \\ & & \mathsf{q} \\ \mathsf{q} & & \\ 0 & & \\ & & 0 \end{array}$$

Conditional The conditional is modelled in a similar fashion to $\mathtt{succ}$ and $\mathtt{pred}$. Given well-typed terms $\Gamma \vdash M : \mathtt{Nat}$, $\Gamma \vdash N : \mathtt{Nat}$ and $\Gamma \vdash L : \mathtt{Nat}$, we set

$$[\![\mathtt{if0}\ M\ N\ L]\!] = \langle [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!], [\![\Gamma \vdash L]\!] \rangle \,;\, \mathsf{if0} : [\![\Gamma]\!] \to \mathbf{N}$$

where $\mathsf{if0} : (\mathbf{N} \times \mathbf{N} \times \mathbf{N}) \Rightarrow \mathbf{N}$ is specified as follows.

$$\begin{array}{ccccccc} & & & & & \mathsf{if0} & \\ (\mathbf{N} & \times & \mathbf{N} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\ & & & & & & \mathsf{q} \\ \mathsf{q} & & & & & & \\ 0 & & & & & & \\ & & \mathsf{q} & & & & \\ & & \mathsf{n} & & & & \\ & & & & & & \mathsf{n} \end{array} \qquad \begin{array}{ccccccc} & & & & & \mathsf{if0} & \\ (\mathbf{N} & \times & \mathbf{N} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\ & & & & & & \mathsf{q} \\ \mathsf{q} & & & & & & \\ \mathsf{n}+1 & & & & & & \\ & & & & \mathsf{q} & & \\ & & & & \mathsf{n} & & \\ & & & & & & \mathsf{n} \end{array}$$

Recursion A well-typed term $\Gamma \vdash M : T \to T$ is modelled by a strategy $[\![M]\!] : [\![\Gamma]\!] \Rightarrow ([\![T]\!] \Rightarrow [\![T]\!])$. We use this to build a chain of strategies on $[\![\Gamma]\!] \Rightarrow [\![T]\!]$.

$$\begin{array}{rcl} f_0 & = & \bot_{[\![\Gamma]\!] \Rightarrow [\![T]\!]} \\ f_{n+1} & = & \langle \mathsf{id}_{[\![\Gamma]\!]}, f_n \rangle \,;\, \Lambda^{-1}([\![M]\!]) \end{array}$$

In other words, we "apply" $[\![M]\!]$ to $\bot$ and then "apply" $[\![M]\!]$ to the result of that and then "apply" $[\![M]\!]$ to the result of that... in the limit, setting:

$$[\![\mathbf{Y}_T\, M]\!] = \bigsqcup_{i \in \mathbf{N}} f_i.$$

Variables We've now covered the game semantics of all the constructs of PCF. To extend this to a model of IA, we must firstly account for the extra base types. The type of commands is easy enough: $[\![\texttt{Com}]\!] = \mathbf{C}$; we model `skip` as $\{\varepsilon, \mathsf{qa}, \mathsf{qaqa}, \ldots\}$, *i.e.* the unique "numeral" of type `Com`. The type of variables is modelled by $\mathbf{N}^{\perp} \times \mathbf{N}$, as described in the earlier section on the innocent factorization.

To model `assign` and `deref`, we proceed analogously to the arithmetic terms of PCF. Given terms $\Gamma \vdash M : \texttt{Var}$ and $\Gamma \vdash N : \texttt{Nat}$, we define:

$$\begin{array}{rcl} [\![\texttt{assign}\, M\, N]\!] & = & \langle [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!] \rangle \,;\, \mathsf{assign} \\ [\![\texttt{deref}\, M]\!] & = & [\![\Gamma \vdash M]\!] \,;\, \mathsf{deref} \end{array}$$

where assign and deref are specified below.

$$\begin{array}{ccccccc} (\mathbf{N}^{\perp} & \times & \mathbf{N}) & \times & \mathbf{N} & \Rightarrow & \mathbf{C} \\ & & & & & & \mathsf{q} \\ & & & & \mathsf{rd} & & \\ & & & & \mathsf{n} & & \\ \mathsf{wr_n} & & & & & & \\ \mathsf{ok} & & & & & & \\ & & & & & & \mathsf{a} \end{array} \qquad \begin{array}{ccccc} (\mathbf{N}^{\perp} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\ & & & & \mathsf{q} \\ & & \mathsf{rd} & & \\ & & \mathsf{n} & & \\ & & & & \mathsf{n} \end{array}$$

Sequencing The sequencing terms both begin by evaluating the first argument. $\texttt{Seq}_{\texttt{Nat}}$ then returns the result of evaluating its second argument; $\texttt{seq}_{\texttt{Com}}$ simply runs its second command argument. In effect, sequencing is a unary case statement; since there are no distinguishable numerals of type `Com`, the "conditional branching" just boils down to the question of convergence.

Given terms $\Gamma \vdash M : \texttt{Com}$, $\Gamma \vdash N : \texttt{Nat}$ and $\Gamma \vdash L : \texttt{Com}$, we define:

$$\begin{array}{rcl} [\![\texttt{seq}_{\texttt{Nat}}\, M\, N]\!] & = & \langle [\![M]\!], [\![N]\!] \rangle \,;\, \mathsf{seqN} : [\![\Gamma]\!] \to \mathbf{N} \\ [\![\texttt{seq}_{\texttt{Com}}\, M\, L]\!] & = & \langle [\![M]\!], [\![L]\!] \rangle \,;\, \mathsf{seqC} : [\![\Gamma]\!] \to \mathbf{C} \end{array}$$

where seqC and seqN are specified as follows.

$$\begin{array}{ccccc} & & & \mathsf{seqC} & \\ (\mathbf{C} & \times & \mathbf{C}) & \Rightarrow & \mathbf{C} \\ & & & & \mathsf{q} \\ \mathsf{q} & & & & \\ \mathsf{a} & & & & \\ & & \mathsf{q} & & \\ & & \mathsf{a} & & \\ & & & & \mathsf{a} \end{array} \qquad \begin{array}{ccccc} & & & \mathsf{seqN} & \\ (\mathbf{C} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\ & & & & \mathsf{q} \\ \mathsf{q} & & & & \\ \mathsf{a} & & & & \\ & & \mathsf{q} & & \\ & & \mathsf{n} & & \\ & & & & \mathsf{n} \end{array}$$

Mkvar The `mkvar` constructor essentially corresponds to pairing in the category. Given terms $\Gamma \vdash M : \texttt{Nat} \to \texttt{Com}$ and $\Gamma \vdash N : \texttt{Nat}$, we define:

$$[\![\texttt{mkvar}\ M\ N : \texttt{Var}]\!] = \langle [\![M]\!]\ ;\ \mathsf{accept}, [\![N]\!] \rangle$$

where the (innocent) strategy accept transforms a strategy for $\mathbf{N} \Rightarrow \mathbf{C}$ to one for $\mathbf{N}^{\perp}$.

$$\begin{array}{ccccc}
(\mathbf{N} & \Rightarrow & \mathbf{C}) & \Rightarrow & \mathbf{N}^{\perp} \\
 & & & & \mathsf{wr}_n \\
 & & \mathsf{q} & & \\
\mathsf{q} & & & & \\
\mathsf{n} & & & & \\
 & & \mathsf{a} & & \\
 & & & & \mathsf{ok}
\end{array}$$

New The most important clause of the semantics of IA is undoubtedly that of `new`. This is the only point where history-sensitive behaviour can be introduced into the model; all strategies used up until now have preserved innocence.

Suppose we have a term $\Gamma \vdash M : \texttt{Var} \to T$ where T is either `Nat` or `Com`. The essential idea of `new` is that a new "storage cell" is locally bound to M's parameter. We first define $\mathsf{Cell}_\Gamma : [\![\Gamma]\!] \to \mathbf{Var}$ by $!_{[\![\Gamma]\!]}\ ;\ \mathsf{cell}$ and then:

$$[\![\texttt{new}_T\ M]\!] = \mathsf{Traces}(\langle \mathsf{id}_{[\![\Gamma]\!]}, \mathsf{Cell}_\Gamma \rangle\ ;\ \mathsf{Rel}(\Lambda^{-1}_{\mathbf{Var}}([\![M]\!]))).$$

To illustrate this definition (which is exactly that of [7]), recall the example, from chapter 2, that highlights the difference between active and passive expressions in IA. We start with a term F defined as:

$$\lambda n.\,\texttt{if}\ n = 2\ \texttt{then}\ (\texttt{if}\ n = 3\ \texttt{then}\ 5\ \texttt{else}\ \Omega\ \texttt{fi})\ \texttt{else}\ \Omega\ \texttt{fi} : \texttt{Nat} \to \texttt{Nat}$$

and build a new term:

$$\texttt{x} : \texttt{Var} \vdash \texttt{x} := 1;\ F(\texttt{x} := !\texttt{x} + 1;\ !\texttt{x}) : \texttt{Nat}.$$

We interpret F and the fragment '$\texttt{x} := !\texttt{x} + 1;\ !\texttt{x}$' with the following strategies.

$$\begin{array}{ccccccc}
\mathbf{N} & \Rightarrow & \mathbf{N} & \qquad & \mathbf{Var} & \Rightarrow & \mathbf{N} \\
 & & \mathsf{q} & & & & \mathsf{q} \\
\mathsf{q} & & & & \mathsf{rd} & & \\
2 & & & & \mathsf{n} & & \\
\mathsf{q} & & & & \mathsf{wr}_{n+1} & & \\
3 & & & & \mathsf{ok} & & \\
 & & 5 & & \mathsf{rd} & & \\
 & & & & \mathsf{m} & & \\
 & & & & & & \mathsf{m}
\end{array}$$

We can now construct the strategy for '$\mathtt{x} : \mathtt{Var} \vdash \mathtt{x} := 1;\ \mathsf{F}(\mathtt{x} := !\mathtt{x} + 1;\ !\mathtt{x}) : \mathtt{Nat}$' essentially by composing these strategies, but with the extra "initializing write" at the beginning.

$$
\begin{array}{ccccc}
\mathbf{Var} & \Rightarrow & \mathbf{N} & \Rightarrow & \mathbf{N} \\
 & & & & \mathsf{q} \\
\mathsf{wr}_1 & & & & \\
\mathsf{ok} & & & & \\
 & & \mathsf{q} & & \\
\mathsf{rd} & & & & \\
\mathsf{n} & & & & \\
\mathsf{wr}_{\mathsf{n}+1} & & & & \\
\mathsf{ok} & & & & \\
\mathsf{rd} & & & & \\
\mathsf{2} & & & & \\
 & & \mathsf{2} & & \\
 & & \mathsf{q} & & \\
\mathsf{rd} & & & & \\
\mathsf{m} & & & & \\
\mathsf{wr}_{\mathsf{m}+1} & & & & \\
\mathsf{ok} & & & & \\
\mathsf{rd} & & & & \\
\mathsf{3} & & & & \\
 & & \mathsf{3} & & \\
 & & & & \mathsf{5}
\end{array}
$$

It's pretty clear that composing the above composite with cell will produce (the well-opened representative of) $[\![5]\!]$, as expected.

Erratic choice Given terms $\Gamma \vdash M : \mathtt{Nat}$ and $\Gamma \vdash N : \mathtt{Nat}$, we define:

$$[\![M \ \mathtt{or}\ N]\!] = \langle [\![M]\!], [\![N]\!] \rangle \ ; \mathsf{choice}$$

where the two typical plays of choice are given below. Just as the strategy cell is the only source of history-sensitive behaviour in the model, the strategy choice is its only source of nondeterminism.

$$
\begin{array}{ccccc}
(\mathbf{N} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\
 & & & & \mathsf{q} \\
\mathsf{q} & & & & \\
\mathsf{n} & & & & \\
 & & & & \mathsf{n} \\
\hline
 & & & & \mathsf{q} \\
 & & \mathsf{q} & & \\
 & & \mathsf{m} & & \\
 & & & & \mathsf{m}
\end{array}
$$

3.8.2 Soundness

Our aim in this section is to show the *soundness* of this model with respect to may contextual approximation, *i.e.* if $[\![M]\!] \leq [\![N]\!]$ then $M \lesssim_{\mathrm{may}} N$.

First of all, we have the usual kind of *substitution lemma*.

Lemma 3.8.1 (substitution) If $\Gamma, x : T \vdash M : U$ and $\Gamma \vdash N : T$ are well-typed terms then so is $\Gamma \vdash M[N/x] : U$ and $[\![M[N/x]]\!] = \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, [\![M]\!]$.

Proof A lengthy but unspectacular induction on the structure of M. The most exciting case is probably where M is of the form $\mathbf{Y}(M')$. In this case, $M[N/x] = \mathbf{Y}(M'[N/x])$ and, by the inductive hypothesis, $[\![M'[N/x]]\!] = \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, [\![M']\!]$.

We build an ω-chain in $[\![\Gamma]\!] \to [\![U]\!]$:

$$
\begin{aligned}
f_0 &= \perp_{\Gamma,U} \\
f_{n+1} &= \langle \mathsf{id}_{[\![\Gamma]\!]}, f_n\rangle \,;\, \Lambda^{-1}[\![M'[N/x]]\!],
\end{aligned}
$$

setting $[\![\mathbf{Y}(M'[N/x])]\!] = \bigsqcup_{i\in\mathbf{N}} f_i$. We interpret $[\![M]\!]$ as the lub of an analogous ω-chain $\{f'_i\}$ in $[\![\Gamma]\!] \times [\![T]\!] \to [\![U]\!]$.

We claim that $f_i = \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, f'_i$, for all $i \in \mathbf{N}$. This is proved by induction on i. The base case is easy. For the inductive step, we exploit naturality of currying to calculate:

$$
\begin{aligned}
f_{i+1} &= \langle \mathsf{id}_{[\![\Gamma]\!]}, f_i\rangle \,;\, \Lambda^{-1}(\langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, [\![M']\!]) \\
&= \langle \mathsf{id}_{[\![\Gamma]\!]}, f_i\rangle \,;\, (\langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \times \mathsf{id}_{[\![U]\!]}) \,;\, \Lambda^{-1}[\![M']\!] \\
&= \langle\langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle, f_n\rangle \,;\, \Lambda^{-1}[\![M']\!] \\
&= \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, \langle \mathsf{id}_{[\![\Gamma]\!]\times[\![T]\!]}, f'_i\rangle \,;\, \Lambda^{-1}[\![M']\!] \\
&= \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, f'_{i+1}.
\end{aligned}
$$

Finally, we use continuity of composition to "reshuffle" our two ω-chains:

$$
\begin{aligned}
[\![M[N/x]]\!] &= \bigsqcup_{i\in\mathbf{N}} f_i \\
&= \bigsqcup_{i\in\mathbf{N}} \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, f'_i \\
&= \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, \bigsqcup_{i\in\mathbf{N}} f'_i \\
&= \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, [\![M]\!],
\end{aligned}
$$

and we're done. ■

Consistency The next stage in the development of our model is the *consistency* result showing that any EIA program that may converge has an appropriate trace in its semantics reflecting this. The first step is just a semantic analogue to the syntactic results concerning higher types in IA.

Lemma 3.8.2 If $\Gamma \vdash M : T$ where T is either of the form $T_1 \to T_2$ or $\mathtt{Var}$ and $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$ then ($s = s'$ and) $[\![M]\!] = [\![V]\!]$.

Proof A simple induction on the derivation that $\langle s, M\rangle \Downarrow \langle s', V\rangle$, using the substitution lemma and continuity of composition. ■

Let $\Gamma = v_1 : \mathtt{Var}, \ldots, v_n : \mathtt{Var}$ be a $\mathtt{Var}$-context and s be a Γ-store. We can represent this as a strategy $[\![s]\!]$ for $\mathbf{1} \Rightarrow (\mathbf{Var} \times \cdots \times \mathbf{Var})$ where $[\![s]\!] = \langle \mathsf{cell}_{s(v_1)}, \ldots, \mathsf{cell}_{s(v_n)}\rangle$. If $s(v_i)$ is undefined, we just use the uninitialized cell strategy. If t is a play in $\mathbf{Var} \times \cdots \times \mathbf{Var}$ behaving like a tuple of cell strategies where each cell is initialized as in s, we say that t *leaves* state s' iff for each v_i in Γ, the last write move of t in the ith copy of $\mathtt{Var}$ sets v_i to $s'(v_i)$. If there is no write move in the ith component, we must have $s'(v_i) = s(v_i)$ if $s(v_i)\downarrow$; otherwise $s'(v_i)$ is undefined.

For example, if we have three cells v_1, v_2 and v_3, initialized as $s(v_1) = 2$, $s(v_2) = 3$ and $s(v_3)$ undefined, the following play leaves state s' given by $v_1 \mapsto 2$, $v_2 \mapsto 3$ and $v_3 \mapsto 5$.

$$
\begin{array}{ccccc}
\mathbf{Var} & \times & \mathbf{Var} & \times & \mathbf{Var} \\
 & & \mathsf{wr}_5 & & \\
 & & \mathsf{ok} & & \\
\mathsf{rd} & & & & \\
2 & & & & \\
 & & & & \mathsf{wr}_5 \\
 & & & & \mathsf{ok} \\
 & & \mathsf{wr}_3 & & \\
 & & \mathsf{ok} & &
\end{array}
$$

Lemma 3.8.3 If $\Gamma \vdash M : T$ where T is $\mathtt{Nat}$ or $\mathtt{Com}$ and $\langle s, M\rangle \Downarrow_\Gamma \langle s', V\rangle$ then for each $it \in \mathsf{Rel}([\![V]\!])$ (where i is an initial move) we have some $it't \in \mathsf{Rel}([\![M]\!])$ such that t' plays solely in $[\![\Gamma]\!]$ and leaves state s'. Moreover, every trace in $\mathsf{Rel}([\![M]\!])$ is of this form.

Proof A straightforward induction on the derivation that $\langle s, M\rangle \Downarrow \langle s', V\rangle$. We consider the two most interesting cases, those of assignment to and derefencing from a variable. In the latter case, we have:

$$\frac{\langle s, M\rangle \Downarrow \langle s', v\rangle \quad s'(v) = n}{\langle s, \mathtt{deref}\ M\rangle \Downarrow \langle s', \mathtt{n}\rangle}$$

By the inductive hypothesis, $[\![M]\!]$ is just the appropriate projection from $[\![\Gamma]\!]$. By definition, $[\![\mathtt{deref}\ M]\!]$ is simply the second projection from $\mathbf{Var}$ so every trace in $\mathsf{Rel}([\![\mathtt{deref}\ M]\!])$ has the form $\mathsf{q} \cdot \mathsf{rd} \cdot \mathsf{s(v)} \cdot \mathsf{s(v)}$, leaving state s $(= s')$ as required.

The case of assignment is very similar to this. The rule is:

$$\frac{\langle s, N\rangle \Downarrow_\Gamma \langle s', \mathtt{n}\rangle \quad \langle s', M\rangle \Downarrow_\Gamma \langle s'', v\rangle}{\langle s, \mathtt{assign}\, M\ N\rangle \Downarrow_\Gamma \langle\langle s'' \mid v \mapsto n\rangle, \mathtt{skip}\rangle}$$

By the inductive hypothesis and the definition of the semantics, $s' = s''$ and we must have an interaction of the following form, where t' leaves state s'.

$$\begin{array}{ccccccc}
\Gamma & \Rightarrow & (\mathbf{Var} & \times & \mathbf{N}) & \Rightarrow & \mathbf{C} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
t' & & & & & & \\
 & & & & \mathsf{n} & & \\
 & & \mathsf{wr_n} & & & & \\
\mathsf{wr_n} & & & & & & \\
\mathsf{ok} & & & & & & \\
 & & \mathsf{ok} & & & & \\
 & & & & & & \mathsf{a}
\end{array}$$

It's then obvious that the whole play leaves state $\langle s' \mid v \mapsto n\rangle$ as desired.

Finally, we consider the case of erratic choice. The left rule is:

$$\frac{\langle s, M\rangle \Downarrow \langle s', \mathtt{n}\rangle}{\langle s, M \ \mathtt{or}\ N\rangle \Downarrow \langle s', \mathtt{n}\rangle}$$

By the inductive hypothesis, we have $\mathsf{q} \cdot t' \cdot \mathsf{n} \in \mathsf{Rel}(\llbracket M\rrbracket)$ where t' leaves state s' and hence, by definition, $\mathsf{q} \cdot t' \cdot \mathsf{n} \in \mathsf{Rel}(\llbracket M \ \mathtt{or}\ N\rrbracket)$ too. The case for the right rule is analogous. Clearly, any trace in $\mathsf{Rel}(\llbracket M \ \mathtt{or}\ N\rrbracket)$ arises from one or the other. ■

Corollary 3.8.4 (consistency) If M is a program, *i.e.* closed term of type $\mathtt{Com}$, and $M \Downarrow^{\mathrm{may}}$ then $\mathsf{qa} \in \llbracket M\rrbracket$.

Computational adequacy We now strengthen consistency by proving that its converse is also true, *i.e.* if a program's semantics contains an appropriate trace then this means that the program may converge. This kind of result is usually known as *computational adequacy* and, together with the consistency result from above, it implies the (in)equational soundness theorem.

Let $\Gamma, \Delta \vdash M : T$ be a term of EIA where Γ is a $\mathtt{Var}$-context. We say that $\Gamma \mid \Delta \vdash M : T$ is a ***split term***. If Δ is empty, M is ***semi-closed***. The semi-closed split terms correspond to those EIA-terms for which the operational semantics is defined: they have no "unbound identifiers", just some "global variables".

We now define a predicate of computability on split terms which, for programs, specializes to the desired converse to consistency.

Definition

- $\Gamma \mid - \vdash M : \mathtt{Com}$ is computable iff, for any Γ-store s, if $\mathsf{qta} \in [\![s]\!] \parallel \mathsf{Rel}([\![M]\!])$ where t leaves s' then $\langle s, M\rangle \Downarrow \langle s', \mathtt{skip}\rangle$.
- $\Gamma \mid - \vdash M : \mathtt{Nat}$ is computable iff, for any Γ-store s, if $\mathsf{qtn} \in [\![s]\!] \parallel \mathsf{Rel}([\![M]\!])$ where t leaves s' then $\langle s, M\rangle \Downarrow \langle s', \mathtt{n}\rangle$.
- if $\Gamma \mid - \vdash M : \mathtt{Var}$ then M is computable iff
  - $\Gamma \mid - \vdash \mathtt{deref}\ M : \mathtt{Nat}$ is computable;
  - for all $n \in \mathbf{N}_0$, $\Gamma \mid - \vdash \mathtt{assign}\, M\, \mathtt{n}$ is computable.
- if $\Gamma \mid - \vdash M : T \to U$ then M is computable iff $\Gamma \mid - \vdash MN : U$ is computable whenever $\Gamma \mid - \vdash N : T$ is.
- if $\Gamma \mid \Delta \vdash M : T$ where $\Delta = x_1 : T_1, \ldots, x_k : T_k$ then M is computable iff $\Gamma \mid - \vdash M[N_1/x_1, \ldots, N_k/x_k] : T$ is computable whenever $\Gamma \mid - \vdash N_i : T_i$ are, for $1 \leq i \leq k$.

Proposition 3.8.5 All split terms are computable.

Proof First of all, we prove that all split terms with no subterms of the form '$\mathbf{Y}(M)$' other than Ω are computable. This is done by induction on the structure of M.

If M is a variable, the result is trivial unless M is $v : \mathtt{Var}$ coming from the $\mathtt{Var}$-context Γ. In this case, we have to check that both '$\mathtt{deref}\ v$' and '$\mathtt{assign}\ v\ \mathtt{n}$' are computable. The latter is easy; for derefencing, suppose that $\mathsf{qn} \in [\![s]\!] \mathbin{;} [\![\mathtt{deref}\ v]\!]$ so we must have an interaction of the following form:

$$
\begin{array}{ccccccccc}
\mathbf{1} & \Rightarrow & (\Gamma & \times & \Delta) & \Rightarrow & \mathbf{Var} & \Rightarrow & \mathbf{N} \\
 & & & & & & & & \mathsf{q} \\
 & & & & & & \mathsf{rd} & & \\
 & & \mathsf{rd} & & & & & & \\
 & & \mathsf{n} & & & & & & \\
 & & & & & & \mathsf{n} & & \\
 & & & & & & & & \mathsf{n}
\end{array}
$$

But then $s(v) = n$ and we can apply the rule:

$$
\frac{\langle s, v\rangle \Downarrow \langle s, v\rangle}{\langle s, \mathtt{deref}\ v\rangle \Downarrow \langle s, \mathtt{n}\rangle}\ s(v) = n
$$

showing that '$\mathtt{deref}\ v$' is computable.

The cases of application and abstraction follow in standard fashion. We illustrate the remaining cases with a few examples.

If $\Gamma \mid \Delta \vdash M : \texttt{Nat}$ is of the form '$M_1 \texttt{ or } M_2$', let M_1' and M_2' be M_1 and M_2 with the free variables from Δ instantiated by semi-closed, computable terms. By the inductive hypothesis, M_1 and M_2 are both computable so, by definition, so are M_1' and M_2'. If $\mathsf{q}t\mathsf{n} \in [\![s]\!] \parallel [\![M_1' \texttt{ or } M_2']\!]$ then $\mathsf{q}t\mathsf{n} \in [\![s]\!] \parallel [\![M_1']\!]$ or similarly for M_2'. By definition, we therefore have $\langle s, M_1' \rangle \Downarrow \langle s', \texttt{n} \rangle$ and hence $\langle s, M_1' \texttt{ or } M_2' \rangle \Downarrow \langle s', \texttt{n} \rangle$ or similarly for M_2'. So '$M_1' \texttt{or} M_2'$' is computable and, by definition, M is computable.

If $\Gamma \mid \Delta \vdash M : \texttt{Com}$ is of the form '$\texttt{seq } M_1\ M_2$', as before we let M_1' and M_2' be M_1 and M_2 with all free variables from Δ instantiated by semi-closed, computable terms. As before, M_1' and M_2' are computable. If $\mathsf{q}t\mathsf{a} \in [\![s]\!] \parallel [\![\texttt{seq } M_1'\ M_2']\!]$ then $t = t_1 t_2$ where $\mathsf{q}t_1\mathsf{a} \in [\![s]\!] \parallel [\![M_1']\!]$:

$$
\begin{array}{ccccccccc}
\mathbf{1} & \Rightarrow & \Gamma & \Rightarrow & (\mathbf{C} & \times & \mathbf{C}) & \Rightarrow & \mathbf{C} \\
 & & & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & & & \\
 & & t_1 & & & & & & \\
 & & & & \mathsf{a} & & & & \\
 & & & & & & \mathsf{q} & & \\
 & & t_2 & & & & & & \\
 & & & & & & \mathsf{a} & & \\
 & & & & & & & & \mathsf{a}
\end{array}
$$

By the inductive hypothesis, $\langle s, M_1' \rangle \Downarrow \langle s', \texttt{skip} \rangle$ for the s' left by t_1. Similarly, we must have $\mathsf{q}t_2\mathsf{a} \in [\![s']\!] \parallel [\![M_2']\!]$ and so $\langle s', M_2' \rangle \Downarrow \langle s'', \texttt{skip} \rangle$ for the s'' left by t_2. So $\langle s, \texttt{seq } M_1'\ M_2' \rangle \Downarrow \langle s'', \texttt{skip} \rangle$, hence '$\texttt{seq } M_1'\ M_2'$' and so M are computable.

Finally, we lift this result to the full language. WLOG, let M be a semi-closed term of type $\texttt{Com}$; the case where M has type $\texttt{Nat}$ is similar. For every $k \in \mathbf{N}_0$, we define M_k to be the term resulting from substituting every occurrence of a subterm of the form $\mathbf{Y}(N)$ with $\mathbf{Y}^k(N)$. In other words, M_k allows recursive unwinding "up to depth k". If $\mathsf{q}t\mathsf{a} \in [\![s]\!] \parallel [\![M]\!]$ then, for some $\ell \in \mathbf{N}_0$, we have $\mathsf{q}t\mathsf{a} \in [\![s]\!] \parallel [\![M_\ell]\!]$. We already know that M_ℓ is computable so $\langle s, M_\ell \rangle \Downarrow \langle s', \texttt{skip} \rangle$ for the s' left by t. But, since $M_\ell \prec M$, we apply lemma 2.3.3 to get $\langle s, M \rangle \Downarrow \langle s', \texttt{skip} \rangle$ and so M is computable. ■

Corollary 3.8.6 (adequacy) If M is a program of EIA and $\mathsf{qa} \in [\![M]\!]$ then $M \Downarrow^{\mathrm{may}}$.

We can put all these results together in the usual way, obtaining an (inequational) *soundness* result for may testing.

Theorem 3.8.7 (soundness) If M and N are closed terms of EIA such that $[\![M]\!] \leq [\![N]\!]$ then $M \lesssim_{\mathrm{may}} N$.

Proof Let $C[-]$ be any program context of EIA and suppose that $C[M] \Downarrow^{\mathrm{may}}$ so that $\mathsf{qa} \in [\![C[M]]\!]$. Since $[\![M]\!] \leq [\![N]\!]$, we have that $[\![C[M]]\!] \leq [\![C[N]]\!]$ and so $\mathsf{qa} \in [\![C[N]]\!]$ and hence $C[N] \Downarrow^{\mathrm{may}}$. ■

3.8.3 Definability

The interpretation of the types of PCF, or of IA, inductively defines a collection of arenas that we'll refer to as the *domaine* of the model. The soundness result proved above tells us that the model "tells no lies" about operational behaviour: if the model says that two terms are "the same" then they are indeed operationally indistinguishable.

In this section, we show a sort of converse to this: that every *compact* strategy for an arena in the domaine of the model is definable in EIA. The proof rests on three things: the two factorizations, taking us from nondeterministic to innocent strategies; and the *innocent definability* result, showing that any innocently compact strategy on an arena in the domaine is definable in IA – `new` [7].

In order to prove this, we actually need to extend the language a little; we add, for every $k \in \mathbf{N}$, a `case` statement

$$\mathtt{case}_{k,B} : \mathtt{Nat} \to \underbrace{(B \to \cdots \to B)}_{k} \to B$$

where B is either `Nat` or `Com`. This is a "harmless" extension in the sense that any term '$\mathtt{case}_k\ M\ M_1 \cdots M_k$' is definable by a term of IA, *i.e.*

$$\mathtt{new}(\lambda v.\ \mathtt{seq}\ (\mathtt{assign}\ v\ M)\ (\mathtt{if0}\ \ (\mathtt{deref}\ v)\ \ M_1\ \ (\mathtt{if0}\ \ (\mathtt{pred}\ (\mathtt{deref}\ v))\ \ M_2\ \ \cdots))).$$

This extended language is called IA'.

Theorem 3.8.8 Suppose that $T_1, \ldots, T_n$ are IA types interpreted respectively by arenas $A_1, \ldots, A_n$ and let $\sigma : (A_1 \times \cdots \times A_i) \Rightarrow ((A_{i+1} \Rightarrow \cdots \Rightarrow A_n) \Rightarrow B)$ be an innocent, well-bracketed strategy with finite view function where B is either $\mathbf{N}$, $\mathbf{C}$ or $\mathbf{Var}$. Then there exists an IA' term $x_1 : T_1, \ldots, x_i : T_i \vdash M : T_{i+1} \to \cdots \to T_n \to T$ such that $\sigma = [\![M]\!]$.

Proof We first consider the case where B is $\mathbf{N}$ or $\mathbf{C}$; the case where B is $\mathbf{Var}$ can be reduced to these. The proof is by induction on the size of σ's view function. We go by a case analysis on the response of σ to the initial Question in B.

- If there's no response from σ, M is just Ω; this is our base case.
- If B is `Nat` and σ answers q with an Answer k then M is a constant term:

  $$x_1 : T_1, \ldots, x_i : T_i \vdash \lambda x_{i+1}. \ldots \lambda x_n. \mathtt{k}.$$

  If B is `Com`, we simply replace the numeral `k` with `skip`.

The interesting case is where σ responds by playing a Question q_j in some A_j. In this case, we uncurry σ, arriving at $\sigma' : (A_1 \times \cdots \times A_n) \Rightarrow B$. We separate out the thread of play in A_j induced by q_j, yielding $\sigma'' : (A_1 \times \cdots \times A_n \times A_j) \Rightarrow B$.

Consider $\ulcorner sab \urcorner$ for some $sab \in \sigma''$. If this takes the form $qq_j ks'$, the subsequence qs' is the P-view of a play in $(A_1 \times \cdots \times A_n) \Rightarrow B$. So, for each $k \in \mathbf{N}_0$, the set of P-views $\{qs' \mid \exists sab \in \sigma''. \ulcorner sab \urcorner = qq_j ks'\}$ determines an innocent strategy σ_k. The view function of σ_k must be strictly smaller than that of σ so, by the inductive hypothesis, we have a term M_k defining σ_k. Let ℓ be such that, for all $\ell' \geq \ell$, $\sigma_{\ell'} = \{\varepsilon\}$. If $\ulcorner sab \urcorner$ is of the form $qq_j as'$, the same argument applies; in this case, we have just a single $\sigma_\bullet$ defined by $M_\bullet$.

The other possibility is that $\ulcorner sab \urcorner = qq_j s'$ contains no Answer to q_j. Since σ is well-bracketed, s' is restricted to playing in the A_is and the extra copy of A_j. Suppose that A_j is of the form $A'_1 \Rightarrow \cdots \Rightarrow A'_l \Rightarrow B_j$. By well-bracketing again, the play in the extra A_j is restricted to the A'_is. We can therefore consider s' as a play in $(A_1 \times \cdots \times A_n) \Rightarrow (A'_1 \times \cdots \times A'_l)$. The set of all P-views of this form induces an innocent strategy for this arena and can therefore be written as $\langle \sigma'_1, \ldots, \sigma'_l \rangle$. Again, the view function of each σ'_i is strictly smaller than that of σ and is therefore definable by some term M'_i.

Having "pulled σ apart" in this way, we can reconstitute it as a term of IA'. In each case, the idea is that σ is some kind of case statement.

- If B_j is $\mathbf{N}$, we have: $\sigma' = [\![\mathtt{case}_{\ell,B}\ (x_j M'_1 \cdots M'_l)\ M_0 \cdots M_{\ell-1}]\!]$.
- If B_j is $\mathbf{C}$, we have: $\sigma' = [\![\mathtt{seq}_B\ (x_j M'_1 \cdots M'_l)\ M_\bullet]\!]$.
- If B_j is $\mathbf{Var} = \mathbf{N} \times \mathbf{N}^\perp$, there are two possibilities.
  - If q_j is in $\mathbf{N}$, we have: $\sigma' = [\![\mathtt{case}_{\ell,B}\ (\mathtt{deref}\ (x_j M'_1 \cdots M'_l))\ M_0 \cdots M_{\ell-1}]\!]$.
  - If q_j is wr_k in $\mathbf{N}^\perp$, we construe σ'' as a strategy for $(A_1 \times \cdots \times A_n \times \mathbf{C}) \Rightarrow B$ and we have: $\sigma' = [\![\mathtt{seq}_B\ (\mathtt{assign}\ (x_j M'_1 \cdots M'_l)\ \mathtt{k})\ M_\bullet]\!]$.

The final case we must deal with is when B is $\mathbf{Var}$. In this case, $\sigma = \langle \sigma_w, \sigma_r \rangle$. Once again, we can define σ_r by some term M_r. For σ_w, let ℓ be such that, for all $\ell' \geq \ell$, σ_w has no response to $\mathsf{wr}_{\ell'}$. We can now construe σ_w as a strategy for $(A_1 \times \cdots \times A_n) \Rightarrow \mathbf{C}^\ell$ which can hence we written in the form $\langle \sigma_w^0, \ldots, \sigma_w^{\ell-1} \rangle$. By the inductive hypothesis, we have a term M_w^i for each of these and:

$$\sigma' = [\![\mathtt{mkvar}\ (\lambda x.\ \mathtt{case}_{\ell,\mathtt{Nat}}\ x\ M_w^0 \cdots M_w^{\ell-1})\ M_r]\!].$$

Finally, in each case, we recover a term defining σ by λ-abstracting $x_1, \ldots, x_i$. ■

Corollary 3.8.9 If $\sigma : [\![\Gamma]\!] \Rightarrow [\![T]\!]$, where Γ and T are respectively an IA context and type, is a compact, deterministic strategy satisfying P-visibility and P-bracketing then σ is definable in IA.

Proof We apply the innocent factorization, yielding an innocent strategy σ' for $([\![\Gamma]\!] \times \mathbf{Var}) \Rightarrow [\![T]\!]$, definable in IA – `new` by some: $\Gamma, v : \mathtt{Var} \vdash M : T$. Then, for example, if $T = \vec{T} \rightarrow \mathtt{Com}$, we define σ by: $\Gamma \vdash \lambda\vec{x}.\ \mathtt{new}\ (\lambda v.\ M\vec{x})$. ■

Corollary 3.8.10 If $\sigma : [\![\Gamma]\!] \Rightarrow [\![T]\!]$ is a compact strategy satisfying P-visibility and P-bracketing then σ is definable in EIA.

Proof By the deterministic factorization, σ can be decomposed as the oracle strategy composed with a compact, deterministic strategy (satisfying P-visibility and P-bracketing) σ' for $([\![\Gamma]\!] \times \mathbf{N}) \Rightarrow [\![T]\!]$. By the previous result, σ' is definable in IA by some term: $\Gamma, x : \mathtt{Nat} \vdash M : T$; and so we can define σ by: $\Gamma \vdash (\lambda x.\ M)\mho$, where $\mho$ is: $\Gamma \vdash \mathbf{Y}_{\mathtt{Nat}}(\lambda z.\ 0\ \mathtt{or}\ \mathtt{succ}\ z)$. ■

Aside This definability result is not as sharp as it can be—obviously there are many non-compact strategies that are definable. This idea can be formalized by considering the *effective* strategies [1, 47], a class strictly extending the compact strategies, and proving that *precisely* the effective (innocent and well-bracketed) strategies are definable in PCF, *i.e.* a universality result.

3.8.4 The intrinsic collapse

Let $\mathcal{C}$ be a Cartesian closed category* and $\mathcal{T}$ be some object of $\mathcal{C}$. If $f : A \to B$ is an arrow of $\mathcal{C}$, the arrow 'f' $: \mathbf{1} \to (A \Rightarrow B)$, defined as $\Lambda(\pi'_{\mathbf{1},A}\ ;\ f)$, is known as the ***name*** of f. Given parallel arrows $f, g : A \to B$ of $\mathcal{C}$, we define:

$$f \simeq g \quad \text{iff} \quad \forall t : (A \Rightarrow B) \to \mathcal{T}.\ `f\text{'}\ ;\ t = `g\text{'}\ ;\ t.$$

This defines an equivalence relation on each homset of $\mathcal{C}$. The following lemma shows that this relation is "compatible" with composition.

Lemma 3.8.11 If $f_1, f_2 : A \to B$ and $g_1, g_2 : B \to C$ are such that $f_1 \simeq f_2$ and $g_1 \simeq g_2$ then $f_1\ ;\ g_1 \simeq f_2\ ;\ g_2$.

Proof We swap f_2 for f_1 by calculating:

$$\begin{aligned} `f_1\ ;\ g_1\text{'}\ ;\ t &= `f_1\text{'}\ ;\ (\mathsf{id}_A \Rightarrow g_1)\ ;\ t \\ &= `f_2\text{'}\ ;\ (\mathsf{id}_A \Rightarrow g_1)\ ;\ t \\ &= `f_2\ ;\ g_1\text{'}\ ;\ t. \end{aligned}$$

The other half follows similarly, using '$f_2\ ;\ g_1$' = 'g_1' ; $(f_2 \Rightarrow \mathsf{id}_C)$. ■

We can therefore define the ***intrinsic quotient*** of $\mathcal{C}$ (with respect to $\mathcal{T}$) just as a standard quotient category [59]. The resulting category is also Cartesian closed; we denote it by $\mathcal{C}/\simeq$.

*In fact, it suffices to ask for an SMCC but we have no need of the extra generality.

Order enrichment If our starting category $\mathcal{C}$ is (the underlying category of) a POSet-enriched category, we can define a preorder on each homset of $\mathcal{C}$ by:

$$f \precsim g \quad \text{iff} \quad \forall t : (A \Rightarrow B) \to \mathcal{T}.\ `f\text{'} ; t \leq `g\text{'} ; t,$$

where $\leq$ is the order inherited from the POSet-enrichment. Clearly, $f \simeq g$ if, and only if, $f \precsim g$ and $g \precsim f$. It follows that $\mathcal{C}/\simeq$ is also POSet-enriched.

If we further assume that $\mathcal{C}$ is CPO-enriched, we might hope that $\mathcal{C}/\simeq$ would be too. This is a bit too much to hope for, as it turns out, but we can verify a weaker property, that of *rationality* [1, 63], which is sufficient for our purposes.

Suppose that $\mathcal{D}$ is a POSet-enriched Cartesian closed category. We say that it's pPO-enriched—"pointed-POSet-enriched"—iff each homset $\mathcal{D}(A, B)$ has a least element, written $\perp_{A,B}$, for which composition is "diagram order" right-strict: for any $f : A \to B$, we have that $f ; \perp_{B,C} = \perp_{A,C}$.

Given any $f : A \to A$ in $\mathcal{D}$, we define an ω-chain in the following fashion.

$$\begin{aligned} f_{(0)} &= \perp_{\mathbf{1},A} \\ f_{(n+1)} &= f_{(n)} ; f. \end{aligned}$$

A pPO-enriched Cartesian closed category $\mathcal{D}$ is ***rational*** iff, for any such f and $g : A \to B$, we have:

- the set $\Delta_g = \{f_{(n)} ; g \mid n \in \mathbf{N}_0\}$ has a lub, written $\bigsqcup \Delta_g$;
- $\bigsqcup \Delta_g = (\bigsqcup_{n \in \mathbf{N}_0} f_{(n)}) ; g$.

In other words, all the chains that we "care about" have lubs that are preserved by composition.

Lemma 3.8.12 If $\mathcal{C}$ is CPO-enriched then $\mathcal{C}/\simeq$ is rational.

Proof Let Δ be a $\leq$-directed set in $\mathcal{C}(A, B)$. Since $f \leq g$ is easily seen to imply $f \precsim g$, we know that the $\leq$-lub of Δ, written as $\bigsqcup \Delta$, is an upper bound with respect to $\precsim$. We wish to show that it's the lub.

First of all, for any $t : (A \Rightarrow B) \to \mathcal{T}$, we have $`\bigsqcup \Delta\text{'} ; t = \bigsqcup_{f \in \Delta}\{`f\text{'} ; t\}$, by continuity. But, if g is any $\precsim$-upper bound for Δ, we know that for each $f \in \Delta$, $f \precsim g$. By definition, for any $t : (A \Rightarrow B) \to \mathcal{T}$ and $f \in \Delta$, $`f\text{'} ; t \leq `g\text{'} ; t$. This means that $\bigsqcup_{f \in \Delta}\{`f\text{'} ; t\} \leq `g\text{'} ; t$ and, by definition, $\bigsqcup \Delta \precsim g$.

So, the lub of any $\leq$-directed set is also a lub with respect to $\precsim$. Since any ω-chain $\{f_{(n)}\}$, as defined above, is $\leq$-directed, it has a $\precsim$-lub and the rest follows. ∎

In the light of this lemma, we can view CPO-enrichment as a sufficient (although unnecessary) property of an intensional semantic universe ensuring "enough" fixpoints when we pass to the intrinsic quotient.

3.8.5 Full abstraction

The soundness result of section §3.8.2 holds in $\mathbf{C}$ and in the subcategories $\mathbf{C}_\nu$, $\mathbf{C}_b$ and $\mathbf{C}_{\nu b}$ of P-vis, P-brk and P-vis & P-brk strategies. In this final section, we restrict our attention to $\mathbf{C}_{\nu b}$, applying the intrinsic quotient construction to this category using the flat arena $\mathbf{C}$ as our "testing object".

By lemma 3.8.12, we know that $\mathbf{C}_{\nu b}/\simeq$ is rational. It's also clear that, for any flat arena $\mathcal{F}$ and strategies σ and τ for $\mathbf{1} \Rightarrow \mathcal{F}$, $\sigma \simeq \tau$ if, and only if, $\sigma = \tau$. In particular, the arenas $\mathbf{N}$ and $\mathbf{C}$ survive the quotient.

We write $\langle\sigma\rangle$ for the $\simeq$-equivalence class containing σ. We define the semantics of EIA in $\mathbf{C}_{\nu b}/\simeq$ simply by setting $\mathcal{E}[\![M]\!] = \langle\mathcal{I}[\![M]\!]\rangle$ where the $\mathcal{E}$ and $\mathcal{I}$ are intended to suggest "extensional" and "intensional", $\mathcal{I}[\![M]\!]$ being just the semantics of §3.8.1. It's easy to see that the soundness theorem migrates to the new model. If M and N are closed terms of type T such that $\mathcal{E}[\![M]\!] \preccurlyeq \mathcal{E}[\![N]\!]$ and, for some program context $C[-]$, $C[M] \Downarrow^{\text{may}}$ then this context corresponds to some test $\alpha : [\![T]\!] \to \mathbf{C}$. But, by definition of $\preccurlyeq$, $\mathcal{I}[\![M]\!] ; \alpha \leq \mathcal{I}[\![N]\!] ; \alpha$ and so, by adequacy of $\mathcal{I}[\![-]\!]$, $C[N] \Downarrow^{\text{may}}$.

In $\mathbf{C}/\simeq$, the converse of this result also holds.

Theorem 3.8.13 (full abstraction) If M and N are programs of EIA then $\mathcal{E}[\![M]\!] \preccurlyeq \mathcal{E}[\![N]\!]$ if, and only if, $M \lesssim_{\text{may}} N$.

Proof We've already seen the left-to-right direction. For the converse, we prove the contrapositive. Suppose $\mathcal{E}[\![M]\!] \not\preccurlyeq \mathcal{E}[\![N]\!]$ so, for some α, $\mathcal{I}[\![M]\!];\alpha \not\leq \mathcal{I}[\![N]\!];\alpha$. This α can clearly be taken to be compact and, by the definability theorem, we have $\alpha = \mathcal{I}[\![x : T \vdash C[-] : \mathtt{Com}]\!]$, so that $\mathcal{E}[\![C[M]]\!] \simeq \mathcal{I}[\![M]\!] ; \alpha$ and similarly for N. This means that $\mathtt{qa} \in \mathcal{I}[\![M]\!] ; \alpha$ but $\mathtt{qa} \notin \mathcal{I}[\![N]\!] ; \alpha$ and, by consistency and adequacy, $C[M] \Downarrow^{\text{may}}$ but $C[N] \Uparrow$. Therefore $M \not\lesssim_{\text{may}} N$, as desired. ■

Chapter 4

Full abstraction for finite nondeterminism

In the previous chapter, we developed the basic theory of HO-style games with a particular focus on one axis of the "semantic cube", namely determinism. By proving a factorization theorem—any strategy can be expressed as the composite of the oracle $\top_{\mathbf{N}}$ with a deterministic strategy—we were able to give a model of EIA which was fully abstract with respect to may contextual equivalence.

Now, while this model is perfectly satisfactory for reasoning about observable properties (what values can this program evaluate to?) it is rather deficient when it comes to subtler aspects of nondeterministic processes. This is essentially due to the fact that the model can't distinguish between a *reliable* program, such as '`tt`', and an *unreliable* program, such as '`tt or` Ω'. It therefore seems reasonable to ask whether there is some way in which we might enrich this basic model in order to characterize the more discriminating may/must contextual equivalence which, as we noted in chapter 2, makes precisely these kinds of distinctions.

Our work in this chapter demonstrates that this is indeed possible. We rework the idea of *divergences* from CSP in the context of game semantics and use this to build a more refined semantic universe that harbours a model of EIA which is fully abstract with respect to may/must contextual equivalence.

4.1 Unreliable strategies

4.1.1 Traces & divergences

In the previous chapter, we defined a strategy to be a non-empty set of even-length legal plays closed under even-length prefixes—in other words, some set of legal plays where Player moved last. We use such plays—which we'll typically refer to as *traces*—to model the potential convergent behaviours of a process: the usual "crib-book interpretation" of a strategy is that, if $sab \in \sigma$ and Opponent (*i.e.* the Environment) plays the move a in context s then Player (*i.e.* the System) *may* respond with the move b. If there's also some other $sac \in \sigma$ then Player might instead choose to play c in this situation.

This kind of "observable" nondeterminism is exemplified by '`tt or ff`'. But now, consider the term '`tt or` Ω'. There is a definite sense in which, despite there being only one possible observable output, this also is nondeterministic: this time, Player's choice is whether or not to reply, rather than a choice between alternative possible replies. This implicit nondeterminism is completely ignored by our basic model since we only consider traces, *i.e.* plays, that end with a Player move—and there is no Player move corresponding to "no comment". Adding such a move leads to various technical complications—not to mention the real philosophical objection to affording divergence an observable status.

Indeed, this last point is an important one. In general, Player's invisible option of "giving up" might not be a matter of choice; she may have no other option. Viewed in this light, it seems more reasonable to regard such unreliability as a phenomenom which is *provoked* by the Environment rather than *chosen* by the System. This judgement is probably a little harsh on the Environment. Nevertheless, it's a fruitful perspective to take since it ties in nicely with the question underlying must contextual equivalence: what actions can the Environment perform without running the risk of divergence?

One way to answer this question is to specify those interactions of the System and its Environment that may result in divergence. The minimal such interactions map out the boundary between the System's reliable and unreliable behaviour. This idea has been used to good effect in CSP where a process is typically specified by, amongst other things, a set of *divergences*. These are just traces intended to represent "points" at which a process may "go wrong".

In the light of the above discussion, we take the view that a divergence should be a trace ending with an action by the Environment. In the context of game semantics, this means that a divergence is a legal play ending with an Opponent move, *i.e.* an odd-length play. With these remarks in mind, we proceed by reformulating strategies as pairs, the first component being the usual set of even-length plays and the second component being a set of odd-length plays representing points of unreliability.

Definition A ***strategy*** σ on an arena A is defined to be a couple (T_σ, D_σ). The first component T_σ is a set of even-length legal plays of A, known as the ***traces*** of σ, satisfying

(**t1**) $\varepsilon \in T_\sigma$

(**t2**) if $sab \in T_\sigma$ then $s \in T_\sigma$.

In other words, T_σ is just a strategy in the sense of the previous chapter. We again write $\mathsf{dom}(\sigma)$ for the ***domain*** of σ, *i.e.* the set $\{sa \mid \exists b.\ sab \in T_\sigma\}$ of (odd-length) plays that σ will respond to, and use $\mathsf{cc}(\sigma)$ to denote $T_\sigma \cup \mathsf{dom}(\sigma)$, the so-called ***contingency closure*** of σ. Given $sa \in \mathsf{dom}(\sigma)$, let $\mathsf{rng}_\sigma(sa) = \{sab \mid sab \in T_\sigma\}$; this is known as the ***range*** of σ at sa.

The second component D_σ is a set of odd-length legal plays of A, known as the ***divergences*** of σ, satisfying

(**d1**) if $s \in T_\sigma$, $sa \in L_A$ and $sa \notin \mathsf{dom}(\sigma)$ then $\exists d \in D_\sigma.\ d \sqsubseteq sa$

(**d2**) if $sa \in D_\sigma$ then $s \in T_\sigma$.

The first of these axioms says that, if we play according to σ and then Opponent makes some move to which we have no reply then we have no option other than to give up. In other words, if a program *cannot* respond at some point then this will be reflected by an appropriate divergence. The other axiom is really just a sanity condition. It could be dropped without fundamentally changing the character of the definition but, as there seems to be little sense in having divergences that are "unreachable", we've left it in.

We write $\mathsf{div}(\sigma)$ for the set of *minimal* divergences of σ. As we hinted at above, these turn out to be the important things; we'll make this more precise in §4.3 when we define the ordering on strategies. This (pre-)ordering gives rise to a notion of equivalence between strategies which can be precisely characterized in terms of the minimal divergences.

Remark One of the main novelties of game semantics is the fundamental separation that it makes between Opponent and Player. It's worth noting how this helps to clarify two orthogonal aspects of nondeterministic processes: observable nondeterminism is the responsibility of Player whereas unobservable nondeterminism is the responsibility of Opponent. In particular, even if a strategy has a deterministic trace set, it may still have some divergences and, therefore, might be considered nondeterministic.

4.1.2 Finite-branching

We saw in chapter 2 that recursion doesn't provide a reliable route from finite to countable nondeterminism. This suggests that we might restrict our attention to a class of "finite-branching" strategies that would, typically, exclude the likes of $(\{\varepsilon\} \cup \{\mathsf{q}n \mid n \in \mathbf{N}\}, \varnothing)$. Clearly, by "finite-branching", we don't mean that our trace sets *cannot* have infinite branching, but that all *reliable* behaviours must have this property: if we want countable nondeterminism, we must pay the price of possible divergence.

We can easily formalize this idea: given a strategy σ on an arena A, we say that σ is ***finite-branching*** iff

(**fb**) if $\mathsf{rng}_\sigma(sa)$ is infinite then $\exists d \in D_\sigma.\ d \sqsubseteq sa$.

The concept of a finitely branching process has often appeared before, in various essentially equivalent guises, sometimes for technical reasons, as in CSP, and at other times with more of a philosophical motive, as in [31].

Central to the idea of finite-branching is the observation that a process trying to make a choice between an infinity of options may take a long time to make up its mind. Another way of looking at this aspect is that, if a finitely branching process can be arbitrarily delayed at some point then it can, in fact, diverge at that point. This is not true in general: it's possible to conceive of processes that may take arbitrarily long to finish their work but for which we have a guarantee that they *will* eventually stop.

In general then, we shouldn't think of our divergences as representing points at which a *finite but unbounded* delay may occur, but rather as representing points where a process is subject to a *potentially infinite* delay. In the sequel, we will—for the most part—only consider finite-branching strategies where this distinction becomes redundant. However, there are places where things only work because of the restriction to finite-branching strategies, so it's important to be aware of this distinction and how it relates to such pathologies. Moreover, this distinction is crucially important in the presence of countable nondeterminism where it is intimately related to the question of *fairness* [76].

4.2 Composition of strategies

We now turn to the question of how we should compose two strategies, $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$. It seems clear that the trace set $T_{\sigma;\tau}$ of the composite should be defined simply as the composition, in the sense of chapter 3, of the trace sets of σ and τ, *i.e.* $T_\sigma ; T_\tau$. The interesting question is: when should a divergence arise in the composite of σ and τ?

4.2.1 Unreliability & livelock

When we compose two strategies, each becomes a *partial* Environment for the other: the response of the composite strategy to a given move in a given context is determined by the interaction that ensues as each strategy takes its next input to be the last output of the other. This interaction ends when one or other of our strategies makes its next move "in the outside", *i.e.* as the "external" Player rather than as an "internal" Opponent. (This is nothing more than an alternative spin on the figure-of-eight.)

We would therefore expect to see a divergence in the composite of σ and τ if one strategy, in the course of such an interaction, produces an input for the other which causes divergence in that strategy. For example, if we provide an input 'zero' to a strategy for 'predecessor' which is defined only on inputs of at least 'one', we could get the following interaction.

$$\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{N} & ; & \mathbf{N} & \Rightarrow & \mathbf{N} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{0} & & & & \\
 & & & & \mathsf{0} & &
\end{array}$$

Since $\mathsf{qq0}$ would be obliged, according to axiom (**d1**), to be a divergence of our 'predecessor' strategy, we would expect this interaction to induce a divergence, q, in the composite. If we instead provide the input 'zero or one', we would still expect q to be a divergence but the trace $\mathsf{q0}$ would now be in the composite.

There is, however, a second way in which a divergence can arise. Consider the strategy $(\{\varepsilon, \mathsf{qff}, \mathsf{qffqff}, \ldots\}, \varnothing)$ on $\mathbf{1} \Rightarrow \mathbf{B}$ and $(\{\varepsilon\} \cup \{\mathsf{qq}(\mathsf{ffq})^\star\} \cup \{\mathsf{qq}(\mathsf{ffq})^\star \mathsf{tttt}\}, \varnothing)$ on $\mathbf{B} \Rightarrow \mathbf{B}$. This latter strategy represents a process that repeatedly asks for an input until it receives 'true' whence it outputs 'true'. It's therefore no surprise that disaster strikes when we compose these two strategies...

$$\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{B} & ; & \mathbf{B} & \Rightarrow & \mathbf{B} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{ff} & & & & \\
 & & & & \mathsf{ff} & & \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{ff} & & & & \\
 & & & & \mathsf{ff} & & \\
 & & & \vdots & & &
\end{array}$$

Since ε is the only possible trace of the composite, axiom (**d1**) forces q to be a divergence. And yet, neither of the two strategies has any divergences.

The basic problem here is that our strategy on $\mathbf{B} \Rightarrow \mathbf{B}$ is blissfully unaware that it's never going to receive a 'true' so it just keeps on asking for more and more inputs. This is a classical example of *livelock*. We can see from this example that livelock is a situation where unreliability in the composite is caused, not by any unreliability in the constituent strategies, but by the fact that the constituents can continue their interaction indefinitely.

4.2.2 Formalizing composition

We must first introduce a little more notation for talking about strings. Given our alphabet Σ, we write Σ^∞ for the set of all infinite strings over Σ. Given some $s \in \Sigma^\infty$ and an $i \in \mathbf{N}$, we write s_i for the ith symbol in s (starting from 1) and $s_{\leqslant i}$ for the (finite) prefix of s with length i.

Infinite interactions We now define an ***infinite interaction*** in arenas A, B and C to be an infinite string $u \in (M_A + M_B + M_C)^\infty$ equipped with "justification pointers" from all (occurrences of) moves not initial in C such that $u \restriction A, C \in L_{A \Rightarrow C}$ and, for all $i \in \mathbf{N}_0$, $u_{<i} \restriction A, B \in L_{A \Rightarrow B}$ and $u_{<i} \restriction B, C \in L_{B \Rightarrow C}$; equivalently, for all $i \in \mathbf{N}$, $u_{<i} \in \mathsf{int}(A, B, C)$. We write $\mathsf{int}_\infty(A, B, C)$ for the set of all infinite interactions of arenas A, B and C.

Since, for any $u \in \mathsf{int}_\infty(A, B, C)$, we ask that $u \restriction A, C \in L_{A \Rightarrow C}$, this means that there can only be a finite amount of play in A and C; hence there must be an "infinite tail" in B. The intention of this definition is to formally capture the idea of "infinite chattering" (the CSP terminology) or "divergence by an infinite τ-computation" (the CCS terminology), both of which are manifestations of the general idea of livelock.

Any $u \in \mathsf{int}_\infty(A, B, C)$ can be equivalently thought of as an infinite, strictly increasing (under the prefix ordering) sequence (u_n) of elements of $\mathsf{int}(A, B, C)$ such that there exists an $N \in \mathbf{N}_0$ such that for all $n \geq N$, $u_n \restriction A, C = u_N \restriction A, C$.

Formalizing composition Suppose we have strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$. We wish to define their composite $\sigma ; \tau$. As we've already noted, $T_{\sigma;\tau}$ is naturally defined to be $T_\sigma ; T_\tau$.

Given the above discussion, we would expect two clauses in the definition of the divergences of $\sigma ; \tau$. Firstly, we define the ***finitely generated*** divergences, written $D_\sigma \between D_\tau$, to be the set of all $u \in \mathsf{int}(A, B, C)$ such that

$$(u \restriction A, B \in T_\sigma \wedge u \restriction B, C \in D_\tau) \vee (u \restriction A, B \in D_\sigma \wedge u \restriction B, C \in T_\tau).$$

In other words, u consists of a trace from σ and a divergence from τ, or *vice versa.* This captures all unreliability caused by unreliability in one or other of σ and τ. Secondly, the ***infinitely generated*** divergences, written $T_\sigma \between T_\tau$, are defined to be the set of all $u \in \mathsf{int}_\infty(A, B, C)$ such that, for all $i \in \mathbf{N}_0$,

$$u_{\leqslant i} \restriction A, B \in \mathsf{cc}(\sigma) \wedge u_{\leqslant i} \restriction B, C \in \mathsf{cc}(\tau).$$

In other words, u is an interaction of σ and τ that has an infinite tail in B and, as such, there is the possibility of livelock.

We now complete the definition of $\sigma ; \tau$ by specifying its divergences as:

$$D_{\sigma;\tau} =_{\mathrm{df}} \{u \restriction A, C \mid u \in D_\sigma \between D_\tau \vee u \in T_\sigma \between T_\tau\}.$$

Aside The above example of two strategies engaging in infinite chattering can be simply modified by allowing the "input" strategy to nondeterministically produce either 'true' or 'false'. A typical interaction now takes the following general shape.

$$
\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{B} & ; & \mathbf{B} & \Rightarrow & \mathbf{B} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{ff} & & & & \\
 & & & & \mathsf{ff} & & \\
 & & & \vdots & & & \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{ff} & & & & \\
 & & & & \mathsf{ff} & & \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{tt} & & & & \\
 & & & & \mathsf{tt} & & \\
 & & & & & & \mathsf{tt}
\end{array}
$$

So we might expect to see an output of 'true' after any *finite* delay (including no delay at all if the input produces 'true' first time). But the possibility of livelock remains if our input is *unfairly* implemented by a process that never produces 'true'. So, the composite strategy here, while it certainly "may converge" (*i.e.* it contains the trace $\mathsf{q}\mathsf{tt}$), also "may diverge" (*i.e.* the string q is an (infinitely generated) divergence). This seems entirely reasonable given our above discussion of livelock. Nonetheless, there seems to be a difference between our two examples: in the first, it's uncontroversial that there will never be an observable response; this is not so in the latter case.

This question of *fairness* is a notorious problem, largely because it's difficult to formally get hold of just what it is that's different about these two examples. David Park's work [76] showed one approach, in the context of relational semantics. A contrasting approach is to consider our nondeterministic input as a *probabilistic** strategy. If this is a "fair coin" (*i.e.* either face is equally likely on any given toss) then the probability of the composite producing 'true' converges, in the limit, to unity. In the jargon of randomized algorithms, we would say that it terminates *almost surely*.

*Although we don't do so here, this notion of "probabilistic strategy" can be formalized and the example mentioned here does indeed work as advertised. This connection is the subject of ongoing work in conjunction with Vincent Danos.

This issue is also intimately tied in with the question of *countable*, or *unbounded*, nondeterminism. To see this, we can replace our strategy on **B** ⇒ **B** with the strategy on **B** ⇒ **N** containing all plays of the following form...

$$\begin{array}{ccc}
\mathbf{B} & \Rightarrow & \mathbf{N} \\
 & & \mathsf{q} \\
\mathsf{q} & & \\
\mathsf{ff} & & \\
\mathsf{q} & & \\
\vdots & & \\
\mathsf{ff} & & \\
\mathsf{q} & & \\
\mathsf{tt} & & \\
 & & \mathsf{n}
\end{array}$$

...where n is the number of times we got 'false' before we got a 'true'. Any "fair" evaluation of this composite would be *guaranteed* to terminate but the output would, in principle, be unbounded.

By the way, this example also demonstrates that *finitely branching* trace sets are not, in general, preserved by composition. This is basically because no account is taken of the "infinite trace" $\mathsf{qqffqffq}\ldots$ with much the same result as when we try to compose ***total*** strategies (where Player can always carry on).

4.2.3 Well-defined composition

We must now demonstrate that composition makes sense. There's nothing new to prove about the trace composition; we just need to ensure that our composite set of divergences is well-defined.

Recall the state diagram for legal interactions from the previous chapter. Now, let $u \in \mathsf{int}(A, B, C)$ and $a \in u$ be a generalized O-move with component X. If a is a move of A or C, we know that $u_{<a}$ is in the "top state" (OOO) of the state diagram, *i.e.* $u_{<a} \upharpoonright X$ and $u_{<a} \upharpoonright Y$ both have even length. Otherwise, if a is a move of B then it must be a P-move in component Y, *i.e.* $u_{\leqslant a} \upharpoonright X$ has odd length but $u_{\leqslant a} \upharpoonright Y$ has even length. In other words, a legal interaction has odd length in at most one component at a time and, if one or other component does have odd length, this means that the legal interaction is in either the "middle" or the "bottom" state of the state diagram, *i.e.* (OPP) or (POP). This observation can be summarised in the following lemma.

Lemma 4.2.1 If $u \in \mathsf{int}(A, B, C)$ and $a \in u$ is a generalized O-move with component X then $u_{\leqslant a} \upharpoonright Y$ has even length.

Proof Immediate from the state diagram. ■

One consequence of this is that any trace in a composite strategy is witnessed by a legal interaction in the (OOO) state and any finitely generated divergence is witnessed by a legal interaction in either the (OPP) or the (POP) state. An infinitely generated divergence corresponds to an infinite sequence of legal interactions that oscillate between (OPP) and (POP).

We are now in a position to prove that composition is well-defined.

Proposition 4.2.2 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ then their composite $\sigma ; \tau$ is a well-defined strategy for $A \Rightarrow C$.

Proof As we've already noted, there's nothing new to check for the traces, so we proceed immediately to the divergences.

First of all, we need to check that all $d \in D_{\sigma;\tau}$ have odd length. If d is finitely generated, it comes from some $u \in D_\sigma \between D_\tau$. If $u \upharpoonright A, B$ has even length then $u \upharpoonright B, C$ has odd length. So, if the restriction to A has even length, so does the restriction to B and hence the restriction to C must have odd length; so $u \upharpoonright A, C$ has odd length, as required. All the other cases use this same "parity" argument. If d is infinitely generated, it is immediate from the state diagram for legal interactions that the last move of A or C was an O-move and hence d has odd length. So we now know that the divergences of $\sigma ; \tau$ are indeed a set of odd-length legal plays. We just have to check that they satisfy (**d1**) and (**d2**).

For (**d1**), suppose that $s \in T_{\sigma;\tau}$, $sa \in L_{A \Rightarrow C}$ but $sa \notin \mathsf{dom}(\sigma ; \tau)$. Let $u \in T_\sigma \parallel T_\tau$ be a witness for $s \in T_{\sigma;\tau}$. Now, since $sa \notin \mathsf{dom}(\sigma ; \tau)$, it must be the case that all extensions $uam_1 \cdots m_k \cdots \in \mathsf{cc}(\sigma) \parallel \mathsf{cc}(\tau)$ of u either get "stuck", *i.e.* one or other of σ or τ is confronted with something to which they have no response, or it continues indefinitely within B. Suppose we're in the first case; so we have some $uam_1 \cdots m_k$ such that $uam_1 \cdots m_k \upharpoonright X$ has odd length but the strategy associated with component X has no response, which we'll write as $uam_1 \cdots m_k \upharpoonright X \notin \mathsf{dom}(X)$. By (**d1**), we have some $tb \in D_X$ such that $tb \sqsubseteq uam_1 \cdots m_k \upharpoonright X$. Since $uam_1 \cdots m_k \upharpoonright Y \in T_Y$ (it's in $\mathsf{cc}(\tau)$ and, by the previous lemma, it has even length) we apply (**t2**) and the previous lemma to get $(uam_1 \cdots m_k)_{\leqslant b} \upharpoonright Y \in T_Y$. Therefore $(uam_1 \cdots m_k)_{\leqslant b} \in D_\sigma \between D_\tau$, witnessing some prefix of sa in $D_{\sigma;\tau}$. If we're in the second case, we must have an infinite interaction $u_\infty = uam_1 \cdots m_k \cdots$ such that all finite prefixes $u' = (u_\infty)_{\leqslant i}$ (for $i \in \mathbf{N}$) satisfy $u' \upharpoonright A, B \in \mathsf{cc}(\sigma)$ and $u' \upharpoonright B, C \in \mathsf{cc}(\tau)$. Moreover, we know that all the m_is are moves of B so $sa = u_\infty \upharpoonright A, C \in D_{\sigma;\tau}$.

For (**d2**), suppose we have some $sa \in D_{\sigma;\tau}$. This must be witnessed either by some $u \in D_\sigma \between D_\tau$ or by some $u_\infty \in T_\sigma \between T_\tau$. In either case, we can apply lemma 3.2.1 to establish that $u_{<a} \upharpoonright X \in L_X^{\mathsf{even}}$ for each component. But this means that $u_{<a} \upharpoonright X \in T_X$ for both components (by various combinations of (**t2**) and (**d2**) as appropriate) so $s = sa_{<a} = u_{<a} \upharpoonright A, C \in T_{\sigma;\tau}$. ∎

4.2.4 Zig-zags & copycats

Given an arena A, the obvious choice for the identity strategy is $(\mathsf{id}_A, \varnothing)$. Such a strategy, where Player always switches, is known as a ***zig-zag*** strategy. It should be clear that composing a strategy σ with a zig-zag strategy can *never* result in divergence by livelock: there can be at most two moves "in the middle" between any two consective moves "on the outside". If the zig-zag strategy has no divergences (as is the case with our proto-identity strategy above) then the only way a divergence can arise in this composite is as a result of a divergence in σ.

Lemma 4.2.3 For any arena B, the strategy $\mathsf{Id}_B = (\mathsf{id}_B, \varnothing)$ is well-defined and, for any strategy $\sigma : A \Rightarrow B$, we have $\mathsf{Id}_A \,;\, \sigma = \sigma = \sigma \,;\, \mathsf{Id}_B$.

Proof It's easy to see that Id_B is well-defined since Player always has a response in id_B. For the second part, consider $\sigma \,;\, \mathsf{Id}_B$. If $d \in D_{\sigma;\mathsf{Id}_B}$ then the above remarks tell us that it must be witnessed by some $u \in \mathsf{int}(A, B_\ell, B_r)$ such that $u \restriction A, B_\ell \in D_\sigma$. But $u \restriction B_\ell, B_r \in \mathsf{id}_B$ so $d = u \restriction A, B_r = u \restriction A, B_\ell \in D_\sigma$. Similarly, if we start with some $d \in D_\sigma$ then it's easy to find $u \in \mathsf{int}(A, B_\ell, B_r)$ witnessing $d \in D_{\sigma;\mathsf{Id}_B}$. ■

We now have a robust notion of composition with the expected identities. We also have, for all arenas, an evident "diagonal" strategy Δ_A with traces derived from the diagonal strategies used in the previous chapter and, like the above identities, no divergences.

4.2.5 Finite-branching

We must now show that the condition (**fb**) is preserved by composition. Should we even expect this to be true? Well, if we start with composable strategies σ and τ that both satisfy this constraint, there are only two ways we can get infinite branching in their composite: *either* one or other of σ and τ is solely responsible, in which case the guilty strategy will already have a divergence which will induce a divergence in the composite; *or* they can conspire to produce an infinite number of responses by interacting for an unbounded amount of time, in which case we'll get a divergence by livelock.

So it seems reasonable that this constraint be preserved by composition; but we still need to formalize the above discussion. First of all, we prove a little lemma that's of use in a number of places to come.

Lemma 4.2.4 Suppose $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are finite-branching strategies. If $sab \in T_{\sigma;\tau}$ has an infinite number of witnesses then $\exists d \in D_{\sigma;\tau}.\ d \sqsubseteq sab$.

Proof Consider the set W, defined to be:

$$\{w \in \mathsf{int}(A, B, C) \mid w \restriction A, B \in \mathsf{cc}(\sigma) \wedge w \restriction B, C \in \mathsf{cc}(\tau) \wedge w \restriction A, C \sqsubseteq sab\}.$$

If there's some $w \in W$ such that $w \in D_\sigma \between D_\tau$ then this witnesses some prefix of sab in $D_{\sigma;\tau}$. Otherwise, applying (**fb**) in both σ and τ, the set W can be regarded as a finitely branching tree. By assumption, there are an infinite number of witnesses of sab; so W is an infinite tree. We now apply König's lemma to W, showing that it has an infinite path. But for any $w \in W$, $w \upharpoonright A, C$ has finite length (it's a prefix of sab) so our infinite path w_∞ must have an infinite tail in B. This means that $w_\infty \in T_\sigma \between T_\tau$ and hence we have some prefix of sab in $D_{\sigma;\tau}$. ■

We can now formally prove that the class of finite-branching strategies is closed under composition.

Proposition 4.2.5 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are both finite-branching then so is their composite $\sigma\,;\tau$.

Proof Suppose the cardinality of $\mathsf{rng}_{\sigma;\tau}(sa)$ is infinite. In particular, this means that $\mathsf{rng}_{\sigma;\tau}(sa)$ is non-empty, so $sa \in \mathsf{dom}(\sigma\,;\tau)$ and hence $s \in T_{\sigma;\tau}$.

We first consider the case where s has a finite number of witnesses. In this case, there must be some witness $w \in T_\sigma \| T_\tau$ for s which extends to witness a countable collection of plays sab_i (for $i \in \mathbf{N}_0$). This means that the set W, defined to be

$$\{\varepsilon\} \cup \{au \mid wau \in \mathsf{cc}(\sigma) \parallel \mathsf{cc}(\tau) \wedge (u \text{ contains at most one move of } A \Rightarrow C)\}$$

can be considered as an infinite tree. If any trace in this set is also in $D_\sigma \between D_\tau$ then it clearly witnesses $sa \in D_{\sigma;\tau}$. Otherwise, since σ and τ satisfy (**fb**), it must be the case that this tree is finitely branching and, by König's lemma, has an infinite path au_∞. By construction, this must be an infinite tail in B and hence wau_∞ lies in $T_\sigma \between T_\tau$, witnessing $sa \in D_{\sigma;\tau}$.

In the other case, where s has an infinite number of witnesses, we can simply apply lemma 4.2.4 to get an appropriate divergence and we're done. ■

4.3 Ordering strategies

4.3.1 The basic ordering

Suppose we have two strategies σ and τ on some arena A. When should we say that σ approximates τ? Well, it surely ought to be the case that whatever σ can observably do, τ should also be capable of doing, *i.e.* the traces of σ are contained within the traces of τ. In the presence of the prefix-closure axiom (**t2**), this can be equivalently stated as $\forall s \in T_\sigma.\ \exists t \in T_\tau.\ s \sqsubseteq t$ which brings out the similarity to the (lower half of the) Egli-Milner ordering.

What about the divergences? The analogy with the upper half of the Egli-Milner ordering suggests: $\forall e \in D_\tau.\ \exists d \in D_\sigma.\ d \sqsubseteq e$. Otherwise put, whenever τ is in a position where it may (already) have diverged, σ should also be so.

Is this sufficient? Well, with this definition we have (on the arena $\mathbf{1} \Rightarrow \mathbf{B}$) that $\sigma = (\{\varepsilon, \mathsf{q}\mathsf{tt}\}, \varnothing)$ approximates $\tau = (\{\varepsilon, \mathsf{q}\mathsf{tt}, \mathsf{q}\mathsf{ff}\}, \varnothing)$. If we were to compose each of these with $(\{\varepsilon, \mathsf{qq}, \mathsf{qq}\mathsf{tt}\mathsf{a}\}, \{\mathsf{qq}\mathsf{ff}\})$ on $\mathbf{B} \Rightarrow \mathbf{C}$, we'd get composites $(\{\varepsilon, \mathsf{q}\mathsf{tt}\}, \varnothing)$ and $(\{\varepsilon, \mathsf{q}\mathsf{tt}\}, \{\mathsf{q}\})$ respectively. But now their order is reversed! Now we don't want to reverse the order of our original two strategies; thus it seems that they ought to be incomparable (as are the corresponding points of the Plotkin power domain [80] over the two-point flat domain).

Our problem here is that τ is trying to "improve" σ by adding more observable behaviours; but since this runs the risk of causing more divergent behaviours in composition, we can only safely add new traces if σ (but not necessarily τ) could already have diverged by that point. This means that $(\{\varepsilon, \mathsf{q}\mathsf{tt}\}, \{\mathsf{q}\})$ should approximate $(\{\varepsilon, \mathsf{q}\mathsf{tt}, \mathsf{q}\mathsf{ff}\}, \varnothing)$ but $(\{\varepsilon, \mathsf{q}\mathsf{tt}\}, \varnothing)$ shouldn't. In the definition of the ordering, we therefore additionally stipulate that the convergent behaviour of σ is *exactly* copied in τ.

Definition Given strategies σ and τ on an arena A, we define

$$\begin{array}{lll} \sigma \leqslant^{\flat} \tau & \text{iff} & T_\sigma \subseteq T_\tau \\ \sigma \leqslant^{\sharp} \tau & \text{iff} & (\forall e \in D_\tau .\, \exists d \in D_\sigma .\, d \sqsubseteq e) \wedge \\ & & (sab \in T_\tau \wedge sab \notin T_\sigma) \Rightarrow \exists d \in D_\sigma .\, d \sqsubseteq sab \\ \sigma \leqslant^{\natural} \tau & \text{iff} & \sigma \leqslant^{\flat} \tau \wedge \sigma \leqslant^{\sharp} \tau. \end{array}$$

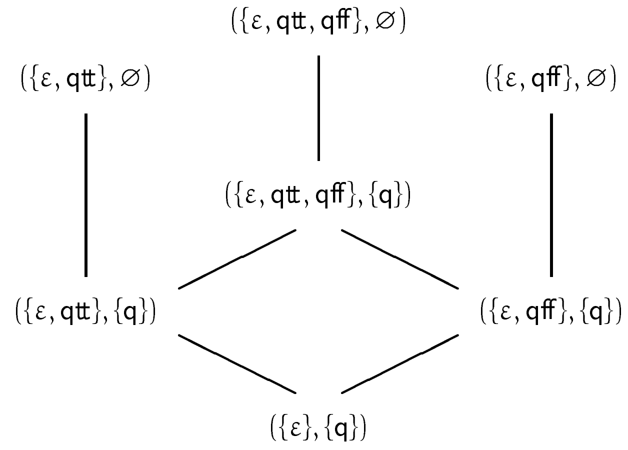
We sometimes refer to $\leqslant^{\flat}$ and $\leqslant^{\sharp}$ as the *lower* and *upper* orderings respectively. The second conjunct of the upper ordering captures the above-mentioned idea that τ can arbitrarily improve σ only once σ may have diverged.

This ordering is reminiscent of Roscoe's "alternative order for the failures model" of CSP [86]. This was developed as a means to extend the failures/divergences model to a wider class of processes: the usual subset/superset ordering required an axiom of finite branching for completeness; the coarser (in the sense that it orders fewer things) "alternative order" was complete even without this. In the categorical setting of game semantics, we're more concerned about monotonicity and continuity of our ordering with respect to composition. As the above example demonstrates, even monotonicity fails for the more simple-minded style of ordering. Our purpose is therefore to show that the coarser ordering is better behaved.

Lemma 4.3.1 For all arenas A, $\leqslant^{\natural}_A$ is reflexive and transitive.

Proof Reflexivity is completely trivial. For transitivity, suppose that $\sigma \leqslant^{\natural} \tau$ and $\tau \leqslant^{\natural} \upsilon$. It's easy to see that $\sigma \leqslant^{\flat} \upsilon$ and also that the first part of $\leqslant^{\sharp}_A$ is transitive. For the second part, suppose $sab \in T_\upsilon$ but $sab \notin T_\sigma$. If $sab \in T_\tau$ then, since $\sigma \leqslant^{\natural} \tau$, we have some $d \in D_\sigma$ such that $d \sqsubseteq sab$. Otherwise, if $sab \notin T_\tau$, since $\tau \leqslant^{\natural} \upsilon$, we have some $e \in D_\tau$ such that $e \sqsubseteq sab$ and, since $\sigma \leqslant^{\natural} \tau$, we have some $d \in D_\sigma$ such that $d \sqsubseteq e \sqsubseteq sab$ as required. ■

For example The simplest interesting example arises by considering the (well-opened) strategies on the **B**ooleans. Ordered by $\leqslant^{\natural}$, this is isomorphic to the Plotkin power domain over the two-point flat domain. Note that the strategies with no divergences are all maximal and incomparable under this ordering.



Although in this example, $\leqslant^{\natural}$ gives rise to a partial order, this isn't true in general. Consider $(\mathsf{id}_{\mathbf{C}}, \{\mathsf{q}\})$ and $(\mathsf{id}_{\mathbf{C}}, \{\mathsf{q}, \mathsf{qqa}\})$ on $\mathbf{C} \Rightarrow \mathbf{C}$. Each of these approximates the other and, furthermore, it seems natural to consider these two strategies to be "equivalent" since they have the same (potential) observable behaviours and are both immediately unreliable. Concretely, they both have the same set of minimal divergences, namely $\{\mathsf{q}\}$, and their extension-closures are therefore also the same, *i.e.* $\{\mathsf{q}, \mathsf{qqa}\}$. Rather than fix, in advance, a choice of concrete representation for strategies, we choose instead to introduce a notion of equivalence. The reason for this is largely pragmatic: different representations can be useful in different contexts and no one choice is ideal for all circumstances.

4.3.2 Equivalence & monotonicity

We write $\mathsf{Str}(A)$ for the set of strategies for A quotiented by the equivalence:

$$\sigma =^{\natural} \tau \text{ iff } \sigma \leqslant^{\natural} \tau \wedge \tau \leqslant^{\natural} \sigma.$$

We can similarly define $=^{\flat}$ and $=^{\sharp}$. For entirely standard order-theoretic reasons, $\leqslant^{\natural}$ lifts to be a partial ordering on $\mathsf{Str}(A)$. We'll denote by $\langle\sigma\rangle$ the $=^{\natural}$-equivalence class containing σ.

If we're going to work with strategies upto $=^{\natural}$ then it's important that composition also "lifts up" in a satisfactory way. In other words, we'd like composition to be a monotone operation with respect to $\leqslant^{\natural}$.

Proposition 4.3.2 If $\sigma, \sigma' : A \Rightarrow B$ and $\tau, \tau' : B \Rightarrow C$ are strategies such that $\sigma \leqslant^{\natural} \sigma'$ and $\tau \leqslant^{\natural} \tau'$ then $\sigma ; \tau \leqslant^{\natural} \sigma' ; \tau'$.

Proof We need to check that $\sigma\,;\tau \leqslant^\flat \sigma'\,;\tau'$ and $\sigma\,;\tau \leqslant^\sharp \sigma'\,;\tau'$. The first of these is obvious since any $u \in T_\sigma \parallel T_\tau$ is also in $T_{\sigma'} \parallel T_{\tau'}$. The second splits into two parts, according to the two halves of the definition of $\leqslant^\natural$. It suffices to show monotonicity in each argument separately. We only give one case, the one where τ is fixed; the other follows similarly.

For the first part, suppose $e \in D_{\sigma';\tau}$. If this is witnessed by $u \in D_{\sigma'} \between D_\tau$, there are two cases: *either* $u \restriction A, B \in D_{\sigma'}$ and $u \restriction B, C \in T_\tau$ *or* $u \restriction A, B \in T_{\sigma'}$ and $u \restriction B, C \in D_\tau$. For the first subcase, applying $\sigma \leqslant^\sharp \sigma'$ gives some $sa \in D_\sigma$ such that $sa \sqsubseteq (u \restriction A, B)$. Consider $u_{\leqslant a}$. By lemma 4.2.1, $u_{\leqslant a} \restriction B, C$ has even length and is therefore a trace of τ. So we have $u_{\leqslant a} \in D_\sigma \between D_\tau$ and clearly this witnesses a prefix of e in $D_{\sigma;\tau}$. For the other subcase, if $u \restriction A, B \in T_\sigma$ then $e \in D_{\sigma;\tau}$. Otherwise, applying $\sigma \leqslant^\sharp \sigma'$, we get some $sa \in D_\sigma$ such that $sa \sqsubseteq u \restriction A, B$. Once again, by lemma 4.2.1, we get $u_{\leqslant a} \in D_\sigma \between D_\tau$ witnessing some prefix of e in $D_{\sigma;\tau}$.

If e is witnessed by some $u_\infty \in T_{\sigma'} \between T_\tau$ then if $u_\infty \in T_\sigma \between T_\tau$ as well then we're done. Otherwise, there must be a $v \sqsubseteq^{\mathsf{fin}} u_\infty$ such that $v \restriction A, B \in T_{\sigma'}$ but $v \restriction A, B \notin T_\sigma$. Applying $\sigma \leqslant^\natural \sigma'$ we get a divergence in σ and the same old argument gives us some $d \in D_{\sigma;\tau}$ such that $d \sqsubseteq e$.

For the second part, suppose that $sab \in T_{\sigma';\tau}$ but $sab \notin T_{\sigma;\tau}$. This means that any witness $u \in \sigma' \parallel \tau$ for sab must satisfy $u \restriction A, B \notin T_\sigma$. So, pick any witness u for sab. Since $\sigma \leqslant^\natural \sigma'$, we have some $s'a' \in D_\sigma$ such that $s'a' \sqsubseteq (u \restriction A, B)$. Using the same sort of reasoning as above, we can deduce that $u_{\leqslant a'} \restriction B, C$ has even length and so $sab \sqsupseteq (u_{\leqslant a'} \restriction A, C) \in D_{\sigma;\tau}$. ■

Notice how the second half of $\leqslant^\sharp$,

$$(sab \in T_\tau \wedge sab \notin T_\sigma) \Rightarrow \exists d \in D_\sigma.\ d \sqsubseteq sab,$$

is used to establish *both* parts of the monotonicity of $\leqslant^\sharp$. This is how we catch the nasty counterexample mentioned above.

We now formalize the intuition that the minimal divergences of a strategy suffice to describe all of its possible divergent behaviour by showing that $=^\natural$-equivalent strategies always share the same set of minimal divergences.

Proposition 4.3.3 Given strategies σ and τ on arena A, we have $\sigma =^\natural_A \tau$ if, and only if, $T_\sigma = T_\tau$ and $\mathsf{div}(\sigma) = \mathsf{div}(\tau)$.

Proof Suppose that $T_\sigma = T_\tau$ and $\mathsf{div}(\sigma) = \mathsf{div}(\tau)$. Clearly $\sigma \leqslant^\flat \tau$ and $\tau \leqslant^\flat \sigma$. To check $\sigma \leqslant^\sharp \tau$, it suffices to check the first part (since $T_\sigma = T_\tau$). Let $e \in D_\tau$. There must be some $d \in \mathsf{div}(\tau) = \mathsf{div}(\sigma)$ such that $d \sqsubseteq e$. Hence $d \in D_\sigma$. We can check $\tau \leqslant^\sharp \sigma$ similarly.

Now suppose $\sigma =^\natural \tau$. This means that $T_\sigma \subseteq T_\tau$ and $T_\tau \subseteq T_\sigma$ and so $T_\sigma = T_\tau$. We now show that $\mathsf{div}(\sigma) \subseteq \mathsf{div}(\tau)$. The other inclusion can be proven similarly and the desired result follows. Let $d \in \mathsf{div}(\sigma)$. Since $\tau \leqslant^\sharp \sigma$, there must be some $e \in D_\tau$ such that $e \sqsubseteq d$. And since $\sigma \leqslant^\sharp \tau$, there must be some $d' \in D_\sigma$ such that $d' \sqsubseteq e$.

So $d' \sqsubseteq d$ and, by minimality of d, $d' = d$. Antisymmetry of $\sqsubseteq$ gives $d = e$, so $d \in D_\tau$. But is it minimal? Well, let $e' \in D_\tau$ such that $e' \sqsubseteq e$. We apply $\sigma \leqslant^\sharp \tau$ to get a $d'' \in D_\sigma$ such that $d'' \sqsubseteq e' \sqsubseteq e \sqsubseteq d$. Once again, $d'' = d$ and so $e' = e$. So $d = e \in \mathsf{div}(\tau)$. ■

This "minimal" representation of an $=^\natural$-equivalence class maps out exactly the *boundary* between the convergent and the divergent behaviour of the strategy in question. An alternative "maximal" representation could, as is done in CSP, consider the divergences to be extension-closed, *i.e.* for strategy σ on arena A, if $sa \in D_\sigma$ and $sabc \in L_A$ then $sabc \in D_\sigma$. (In fact, in the presence of the (**d2**) axiom, we'd also have to have $sab \in T_\sigma$.). This representation better reflects the idea that divergence is not something that can be "recovered from"; once a strategy can be divergent, it always remains so.

4.3.3 Continuity & algebraicity

We now investigate the continuity and algebraicity properties of our ordering on strategies. Much of what follows in this section holds independently of the finite-branching condition but, unsurprisingly, the continuity property of composition does crucially rely on this.

Directed lubs Suppose we have a directed set, Δ, of strategies for an arena A. We'd like to be able to define a least upper bound (lub) for Δ; but what should this look like? Well, first of all, it's clear that if a trace appears in *some* strategy in Δ, it should also appear in the traces of the lub. As for the divergences, if the lub of Δ can diverge at some point then this must surely mean that *every* strategy in Δ could have diverged too. These intuitions can be formalized with the following definition.

$$\bigsqcup \Delta =_{\mathrm{df}} \Big(\bigcup_{\sigma \in \Delta} T_\sigma, \{ sa \in L_A \mid s \in \bigcup_{\sigma \in \Delta} T_\sigma \wedge \forall \sigma \in \Delta. \exists d \in D_\sigma.\ d \sqsubseteq sa \} \Big).$$

Lemma 4.3.4 Given such a directed set Δ, $\bigsqcup \Delta$ is a well-defined strategy.

Proof It's easy to see that the traces satisfy (**t1**) and (**t2**). It's also clear from the definition that $\bigsqcup \Delta$ satisfies (**d2**). So we just need to check (**d1**). Suppose that $s \in T_{\bigsqcup \Delta}$, $sa \in L_A$ but $sa \notin \mathsf{dom}(\bigsqcup \Delta)$. Since $s \in T_{\bigsqcup \Delta}$, we can find some $\sigma \in \Delta$ such that $s \in T_\sigma$. So, consider any $\tau \in \Delta$. If $s \in T_\tau$ then, since $sa \notin \mathsf{dom}(\bigsqcup \Delta)$, $sa \notin \mathsf{dom}(\tau)$ and, by (**d2**), we have $d_\tau \in D_\tau$ such that $d_\tau \sqsubseteq sa$. If $s \notin T_\tau$ then, by directedness of Δ, there must be some $\upsilon \in \Delta$ such that $\sigma \leqslant^\natural \upsilon$ and $\tau \leqslant^\natural \upsilon$. So $s \in T_\upsilon$ and, with the same reasoning as above, there exists $d_\upsilon \in D_\upsilon$ such that $d_\upsilon \sqsubseteq sa$. Now, since $\tau \leqslant^\natural \upsilon$, there must be some $d_\tau \in D_\tau$ such that $d_\tau \sqsubseteq d_\upsilon \sqsubseteq sa$. In summary, for all $\tau \in \Delta$, there's some $d_\tau \in D_\tau$ such that $d_\tau \sqsubseteq sa$. Since $s \in T_{\bigsqcup \Delta}$ by hypothesis, $sa \in D_{\bigsqcup \Delta}$ and we're done. ■

So, our candidate lub is indeed a good strategy. Before we show that it's a least upper bound of Δ, we must first check that the definition works well with finite-branching strategies, *i.e.* those satisfying (**fb**).

Lemma 4.3.5 If each $\sigma \in \Delta$ is finite-branching then so is $\bigsqcup \Delta$.

Proof Suppose that $sa \in \mathsf{dom}(\bigsqcup \Delta)$ and that there are an infinite number of responses to this. Clearly $s \in T_{\bigsqcup \Delta}$. Consider $\sigma \in \Delta$. If $\mathsf{rng}_\sigma(sa)$ is infinite then, by (**fb**), we have some $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq sa$. Otherwise σ must be missing some sab. But since this $sab \in T_{\bigsqcup \Delta}$, it must have come from some other $\tau \in \Delta$ and, since Δ is directed, we have $\upsilon \in \Delta$ such that $\sigma \leqslant^\natural \upsilon$ and $\tau \leqslant^\natural \upsilon$. The latter of these gives $sab \in T_\upsilon$ and the former therefore requires some $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq sab$. So every $\sigma \in \Delta$ has a divergence $d_\sigma \sqsubseteq sa$ and we therefore have $sa \in D_{\bigsqcup \Delta}$. ■

Aside It's worth noting that, had we insisted that divergence sets be extension-closed, the definition of $\bigsqcup \Delta$ could have been considerably simplified: it would have sufficed to take the intersection of all the D_{σ_i}s. However, this would have caused complications elsewhere (particularly in the definition of composition) so we prefer to work with $=^\natural$-equivalence classes, safe in the knowledge that we have various concrete representations at hand, should we need them.

Least upper bounds & continuity We're under an obligation to show that $\bigsqcup \Delta$ is a least upper bound for Δ. In general, it won't be *the* least upper bound, but obviously lubs are unique up to $=^\natural$-equivalence.

Proposition 4.3.6 The strategy $\bigsqcup \Delta$ is a least upper bound of Δ.

Proof We first show that it's an upper bound. Obviously, for any $\sigma \in \Delta$, we have $\sigma \leqslant^\flat \bigsqcup \Delta$. So we just have to check $\sigma \leqslant^\sharp \bigsqcup \Delta$. If $d \in D_{\bigsqcup \Delta}$ then, by definition, we have some $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq d$ for all $\sigma \in \Delta$. So the first half of $\leqslant^\sharp$ is okay. For the second half, suppose we have $sab \in T_{\bigsqcup \Delta}$ but, for some $\sigma \in \Delta$, $sab \notin T_\sigma$. Since $sab \in T_{\bigsqcup \Delta}$, there must be some other $\tau \in \Delta$ such that $sab \in T_\tau$. Once again, we apply directedness of Δ to find $\upsilon \in \Delta$ such that $\sigma \leqslant^\natural \upsilon$ and $\tau \leqslant^\natural \upsilon$. We therefore have $sab \in T_\upsilon$ and we can deduce that there's some $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq sab$. Therefore $\bigsqcup \Delta$ is an upper bound for Δ.

Let δ be any upper bound of Δ. If $s \in T_{\bigsqcup \Delta}$ then, for some $\sigma \in \Delta$, we have $s \in T_\sigma$. Since $\sigma \leqslant^\flat \delta$, we also have $s \in T_\delta$ so $\bigsqcup \Delta \leqslant^\flat \delta$. To see that $\bigsqcup \Delta \leqslant^\sharp \delta$, consider $e \in D_\delta$. As δ is an upper bound for Δ, every strategy $\sigma \in \Delta$ has a divergence $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq e$. Consider the maximal (*i.e.* longest) such divergence; call it d. Clearly $d \sqsubseteq e$ and, by definition, $d \in D_{\bigsqcup \Delta}$. Finally, suppose that $sab \in T_\delta$ but $sab \notin T_{\bigsqcup \Delta}$. This means that for no $\sigma \in \Delta$ do we have $sab \in T_\sigma$. So, for all $\sigma \in \Delta$, we have $d_\sigma \in D_\sigma$ such that $d_\sigma \sqsubseteq sab$. Again, consider the maximal such divergence; it's in $D_{\bigsqcup \Delta}$ by definition and clearly is a prefix of sab. ■

We already know that, for any arena A, the set $\mathsf{Str}(A)$ of $=^\natural$-equivalence classes of strategies for A is partially ordered by $\leqslant^\natural$. The above lemma says that all directed sets of $\mathsf{Str}(A)$ have lubs and it's clear from the definition that "equivalent" directed sets have equivalent lubs; *i.e.* $\mathsf{Str}(A)$ is a CPO. We also have the following continuity property, making essential use of the finite-branching axiom.

Proposition 4.3.7 Let Δ be a directed collection of finite-branching strategies for $B \Rightarrow C$ and let $\sigma : A \Rightarrow B$ and $\upsilon : C \Rightarrow D$ be finite-branching strategies. Then $\sigma ; (\bigsqcup \Delta) =^\natural \bigsqcup(\sigma ; \Delta)$ and $(\bigsqcup \Delta) ; \upsilon =^\natural \bigsqcup(\Delta ; \upsilon)$.

Proof We consider only the first equivalence; the other is just its mirror image. Since composition is monotone with respect to $\leqslant^\natural$, we have $\bigsqcup(\sigma ; \Delta) \leqslant^\natural \sigma ; \bigsqcup \Delta$. It's easy to see that $\sigma ; (\bigsqcup \Delta) =^\flat \bigsqcup(\sigma ; \Delta)$ so it suffices to check that, for all $d \in D_{\bigsqcup(\sigma;\Delta)}$ we have some $d' \in D_{\sigma;(\bigsqcup \Delta)}$ such that $d' \sqsubseteq d$. If $d \in D_{\bigsqcup(\sigma;\Delta)}$ then, for all $\tau \in \Delta$, we have some $d_\tau \in D_{\sigma;\tau}$ such that $d_\tau \sqsubseteq d$. If one of these d_τs arises from an infinite chatter between σ and the appropriate τ, the same infinite chatter witnesses $d \in D_{\sigma;\bigsqcup \Delta}$. Similarly, if some d_τ is witnessed by a divergence of σ and a trace of τ, this same witness occurs in $\sigma ; \bigsqcup \Delta$. We're left with the case where *every* d_τ arises from a trace of σ and a divergence of τ. Let W be the prefix-closure of the set $\{u \in \mathsf{int}(A, B, C) \mid u \restriction A, B \in T_\sigma \wedge \exists \tau \in \Delta.\ u \restriction B, C \in D_\tau \wedge u \restriction A, C \sqsubseteq d\}$. This W is a tree with finite depth. If it's infinite, we apply König's lemma to find a point of infinite branching in some $\tau \in \Delta$ so that, for some $u \in W$, $u \in D_\sigma \between D_{\bigsqcup \Delta}$ witnesses a prefix of d. If W is finite, we might have some $u \in W$ such that $u \restriction B, C$ has a divergent prefix in every $\tau \in \Delta$. In this case, $u \restriction B, C \in D_{\bigsqcup \Delta}$, witnessing some $d' \in D_{\sigma;(\bigsqcup \Delta)}$ such that $d' \sqsubseteq d$. If there is no such $u \in W$ then, for any $v \in W$, we can find $\tau_v \in \Delta$ such that no prefix of $v \restriction B, C$ is in D_{τ_v}. Since W is finite, the set $\{\tau_v \mid v \in W\}$ is finite and hence has an upper bound in Δ; call it τ. But then, for all $u \in W$, no prefix of $u \restriction B, C$ is in D_τ so we have no prefix of d in $D_{\sigma;\tau}$ and hence no prefix of d in $D_{\bigsqcup(\sigma;\Delta)}$—which is a contradiction. ■

Algebraicity In the previous chapter, we saw a very simple characterization of the compact strategies in the subset ordering: they were those of finite cardinality. At a first glance, we might imagine that, in the setting with divergences, a strategy is compact iff its trace set and its divergence set both have finite cardinality. However, this is far too strong a condition. Consider a strategy on $\mathbf{N} \Rightarrow \mathbf{N}$ with trace set $\{\varepsilon, \mathsf{qq}, \mathsf{qq00}\}$. Intuitively, this strategy ought to be compact but, according to axiom (**d1**), it's obliged to have an infinite set of divergences.

On second thoughts, what we want is that the strategy has a finite set of *interesting* divergences. Formally, we say that $sa \in D_\sigma$ is ***interesting*** iff $sa \in \mathsf{dom}(\sigma)$; otherwise it's *uninteresting*. Clearly, any strategy with a finite trace set can have only a finite number of interesting divergences. Another useful way to look at this is in terms of the *reliable points* of a strategy: we say that sa is a ***reliable point*** of σ, written $sa \in \mathsf{rp}(\sigma)$, iff $s \in T_\sigma$ and, for all $d \sqsubseteq^{\mathsf{odd}} sa$, $d \notin D_\sigma$. In other words, sa isn't in the extension-closure of σ's divergences.

Lemma 4.3.8 For strategies σ and τ such that $\sigma =^\flat \tau$, $\sigma \leqslant^\natural_A \tau$ if, and only if, $sa \in \mathsf{rp}(\sigma) \Rightarrow sa \in \mathsf{rp}(\tau)$.

Proposition 4.3.9 A finite-branching strategy σ on an arena A is compact if, and only if, its trace set T_σ is finite.

Proof We prove the "if" direction first. Let Δ be a directed set of strategies for A and suppose that $\sigma \leqslant^\natural \bigsqcup\Delta$. Therefore, for any $s \in T_\sigma$, also $s \in T_{\bigsqcup\Delta}$ from which we deduce the existence of some $\sigma_s \in \Delta$ such that $s \in T_{\sigma_s}$. Now, if $sa \in \mathsf{rp}(\sigma)$ then, since $\sigma \leqslant^\sharp \bigsqcup\Delta$, $sa \notin D_{\bigsqcup\Delta}$ from which we deduce the existence of some $\sigma_{sa} \in \Delta$ such that $sa \in \mathsf{rp}(\sigma_{sa})$. Since T_σ is finite, we have a finite subset $\Delta' \subseteq^{\mathsf{fin}} \Delta$ consisting of all the σ_ss and all the σ_{sa}s. By directedness of Δ, Δ' has an upper bound (call it τ) in Δ. Clearly $\sigma \leqslant^\flat \tau$. If $sab \in T_\tau$ but $sab \notin T_\sigma$, we have $sab \in T_{\bigsqcup\Delta}$ so, applying $\sigma \leqslant^\natural \bigsqcup\Delta$, we find some $d \in D_\sigma$ such that $d \sqsubseteq sa$. If $sa \in D_\tau$ then, since τ is an upper bound for Δ', we have $sa \notin \mathsf{rp}(\sigma)$. If $s \in T_\sigma$ then, by definition, some prefix of sa is in D_σ. If $s \notin T_\sigma$ then, since $s \in T_\tau$, we have $s \in T_{\bigsqcup\Delta}$ and, since $\sigma \leqslant^\sharp \bigsqcup\Delta$, the above argument provides a prefix of sa in D_σ.

For the "only if" direction, consider the set Δ_σ of all strategies σ' with finite trace set such that $\sigma' \leqslant^\natural \sigma$. Given σ_1 and σ_2 in Δ_σ, we define σ_3 by: $T_{\sigma_3} = T_{\sigma_1} \cup T_{\sigma_2}$ and $D_{\sigma_3} = \{sa \in L_A^{\mathsf{odd}} \mid s \in T_{\sigma_3} \wedge \exists d_1 \in D_{\sigma_1}.\, d_1 \sqsubseteq sa \wedge \exists d_2 \in D_{\sigma_2}.\, d_2 \sqsubseteq sa\}$. It's clear that σ_3 is an upper bound for σ_1 and σ_2 and that it has a finite trace set. It's also not too hard to check that $\sigma_3 \leqslant^\natural \sigma$, so the set Δ_σ is directed and $\bigsqcup\Delta_\sigma \leqslant^\natural \sigma$. We wish to establish the converse, *i.e.* $\sigma \leqslant^\natural \bigsqcup\Delta_\sigma$.

First of all, suppose that $sa \in \mathsf{rp}(\sigma)$. Since σ is finite-branching, for all $s'a' \sqsubseteq^{\mathsf{odd}} sa$, $\mathsf{rng}_\sigma(s'a')$ is finite. We build a strategy σ_{sa} where

$$T_{\sigma_{sa}} = \{\varepsilon\} \cup \{s'a'b' \in \mathsf{rng}_\sigma(s'a') \mid s'a' \in \mathsf{dom}(\sigma) \wedge s'a' \sqsubseteq sa\}.$$

This is clearly a finite trace set. We specify the divergences of σ_{sa} to be precisely those legal plays that are an immediate legal extension tb of some $t \in T_{\sigma_{sa}}$ such that $tb \not\sqsubseteq sa$, *i.e.* they're all uninteresting. In order for σ_{sa} to be in Δ_σ, we need $\sigma_{sa} \leqslant^\natural \sigma$. Clearly $\sigma_{sa} \leqslant^\flat \sigma$. So, suppose that $d \in D_\sigma$. Let s' be the longest common prefix of d and sa. If s' has even length, we write d as $s'a's''$ and, by definition, $s'a' \in D_{\sigma_{sa}}$. If s' has odd length, we can write d as $s'a'b's''$ and, by construction, $s'a'b' \in D_{\sigma_{sa}}$. The other half of $\leqslant^\sharp$ is very similar so we have $\sigma_{sa} \in \Delta_\sigma$.

The other case is where $sa \notin \mathsf{rp}(\sigma)$. Let $s'a'$ be the longest prefix of sa such that $s'a' \in \mathsf{rp}(\sigma)$ and d_{sa} be the shortest unreliable prefix of sa, *i.e.* $d_{sa} = s'a'mn$ for the appropriate moves m and n. We firstly build up a finite trace set $T_{s'a'}$ exactly as above. Now, for each $sab \in \mathsf{rng}_\sigma(sa)$, we define a strategy σ_{sab} with trace set

$$T_{\sigma_{s'a'}} \cup \{s' \in L_A^{\mathsf{even}} \mid s' \sqsubseteq sab\}$$

and the single interesting divergence d_{sa}. Very similar reasoning as above shows that $\sigma_{sab} \in \Delta_\sigma$.

So, for any $s \in T_\sigma$, we can find a strategy in Δ_σ containing it; so $\sigma \leqslant^\flat \bigsqcup \Delta_\sigma$. To establish $\sigma \leqslant^\sharp \bigsqcup \Delta_\sigma$ it suffices, by lemma 4.3.8, to show that $\mathsf{rp}(\sigma) \subseteq \mathsf{rp}(\bigsqcup \Delta_\sigma)$. If $sa \in \mathsf{rp}(\sigma)$, we have $\sigma_{sa} \in \Delta_\sigma$ as defined above. By the definition of $\bigsqcup \Delta_\sigma$, $sa \notin D_{\bigsqcup \Delta_\sigma}$, *i.e.* $sa \in \mathsf{rp}(\bigsqcup \Delta_\sigma)$ as required. Therefore, any finite-branching strategy σ is $=^\natural$-equivalent to the lub of a directed set of strategies with finite trace sets; hence, in the CPO of $=^\natural$-equivalence classes of finite-branching strategies for A, no strategy with an infinite trace set can be compact. ■

Aside The above construction of the strategies σ_{sa} may seem overly complicated. Unfortunately, this complexity is necessary; we can illustrate this with a small example. Consider the arena $(\mathbf{B} \Rightarrow \mathbf{C}) \Rightarrow \mathbf{C}$ and the strategy σ with trace set $\{\varepsilon, \mathsf{qq}, \mathsf{qqqtt}, \mathsf{qqqff}, \mathsf{qqqttaa}, \mathsf{qqqffaa}\}$ and a single interesting divergence qqqffa. We wish to construct a strategy σ' containing the trace $\mathsf{qqqttaa}$ which approximates σ. But the presence of the divergence qqqffa in σ forces us to include qqq as a (rather unnatural) divergence of σ'. This problem of "phantom divergences" is caused by Player's branching (in this case, the choice between tt and ff) and, as such, can never arise in a deterministic strategy. The solution exploits the finite-branching property of strategies by "covering" all of Player's possible branching decisions while σ's behaviour is still reliable. Once σ becomes unreliable, we can ignore any subsequent branching options for Player since we already have a divergence.

4.4 A category of arenas & strategies

4.4.1 Composition is "associative"

So far, we've introduced strategies and their composition, shown that this is well-defined and has identities. We've also defined an ordering on strategies, inducing a notion of equivalence $=^\natural$. The next step in our development is to establish an associativity property of composition. With that in place, we can then build a (symmetric monoidal closed) category of arenas and strategies.

Infinite zipping... First of all, we extend the notion of an infinite interaction to four arenas. Formally, given an infinite string u over the alphabet $M_A+M_B+M_C+M_D$ equipped with "justification pointers" in such a way that all finite prefixes u' of u satisfy $u' \upharpoonright A, B \in J_{A\Rightarrow B}$, $u' \upharpoonright B, C \in J_{B\Rightarrow C}$ and $u' \upharpoonright C, D \in J_{C\Rightarrow D}$ then u is an ***infinite interaction of four arenas*** iff $u \upharpoonright A, D \in L_{A\Rightarrow D}$ and, for all $i \in \mathbf{N}$, $u_{<i} \in \mathsf{int}(A, B, C, D)$. We write $\mathsf{int}_\infty(A, B, C, D)$ for the set of all such u. Note that u can have an infinite tail of moves from M_B, from M_C or from $M_B + M_C$.

In order to prove an associativity result for strategies with divergences, we need to extend the zipping lemma from the previous chapter. Firstly, a useful bit of notation when dealing with infinite strings: if u is a (finite or) infinite string and

a is an occurrence of a symbol in u, we write $u_{>a}$ (respectively $u_{\geqslant a}$) for the suffix of u obtained by discarding the prefix $u_{\leqslant a}$ (respectively $u_{<a}$).

Lemma 4.4.1 If $u \in \mathsf{int}(A, C, D)$ and $v_\infty \in \mathsf{int}_\infty(A, B, C)$ such that $u \restriction A, C = v_\infty \restriction A, C$ then there's a unique $w_\infty \in \mathsf{int}_\infty(A, B, C, D)$ such that $w_\infty \restriction A, C, D = u$ and $w_\infty \restriction A, B, C = v_\infty$.

Proof Since $v_\infty \in \mathsf{int}_\infty(A, B, C)$, it must have an infinite tail of moves from B. Consider the last move (from either A or C) before the infinite tail gets going; call it m. By definition, $(v_\infty)_{\leqslant m} \in \mathsf{int}(A, B, C)$. We apply the zipping lemma 3.2.3 to u and $(v_\infty)_{\leqslant m}$ obtaining a unique $(w_\infty)_{\leqslant m} \in \mathsf{int}(A, B, C, D)$ such that $(w_\infty)_{\leqslant m} \restriction A, B, C = (v_\infty)_{\leqslant m}$ and $(w_\infty)_{\leqslant m} \restriction A, C, D = u$. Let w_∞ be $(w_\infty)_{\leqslant m} \cdot (v_\infty)_{>m}$. It's not too hard to see that $w_\infty \in \mathsf{int}_\infty(A, B, C, D)$ and it's clearly unique in satisfying the required conditions. ■

We are now in a position to prove that composition of finite-branching strategies is morally associative, *i.e.* associative up to $=^\natural$.

Proposition 4.4.2 If $\sigma : A \Rightarrow B$, $\tau : B \Rightarrow C$ and $\upsilon : C \Rightarrow D$ are finite-branching strategies then $(\sigma \,;\, \tau) \,;\, \upsilon =^\natural \sigma \,;\, (\tau \,;\, \upsilon)$.

Proof By proposition 3.2.4, we know that $T_{(\sigma;\tau);\upsilon} = T_{\sigma;(\tau;\upsilon)}$. Therefore it suffices to show that, for any d in the divergences of one association, there exists an e in the divergences of the other association such that $e \sqsubseteq d$. We show this for the left-to-right direction; the converse follows similarly.

So, if $d \in D_{(\sigma;\tau);\upsilon}$ is witnessed by $u \in D_{\sigma;\tau} \between D_\upsilon$ then there are two cases: *either* $u \restriction A, C \in T_{\sigma;\tau}$ and $u \restriction C, D \in D_\upsilon$ *or* $u \restriction A, C \in D_{\sigma;\tau}$ and $u \restriction C, D \in T_\upsilon$. In the first case, we have some $v \in T_\sigma \parallel T_\tau$ such that $u \restriction A, C = v \restriction A, C$. We zip u and v together yielding some $w \in \mathsf{int}(A, B, C, D)$ such that $w \restriction A, B \in T_\sigma$, $w \restriction B, C \in T_\tau$ and $w \restriction C, D \in D_\upsilon$. The usual state diagram argument shows that $w \restriction B, D$ is legal, so $w \restriction A, B, D \in D_\sigma \between D_{\tau;\upsilon}$ and $d = w \restriction A, D \in D_{\sigma;(\tau;\upsilon)}$. The second case subdivides into two further cases: $u \restriction A, C$ is either a finitely or an infinitely generated divergence. In the former case, the argument is much the same as the above. In the latter case, $u \restriction A, C$ is witnessed by some $v_\infty \in T_\sigma \between T_\tau$. We apply the above lemma to u and v_∞ to build $w_\infty \in \mathsf{int}_\infty(A, B, C, D)$, with an infinite tail in B, such that $w_\infty \restriction A, C, D = u$ and $w_\infty \restriction A, B, C = v_\infty$. All finite prefixes of $w_\infty \restriction A, B$ are in $\mathsf{cc}(\sigma)$ and all finite prefixes of $w_\infty \restriction B, D$ are in $\mathsf{cc}(\tau \,;\, \upsilon)$ so $w_\infty \restriction A, B, D \in T_\sigma \between T_{\tau;\upsilon}$ which witnesses $d = w_\infty \restriction A, D \in D_{\sigma;(\tau;\upsilon)}$.

The other case is when $d \in D_{(\sigma;\tau);\upsilon}$ is witnessed by some $u_\infty \in T_{\sigma;\tau} \between T_\upsilon$ so that we have an infinite chatter in C between $\sigma \,;\, \tau$ and υ. Consider the set W, defined as

$$\{\, w \in \mathsf{int}(A, B, C) \mid w \restriction A, B \in \mathsf{cc}(\sigma) \wedge w \restriction B, C \in \mathsf{cc}(\tau) \wedge w \restriction A, C \sqsubseteq u_\infty \restriction A, C \,\},$$

of "legal interactions of σ and τ heading towards $u_\infty \restriction A, C$". This is prefix-closed and hence can be considered as an infinite tree.

If it's an infinitely branching tree, there must be some $v \in W$ to which one or other of σ and τ has an infinite number of possible responses. Let X be the component where v has odd length. By (**fb**) in X, there is some $sa \in D_X$ such that $sa \sqsubseteq v \upharpoonright X$. Since W is prefix-closed, $v_{\leqslant a} \in W$ and $(v_{\leqslant a}) \upharpoonright X = v \upharpoonright a_{\leqslant X} = sa \in D_X$ so that $v_{\leqslant a} \in D_\sigma \between D_\tau$. We want to zip this up with a long enough prefix of u_∞ to get some $w \in \mathsf{int}(A, B, C, D)$ such that $w \upharpoonright C, D \in T_\upsilon$ and $w \upharpoonright A, B, C \in D_\sigma \between D_\tau$. If a is a move of A or C, it suffices to consider $(u_\infty)_{\leqslant a}$. If $a \in M_B$, we have to consider $(u_\infty)_{\leqslant a'}$ where a' is the last occurrence of A or C in $v_{\leqslant a}$. So we obtain our w by applying zipping to $v_{\leqslant a}$ and the appropriate prefix of u_∞. We can hide C as usual, yielding $w \upharpoonright A, B, D \in D_\sigma \between D_{\tau;\upsilon}$, witnessing some $d' \in D_{\sigma;(\tau;\upsilon)}$ such that $d' \sqsubseteq d$.

Otherwise, W is finitely branching so that, by König's lemma, it has an infinite path. In other words, we have an infinite, increasing sequence of legal interactions in $\mathsf{int}(A, B, C)$. We apply lemma 4.4.1 to each of these with u_∞, yielding an infinite, increasing sequence of legal interactions in $\mathsf{int}(A, B, C, D)$. This corresponds to some $w_\infty \in \mathsf{int}_\infty(A, B, C, D)$ satisfying, for all $w' \sqsubseteq^{\mathsf{fin}} w_\infty$, $w' \upharpoonright A, B \in \mathsf{cc}(\sigma)$, $w' \upharpoonright B, C \in \mathsf{cc}(\tau)$ and $w' \upharpoonright C, D \in \mathsf{cc}(\upsilon)$. This w_∞ has an infinite tail in B and/or C. If $w_\infty \upharpoonright A, B, D$ is finite then the infinite tail is solely in C and so $w_\infty \upharpoonright B, C, D \in T_\tau \between T_\upsilon$. This means that $w_\infty \upharpoonright A, B, D \in D_\sigma \between D_{\tau;\upsilon}$ and $d = w_\infty \upharpoonright A, D \in D_{\sigma;(\tau;\upsilon)}$. If $w \upharpoonright A, B, D$ is itself an infinite interaction then it's in $T_\sigma \between T_{\tau;\upsilon}$, witnessing an infinite chatter between σ and $\tau\,;\upsilon$ and again, $d = w_\infty \upharpoonright A, D \in D_{\sigma;(\tau;\upsilon)}$. ■

So we don't have associativity "on the nose" but only upto $=^\natural$-equivalence. As we remarked previously, it seems entirely natural to consider strategies upto this notion of equivalence so we proceed by defining a category with objects as arenas and arrows as $=^\natural$-equivalence classes of strategies.

Aside Notice that there was a crucial step in the proof of associativity where we made use of the *finite-branching* condition on strategies. Composition isn't associative in general, as the following example illustrates.

Let $\sigma : \mathbf{1} \Rightarrow \mathbf{C}$ be the strategy for `skip` with no divergences; let $\tau : \mathbf{C} \Rightarrow \mathbf{C} \times \mathbf{N}$ be the "pairing" of $\mathsf{id}_\mathbf{C}$ and the strategy (manifestly violating finite-branching) for 'any positive integer' with no divergences; and let $\upsilon : \mathbf{C} \times \mathbf{N} \Rightarrow \mathbf{C}$ be the obvious strategy interpreting the following "term".

$$
\begin{array}{rcl}
\mathtt{c} : \mathtt{Com}, n : \mathtt{Nat} & \vdash & \mathtt{case}\ n\ \mathtt{of} \\
& & \quad 0 \Rightarrow \mathtt{skip} \\
& & \quad 1 \Rightarrow \mathtt{c} \\
& & \quad 2 \Rightarrow \mathtt{c}\,;\mathtt{c} \\
& & \qquad \vdots
\end{array}
$$

So υ first of all asks for Opponent to think of a number and then runs its command argument that many times.

In $(\sigma;\tau);\upsilon$, interactions take the following general form with no divergence in the resulting strategy.

$$
\begin{array}{ccccccccccc}
\mathbf{1} & \Rightarrow & \mathbf{C} & \times & \mathbf{N} & ; & \mathbf{C} & \times & \mathbf{N} & \Rightarrow & \mathbf{C} \\
 & & & & & & & & & & \mathsf{q} \\
 & & & & & & & & \mathsf{q} & & \\
 & & & & \mathsf{q} & & & & & & \\
 & & & & \mathsf{n} & & & & & & \\
 & & & & & & & & \mathsf{n} & & \\
 & & & & & & \mathsf{q} & & & & \\
 & & \mathsf{q} & & & & & & & & \\
 & & \mathsf{a} & & & & & & & & \\
 & & & & & & \mathsf{a} & & & & \\
 & & & & & \vdots & & & & & \\
 & & & & & & \mathsf{q} & & & & \\
 & & \mathsf{q} & & & & & & & & \\
 & & \mathsf{a} & & & & & & & & \\
 & & & & & & \mathsf{a} & & & & \\
 & & & & & & & & & & \mathsf{a}
\end{array}
$$

In $\sigma\,;(\tau\,;\upsilon)$, interactions take the following general forms...

$$
\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{C} & ; & \mathbf{C} & \Rightarrow & \mathbf{C} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
 & & & & \mathsf{a} & & \\
 & & & & & & \mathsf{a} \\
 & & & & & & \\
 & & & & & & \\
 & & & & & & \\
 & & & & & &
\end{array}
\qquad
\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{C} & ; & \mathbf{C} & \Rightarrow & \mathbf{C} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
 & & & & \mathsf{a} & & \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
 & & & & \mathsf{a} & & \\
 & & & & & & \mathsf{q}
\end{array}
\qquad \ldots
$$

...and so on, with the result that we get a divergence by infinite chattering in this association. Roughly speaking, one way to look at this is that the $(\sigma\,;\tau)\,;\upsilon$ association is `skip` with any *finite* delay whereas the other assocation is `skip` with a potentially *infinite* delay.

This isn't a problem if we restrict attention to finite-branching strategies; roughly speaking, any strategy which can trick one association into diverging will itself necessarily make use of infinite branching, so there will be a divergence anyway. This is precisely the point where finite-branching was used in the associativity proof. This difficulty is not peculiar to game semantics: precisely the same thing arises in CSP, manifesting itself in problems with the "hiding" combinator; see [87] for a good discussion of this point.

4.4.2 An SMCC of arenas & strategies

The basic category We now have enough in place to build a basic category **G** of arenas and strategies with divergences. An arrow $f : A \to B$ is a $=^{\natural}$-equivalence class of finite-branching strategies for $A \Rightarrow B$. Given two such arrows, $f : A \to B$ and $g : B \to C$, induced by equivalence classes $\langle\sigma_f\rangle$ and $\langle\tau_g\rangle$ respectively, we define their composite $f\,;g$ to be $\langle\sigma_f\,;\tau_g\rangle$.

This composition is well-defined by proposition 4.2.2 (composition of strategies is well-defined), proposition 4.2.5 (the finite-branching condition is robust) and proposition 4.3.2 (composition is monotone with respect to $\leqslant^{\natural}$). The identity for arena A is $\langle \mathsf{Id}_A\rangle$ (lemma 4.2.3) and associativity of composition is guaranteed by proposition 4.4.2.

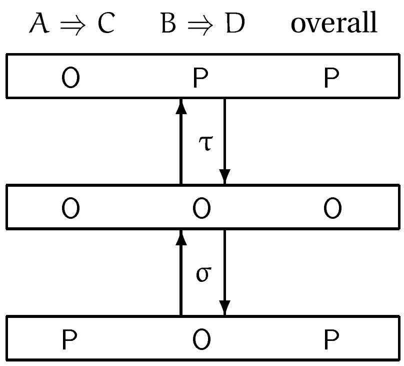
Product & arrow as bifunctors We'd like to extend the $\times$ and $\Rightarrow$ operations on arenas to strategies. To this end, suppose we have two strategies $\sigma : A \Rightarrow C$ and $\tau : B \Rightarrow D$. It seems reasonable to define the traces of $\sigma \times \tau$ by:

$$T_{\sigma\times\tau} = \{s \in L_{(A\times B)\Rightarrow(C\times D)} \mid s\restriction A, C \in T_\sigma \wedge s\restriction B, D \in T_\tau\}.$$

When should $\sigma \times \tau$ have a divergence? Well, since only one or other of σ and τ can be "active" at any given time, we'd expect a divergence to be provoked by the action of just one strategy. So, if we have a legal play of $(A \times B) \Rightarrow (C \times D)$ which, for example, restricts to be a trace of σ and a divergence of τ then we'd expect that play to be a divergence of $\sigma \times \tau$.

$$\begin{array}{rcl} D_{\sigma\times\tau} & = & \{s \in L_{(A\times B)\Rightarrow(C\times D)} \mid s\restriction A, C \in T_\sigma \wedge s\restriction B, D \in D_\tau\} \\ & \cup & \{s \in L_{(A\times B)\Rightarrow(C\times D)} \mid s\restriction A, C \in D_\sigma \wedge s\restriction B, D \in T_\tau\}. \end{array}$$

Recall the state diagram for $\sigma \times \tau$ from chapter 3.



Observe that a legal play can be a trace of $\sigma \times \tau$ if, and only if, it's in the middle state (OOO) and it can be a divergence of $\sigma \times \tau$ if, and only if, it's in either the top (OPP) or the bottom (POP) state. From this, it's easy to see that $\sigma \times \tau$ is a well-defined strategy. A similar definition can be made for $\sigma \Rightarrow \tau$.

Consider the strategies $\sigma : A \Rightarrow B$, $\tau : B \Rightarrow C$, $\sigma' : A' \Rightarrow B'$ and $\tau' : B' \Rightarrow C'$. We already know that $T_{(\sigma\times\sigma');(\tau\times\tau')} = T_{(\sigma;\tau)\times(\sigma';\tau')}$. Furthermore, it's clear that $\mathsf{Id}_A \times \mathsf{Id}_B = \mathsf{Id}_{A\times B}$ since, for any arena A, there are no divergences in Id_A. Before we can show that $D_{(\sigma;\tau)\times(\sigma';\tau')} = D_{(\sigma\times\sigma');(\tau\times\tau')}$, we need to rework lemma 3.3.2 in the same way that lemma 4.4.1 is a reworking of the original zipping lemma 3.2.3.

Lemma 4.4.3 If $s \in L_{(A\times A')\Rightarrow(C\times C')}$, $u_\infty \in \mathsf{int}_\infty(A, B, C)$ and $u' \in \mathsf{int}(A', B', C')$ in such a way that $s \restriction A, C = u \restriction A, C$ and $s \restriction A', C' = u' \restriction A', C'$ then there exists a unique $v_\infty \in \mathsf{int}_\infty(A \times A', B \times B', C \times C')$ such that $v_\infty \restriction A, B, C = u_\infty$, $v_\infty \restriction A', B', C' = u'$ and $v_\infty \restriction A, A', C, C' = s$.

Proof We proceed analogously to lemma 4.4.1. ∎

Lemma 4.4.4 If we have $\sigma : A \Rightarrow B$, $\sigma' : A' \Rightarrow B'$, $\tau : B \Rightarrow C$ and $\tau' : B' \Rightarrow C'$ then $D_{(\sigma;\tau)\times(\sigma';\tau')} = D_{(\sigma\times\sigma');(\tau\times\tau')}$.

Proof To avoid excessive subscripts, we abbreviate $(\sigma\,;\tau) \times (\sigma'\,;\tau')$ by LHS and $(\sigma \times \sigma')\,;(\tau \times \tau')$ by RHS.

If $d \in D_{LHS}$ then, WLOG, we have $d \restriction A, C \in D_{\sigma;\tau}$ and $d \restriction A', C' \in T_{\sigma';\tau'}$. The latter is witnessed by some $u' \in T_{\sigma'} \parallel T_{\tau'}$. The argument splits into two cases, according to whether $d \restriction A, C$ is witnessed by $u \in D_\sigma \lightning D_\tau$ or by $u_\infty \in T_\sigma \lightning T_\tau$. If $u \in D_\sigma \lightning D_\tau$ then, WLOG, assume that $u \restriction A, B \in D_\sigma$ and $u \restriction B, C \in T_\tau$. We now apply the zipping lemma 3.3.2 to d, u and u' yielding some $v \in \mathsf{int}(A \times A', B \times B', C \times C')$ such that $v \restriction A, B = u \restriction A, B \in D_\sigma$, $v \restriction B, C = u \restriction B, C \in T_\tau$, $v \restriction A', B' = u' \restriction A', B' \in T_{\sigma'}$ and $v \restriction B', C' = u' \restriction B', C' \in T_{\tau'}$. By definition, $v \restriction A, A', B, B' \in D_{\sigma\times\sigma'}$ and $v \restriction B, B', C, C' \in T_{\tau\times\tau'}$ and so $d = v \restriction A, A', C, C' \in D_{RHS}$. If d is witnessed by some $u_\infty \in T_\sigma \lightning T_\tau$ then we apply the infinite zipping lemma 4.4.3 to d, u_∞ and u' yielding some $v_\infty \in \mathsf{int}_\infty(A \times A', B \times B', C \times C')$. Let v' be any finite prefix of v_∞. We have $v' \restriction A, A', B, B' \in \mathsf{cc}(\sigma \times \sigma')$ and $v' \restriction B, B', C, C' \in \mathsf{cc}(\tau \times \tau')$. Therefore $v_\infty \in (\sigma \times \sigma') \lightning (\tau \times \tau')$ and so $d = v_\infty \restriction A, A', C, C' \in D_{RHS}$.

If $d \in D_{RHS}$ then once again the argument splits into two cases according to whether d is finitely or infinitely generated. In the former case, WLOG suppose that d is witnessed by some $u \in \mathsf{int}(A \times A', B \times B', C \times C')$ where $u \upharpoonleft C, C' \in D_{\sigma\times\sigma'}$ and $u \upharpoonleft A, A' \in T_{\tau\times\tau'}$. Again, WLOG suppose that $u \restriction A, B \in T_\sigma$ and $u \restriction A', B' \in D_{\sigma'}$. By lemma 3.3.1, $u \restriction A, B, C \in T_\sigma \parallel T_\tau$ and $u \restriction A', B', C' \in D_{\sigma'} \lightning D_{\tau'}$ so we have that $u \restriction A, C \in T_{\sigma;\tau}$ and $u \restriction A', C' \in D_{\sigma';\tau'}$ and $d = u \restriction A, A', C, C' \in D_{LHS}$. The final case is if d is infinitely generated. In this situation, we have some $u_\infty \in \mathsf{int}_\infty(A \times A', B \times B', C \times C')$ such that, WLOG, $u_\infty \restriction A, B, C \in T_\sigma \lightning T_\tau$ and $u_\infty \restriction A', B', C' \in T_{\sigma'} \parallel T_{\tau'}$. But then $u_\infty \restriction A, C \in D_{\sigma;\tau}$ and $u_\infty \restriction A', C' \in T_{\sigma';\tau'}$ so $d = u_\infty \restriction A, A', C, C' \in D_{LHS}$. ∎

We now know that $\times$ is a "bifunctor" at the level of strategies. The final step is to show that $\times$ is "monotone" with respect to the $=^\natural$ equivalence.

Lemma 4.4.5 If $\sigma, \sigma' : A \Rightarrow C$ and $\tau, \tau' : B \Rightarrow D$ are such that $\sigma =^\natural \sigma'$ and $\tau =^\natural \tau'$ then $(\sigma \times \tau) =^\natural (\sigma' \times \tau')$.

Proof It's easy to see that $(\sigma \times \tau) =^\flat (\sigma' \times \tau')$ and, since $\mathsf{div}(\sigma) = \mathsf{div}(\sigma')$ and $\mathsf{div}(\tau) = \mathsf{div}(\tau')$, $\mathsf{div}(\sigma \times \tau) = \mathsf{div}(\sigma' \times \tau')$ and we're done. ■

With this in place, we define, for $\sigma : A \Rightarrow C$ and $\tau : B \Rightarrow D$, $\langle\sigma\rangle \times \langle\tau\rangle =_{\mathrm{df}} \langle\sigma \times \tau\rangle$. It's easy to show, using the above lemmas, that this defines a bifunctor on **G**.

SMCC structure The category **G** inherits the symmetric monoidal structure of the category from chapter 3: all the necessary structure is provided by zig-zag strategies with no divergences which means that all the required equations just collapse to those of the previous chapter.

4.5 A Cartesian closed category

4.5.1 Single-threaded strategies

In the previous chapter, we introduced the class of single-threaded strategies with the guiding intuition that the response of such a strategy depends solely on the current thread of the play to date. We must extend this idea for our new strategies to take the divergences into account.

It seems clear that for $\sigma = (T_\sigma, D_\sigma)$ to be single-threaded, its trace set T_σ should be single-threaded in the sense of the previous chapter. But what should this mean for divergences? Viewing divergence as a "give up" option for Player, what we want is for any divergence of σ to have been "caused" by the play in a single thread. More concretely, if $sa \in D_\sigma$ then either $\lceil sa \rceil \in D_\sigma$ or, more generally, some prefix $tb \sqsubseteq sa$ is such that $\lceil tb \rceil \in D_\sigma$. This says that, at some (possibly) earlier stage, the play in a single thread could have caused divergence. It should also be the case that, if we've reached a point $s \in T_\sigma$ and Opponent now plays the move a then, if $\lceil sa \rceil \in D_\sigma$ then this should suffice to make sa a divergence too.

Definition A strategy $\sigma = (T_\sigma, D_\sigma)$ for an arena A is ***single-threaded*** iff T_σ is well-threaded and

$$(sab \in T_\sigma \wedge t \in T_\sigma \wedge ta \in L_A \wedge \lceil sa \rceil = \lceil ta \rceil) \Rightarrow \mathsf{Match}(sab, ta) \in T_\sigma$$

and, additionally, for $s \in T_\sigma$ we have:

(**st1**) $sa \in D_\sigma \Rightarrow \exists tb \sqsubseteq sa.\ \lceil tb \rceil \in D_\sigma$

(**st2**) $\lceil sa \rceil \in D_\sigma \Rightarrow \exists tb \sqsubseteq sa.\ tb \in D_\sigma$.

The plan now is the same as before; we are seeking a one-to-one correspondence between the single-threaded strategies and the comonoid homomorphisms, *i.e.* those $\sigma : A \Rightarrow B$ such that $\sigma ; \Delta_B =^{\natural} \Delta_A ; (\sigma \times \sigma)$.

Lemma 4.5.1 If $\sigma : A \Rightarrow B$ then $sa \in D_{\Delta_A;(\sigma\times\sigma)}$ if, and only if, $sa \in L_{A\Rightarrow(B\times B)}$ and either $(sa \upharpoonright \ell \in D_\sigma \wedge sa \upharpoonright r \in T_\sigma)$ or $(sa \upharpoonright \ell \in T_\sigma \wedge sa \upharpoonright \ell \in D_\sigma)$.

Proof First of all, suppose that $sa \in D_{\Delta_A;(\sigma\times\sigma)}$. Clearly $sa \in L_{A\Rightarrow(B\times B)}$. There must be a witness $u \in \mathsf{int}(A, A \times A, B \times B)$ for sa such that $u \upharpoonright A, A \times A \in T_{\Delta_A}$ and $u \upharpoonright A \times A, B \times B \in D_{\sigma\times\sigma}$. Suppose, WLOG, that $u \upharpoonright A_\ell, B_\ell \in D_\sigma$ and $u \upharpoonright A_r, B_r \in T_\sigma$. But, by lemma 3.5.3, $s \upharpoonright \ell = u \upharpoonright A_\ell, B_\ell$ and $s \upharpoonright r = u \upharpoonright A_r, B_r$ and we're done. For the converse, suppose that $sa \in L_{A\Rightarrow(B\times B)}$ such that $sa \upharpoonright \ell \in D_\sigma$ and $s \upharpoonright r \in T_\sigma$. So $s \in L_{A\Rightarrow(B\times B)}$ such that $s \upharpoonright \ell \in T_\sigma$ and $s \upharpoonright r \in T_\sigma$. By lemma 3.5.4 we have $s \in T_{\Delta_A;(\sigma\times\sigma)}$. This must have a witness $u \in T_{\Delta_A} \parallel T_{\sigma\times\sigma}$ and u can be extended as in lemma 3.5.4 to witness $sa \in D_{\Delta_A;(\sigma\times\sigma)}$. ■

We inductively define, for $n \geq 2$, σ^n and Δ^n_A by:

$$\begin{aligned} \sigma^2 &=_{\mathrm{df}} \sigma \times \sigma \\ \sigma^{n+1} &=_{\mathrm{df}} \sigma \times \sigma^n. \\ \Delta^2_A &=_{\mathrm{df}} \Delta_A \\ \Delta^{n+1}_A &=_{\mathrm{df}} \Delta_A ; (\mathsf{id}_A \times \Delta^n_A). \end{aligned}$$

It's a straightforward verification to establish that, for $\sigma : A \Rightarrow B$,

$$\sigma ; \Delta_B = \Delta_A ; (\sigma \times \sigma) \text{ if, and only if, for all } n \geq 2,\ \sigma ; \Delta^n_B = \Delta^n_A ; \sigma^n.$$

This observation is vital in the proof of the following result. But first of all, some useful notation. If $\sigma : A \Rightarrow B$ then we write LHS as shorthand for $\sigma ; \Delta_B$ and RHS as shorthand for $\Delta_A ; (\sigma \times \sigma)$.

Proposition 4.5.2 A strategy $\sigma : A \Rightarrow B$ is single-threaded if, and only if, it's a comonoid homomorphism.

Proof We prove the "only if" direction first. This entails showing that, if σ is single-threaded then $\sigma ; \Delta_B =^{\natural} \Delta_A ; (\sigma \times \sigma)$. We already know that $T_{\mathrm{LHS}} = T_{\mathrm{RHS}}$ so it suffices to check that $\sigma ; \Delta_B =^{\sharp} \Delta_A ; (\sigma \times \sigma)$.

If $sa \in D_{\mathrm{LHS}}$ then it must have a witness $u \in \mathsf{int}(A, B, B \times B)$ such that $u \upharpoonright A, B \in D_\sigma$. Since σ is single-threaded we have some $tb \sqsubseteq u \upharpoonright A, B$ such that $\lceil tb \rceil \in D_\sigma$. WLOG, suppose that b is hereditarily justified by an initial occurrence from L (L and R are defined as before) so that $\lceil tb \upharpoonright L \rceil = \lceil tb \rceil$. Applying single-threading of σ again, we get some $t'b' \sqsubseteq tb \upharpoonright L$ satisfying $t'b' \in D_\sigma$. Consider $u_{\leqslant b'}$. Now, $(u_{\leqslant b'} \upharpoonright A, B \times B) \upharpoonright \ell = (u_{\leqslant b'} \upharpoonright A, B) \upharpoonright L = t'b' \in D_\sigma$. Similarly we have $(u_{\leqslant b'} \upharpoonright A, B \times B) \upharpoonright r = (u_{\leqslant b'} \upharpoonright A, B) \upharpoonright R \in T_\sigma$ and, by the previous lemma, $sa \sqsupseteq u_{\leqslant b'} \upharpoonright A, B \times B \in D_{\mathrm{RHS}}$.

Now, suppose we have $sa \in D_{\mathrm{RHS}}$. This has a witness $u \in \mathrm{int}(A, A \times A, B \times B)$ such that $u \upharpoonright A \times A, B \times B \in D_{\sigma\times\sigma}$. WLOG, suppose that $u \upharpoonright A_\ell, B_\ell \in D_\sigma$ and $u \upharpoonright A_r, B_r \in T_\sigma$. Since σ is single-threaded, we have some $tb \sqsubseteq u \upharpoonright A_\ell, B_\ell$ such that $\lceil tb \rceil \in D_\sigma$. So consider $u_{\leqslant b}$. By an easy zipping-style argument, we can combine $u_{\leqslant b} \upharpoonright A_\ell, B_\ell$ and $u_{\leqslant b} \upharpoonright A_r, B_r$ into $z \in L_{A \Rightarrow B}$ such that $\lceil z \rceil \in D_\sigma$. We apply the single-threading of σ the other way to get some $t'b' \sqsubseteq z$ such that $t'b' \in D_\sigma$. If we now play $t'b'$ out in $\mathrm{int}(A, B, B \times B)$, splitting the initial occurrences between the two copies of B just as in sa, we get a prefix of sa in D_{LHS} as required.

We now consider the "if" direction. For (**st1**), let $\sigma : A \Rightarrow B$ be a comonoid homomorphism and let $sa \in D_\sigma$. Suppose there are n initial occurrences in sa. Then we can find $u \in \mathrm{int}(A, B, B^n)$ such that $sa = u \upharpoonright A, B$, $u \upharpoonright A, B^n \in D_{\sigma;\Delta_B^n}$ and $u \upharpoonright B, B^n \in \Delta_B^n$ where each initial occurrence in u occurs in a different copy of B in B^n. Since $\Delta_A^n ; \sigma^n \leqslant^\natural \sigma ; \Delta_B^n$, we have some $tb \sqsubseteq u \upharpoonright A, B^n$ such that $tb \in D_{\Delta_A^n;\sigma^n}$. But $\lceil tb \rceil \in D_\sigma$ so, if we "compress" tb into $z \in L_{A\Rightarrow B}$ by once again playing all the initial occurrences in a single copy of B, we have $z \sqsubseteq sa$ but also $\lceil z \rceil = \lceil tb \rceil \in D_\sigma$.

For (**st2**), suppose that $s \in T_\sigma$ and $\lceil sa \rceil \in D_\sigma$. Let m be the hereditary justifier of a in sa and let $s' \in L_{A \Rightarrow (B \times B)}$ be the legal play identical to s except that we play m in the left-hand copy of B and all the other initial occurrences in the right-hand copy. By lemma 3.5.1, we know that both $s \upharpoonright m$ and $s \upharpoonleft m$ are traces of σ and so it's easy to find $u \in \mathrm{int}(A, A \times A, B \times B)$ witnessing s' such that $u \upharpoonright A, A \times A \in \Delta_A$ and $u \upharpoonright A \times A, B \times B \in T_{\sigma\times\sigma}$. We now extend u (by one or two moves) to some v satisfying $v \upharpoonright A_\ell, B_\ell = \lceil sa \rceil \in D_\sigma$ and $v \upharpoonright A_r, B_r = s' \upharpoonleft m \in T_\sigma$. By lemma 3.5.3, $(v \upharpoonright A, B \times B) \upharpoonright \ell = v \upharpoonright A_\ell, B_\ell \in D_\sigma$ and $(v \upharpoonright A, B \times B) \upharpoonright r = v \upharpoonright A_r, B_r \in T_\sigma$. So, by lemma 4.5.1, we have $v \upharpoonright A, B \times B \in D_{\mathrm{RHS}}$. Since σ is a comonoid homomorphism, there must be some $tb \sqsubseteq v \upharpoonright A, B \times B$ such that $tb \in D_{\mathrm{LHS}}$. Finally, we "compress" this tb to $z \in L_{A\Rightarrow B}$ by playing all the initial occurrences in the same copy of B. This yields $z \in D_{\mathrm{LHS}}$ satisfying $z \sqsubseteq sa$. ■

Aside There is an alternative characterization of the property of a strategy being single-threaded which can sometimes be more convenient. A strategy σ on arena A is single-threaded if, and only if, for all $sa \in L_A$,

$$\exists tb \sqsubseteq sa.\ tb \in D_\sigma \iff \exists tb \sqsubseteq sa.\ \lceil tb \rceil \in D_\sigma.$$

4.5.2 Cartesian closure

In the light of our characterization of the single-threaded strategies as comonoid homomorphisms, the same argument as in the previous chapter establishes that this class of strategies is closed under composition. We therefore have a subcategory, call it **C**, of **G** where an arrow $f : A \to B$ is a $=^\natural$-equivalence class of single-threaded, finite-branching strategies for $A \Rightarrow B$. Since for any such f we have $f ; \langle \Delta_B \rangle = \langle \Delta_A \rangle ; (f \times f)$, the arrows $\langle \Delta_A \rangle$ form the components of a natural transformation on **C**.

Diagonal categories Let's briefly review the notion of a *diagonal* category from [34]. Suppose we have an "affine" SMCC (*i.e.* where the tensor unit is terminal) equipped with a natural transformation with components $\delta_A : A \to A \otimes A$. This is a ***diagonal*** category iff δ_A is coassociative, cocommutative and the following diagram commutes, asserting a "commutation" between δ and $\otimes$.

$$\begin{tikzcd} & A \otimes B \arrow[dl, "\delta_{A\otimes B}"'] \arrow[dr, "\delta_A \otimes \delta_B"] & \\ A \otimes B \otimes A \otimes B \arrow[rr, "1_A \otimes \gamma_{A,B} \otimes 1_B"'] & & A \otimes A \otimes B \otimes B \end{tikzcd}$$

It's clear that our natural transformation with components $\langle \Delta_A \rangle$ satisfies these conditions so that we may view **C** as a diagonal category. By proposition 4.5 of [34], a diagonal category is Cartesian closed if, and only if, the following diagram commutes.

$$\begin{tikzcd} A \arrow[r, "\delta_A"] \arrow[dr, "1_A"'] & A \otimes A \arrow[d, "!_A \otimes 1_A"] \\ & A \end{tikzcd}$$

This condition is also easily seen to hold for **C** and we conclude that we have a Cartesian closed category.

4.5.3 Order enrichment

We now turn to the order enrichment of **C**. Recall the correspondence established previously (in §3.5) between single-threaded and well-opened trace sets. We seek to extend this to the case where strategies are also equipped with divergences.

Given a single-threaded strategy σ for an arena A, we define the thread relation, written $\mathsf{Rel}(\sigma)$, exactly as before. The ***divergent threads*** of σ, written $\mathsf{DT}(\sigma)$, are those divergences that are well-opened plays.

$$\mathsf{DT}(\sigma) = \{sa \in D_\sigma \mid sa = \lceil sa \rceil\}$$

A strategy σ for an arena A is ***well-opened*** iff T_σ is a well-opened trace set, in the sense of the previous chapter, and all the *interesting* divergences of D_σ are well-opened legal plays. Given such a strategy, we define $\mathsf{Traces}(\sigma)$ as in chapter 3. For the divergences, we define:

$$\begin{aligned} D_n(D_\sigma) &= \{sa \in L_A \mid s \in T_n(T_\sigma) \wedge \lceil sa \rceil \in D_\sigma\} \\ \mathsf{Divs}(\sigma) &= \bigcup_{n \in \mathbf{N}_0} D_n(D_\sigma). \end{aligned}$$

It's straightforward to check that, for a single-threaded strategy σ,

$$\mathsf{WO}(\sigma) = (\mathsf{Rel}(\sigma), \mathsf{DT}(\sigma) \cup \{sa \mid s \in \mathsf{Rel}(\sigma) \wedge a \text{ is initial}\})$$

is a well-opened strategy. It's also pretty easy to see that, if $\omega = (T_\omega, D_\omega)$ is well-opened then $\mathsf{ST}(\omega) = (\mathsf{Traces}(T_\omega), \mathsf{Divs}(D_\omega))$ is a single-threaded strategy. Much as before, we have a one-to-one correspondence between ($=^\natural$-equivalence classes of) single-threaded and well-opened strategies:

$$\begin{aligned} \sigma &=^\natural \mathsf{ST}(\mathsf{WO}(\sigma)) \\ \mathsf{WO}(\mathsf{ST}(\omega)) &=^\natural \omega \end{aligned}$$

Lemma 4.5.3 If σ and τ are single-threaded strategies then $\sigma \leqslant^\natural \tau$ if, and only if, $\mathsf{WO}(\sigma) \leqslant^\natural \mathsf{WO}(\tau)$.

Proof The part concerning the traces, *i.e.* $\sigma \leqslant^\flat \tau$ if, and only if, $\mathsf{Rel}(\sigma) \leqslant^\flat \mathsf{Rel}(\tau)$, is known from the previous chapter. For the upper half, the left-to-right case is easy. For the other direction, let $d \in D_\tau$. By single-threading, we have some $d' \sqsubseteq d$ such that $\lceil d' \rceil \in D_\tau$. So $\lceil d' \rceil \in \mathsf{DT}(\tau)$ and, since $\mathsf{WO}(\sigma) \leqslant^\natural \mathsf{WO}(\tau)$, we have some $e' \sqsubseteq \lceil d' \rceil$ such that $e' \in \mathsf{DT}(\sigma)$. So $e' \in D_\sigma$ and, by single-threading, we have some $e \sqsubseteq d$ such that $e \in D_\sigma$. For the final bit, suppose $sab \in T_\tau$ but $sab \notin T_\sigma$. We must have some $s'a'b' \sqsubseteq sab$ such that $s' \in T_\sigma$ but $s'a'b' \notin T_\sigma$. Since $T_\sigma = \mathsf{Traces}(\mathsf{Rel}(\sigma))$, we have $\lceil s'a'b' \rceil \in \mathsf{Rel}(\tau)$ but $\lceil s'a'b' \rceil \notin \mathsf{Rel}(\sigma)$. Applying $\mathsf{WO}(\sigma) \leqslant^\natural \mathsf{WO}(\tau)$, we have $d' \in \mathsf{DT}(\sigma)$ such that $d' \sqsubseteq \lceil s'a'b' \rceil$. But then $d' \in D_\sigma$ and, by single-threading, we have some $d \in D_\sigma$ such that $d \sqsubseteq s'a'b' \sqsubseteq sab$. ∎

Lemma 4.5.4 The set of ($=^\natural$-equivalence classes of) single-threaded strategies for A ordered by $\leqslant^\natural$, written $\mathsf{STr}(A)$, is an algebraic CPO. The compact elements are those $\langle \sigma \rangle$ such that $\mathsf{Rel}(\sigma)$ is finite.

Proof Let Δ be a directed subset of $\mathsf{STr}(A \Rightarrow B)$. We have:

$$(\bigsqcup \Delta) ; \Delta_A =^\natural \bigsqcup (\Delta ; \Delta_A) =^\natural \bigsqcup (\Delta_B ; \Delta^2) =^\natural \Delta_B ; \bigsqcup \Delta^2 =^\natural \Delta_B ; (\bigsqcup \Delta)^2$$

so that $\bigsqcup \Delta$ is single-threaded. For compactness, let Δ be a directed set such that $\sigma \leqslant^\natural \bigsqcup \Delta$. By the above lemma, $\mathsf{WO}(\Delta) = \{\mathsf{WO}(\tau) \mid \tau \in \Delta\}$ is directed and $\mathsf{WO}(\sigma) \leqslant^\natural \mathsf{WO}(\bigsqcup \Delta) = \bigsqcup \mathsf{WO}(\Delta)$. By proposition 4.3.9, if $\mathsf{Rel}(\sigma)$ is finite then we have $\mathsf{WO}(\tau) \in \mathsf{WO}(\Delta)$ such that $\mathsf{WO}(\sigma) \leqslant^\natural \mathsf{WO}(\tau)$ and so $\sigma \leqslant^\natural \tau$ for some $\tau \in \Delta$. If $\mathsf{Rel}(\sigma)$ is infinite, the argument of proposition 4.3.9 yields a directed set Δ_σ of well-opened strategies, all with finite thread relations, such that $\mathsf{WO}(\sigma) =^\natural \bigsqcup \Delta_\sigma$. Hence, for no $\tau \in \Delta_\sigma$ do we have $\mathsf{WO}(\sigma) \leqslant^\natural \tau$ and so, for no $\mathsf{ST}(\tau) \in \mathsf{ST}(\Delta_\sigma) = \{\mathsf{ST}(\tau) \mid \tau \in \Delta_\sigma\}$ do we have $\sigma \leqslant^\natural \mathsf{ST}(\tau)$. So a strategy σ is compact if, and only, if $\mathsf{Rel}(\sigma)$ is finite and, by the same argument as for proposition 4.3.9, σ is (equivalent to) the lub of the compact elements below it. ∎

4.6 Reliable deterministic factorization

In the previous chapter, we proved a factorization theorem showing that any nondeterministic strategy can be expressed as the composite of a nondeterministic "oracle" strategy with a deterministic strategy. However, to do this in our new setting with divergences, we must factor out not only the "observable" nondeterminism, as exemplified by $\{\varepsilon, \mathsf{q}\mathtt{tt}, \mathsf{q}\mathtt{ff}\}$, but also the "invisible" nondeterminism exemplified by $\langle\{\varepsilon, \mathsf{q}\mathtt{tt}\}, \{\mathsf{q}\}\rangle$.

Consider axiom (**d1**). This says that, if a strategy reaches a point where it has no possible move to play—a *zero-choice point*—then there must be a divergence at that point, if not earlier. But if we wish to factor out "invisible" nondeterminism, it must be the case that we have a divergence in our strategy precisely when a zero-choice point is reached. We now formalize this idea by introducing the class of *reliable* strategies.

4.6.1 Reliability & subdeterminism

In the light of the above informal discussion, we'd expect a reliable strategy to have no points at which it could "choose" between diverging and continuing. In other words, strategies like $(\{\varepsilon, \mathsf{q}0\}, \{\mathsf{q}\})$ are definitely out. But what about a strategy like $(\{\varepsilon, \mathsf{q}0, \mathsf{q}1\}, \varnothing)$? It could be argued that, since this is certainly always guaranteed to *respond* to any request, it should be considered reliable. On the other hand, we could never *rely* on this strategy to produce a 'one' at any given moment. For example, if we were to provide this as input to a strategy for 'predecessor' that diverges on an input of 'zero', we might see the following:

$$
\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{N} & ; & \mathbf{N} & \Rightarrow & \mathbf{N} \\
 & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & \\
 & & \mathsf{q} & & & & \\
 & & 0 & & & & \\
 & & & & 0 & &
\end{array}
$$

But this composition could equally well succeed, producing an output of 'zero', so if we want a robust notion of reliability, we're obliged to exclude $(\{\varepsilon, \mathsf{q}0, \mathsf{q}1\}, \varnothing)$. The most natural way to proceed seems to be that of restricting to strategies with deterministic trace sets.

Definition A strategy σ on an arena A is ***reliable*** iff T_σ is deterministic and if $sa \in D_\sigma$ then $sa \notin \mathsf{dom}(\sigma)$. All the divergences in a reliable strategy are therefore uninteresting; the "boundary" between the strategy's reliable and unreliable behaviour is entirely determined by its trace set.

Proposition 4.6.1 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are reliable strategies then so is $\sigma\,;\tau : A \Rightarrow C$.

Proof We already know that $T_{\sigma;\tau}$ is deterministic. So, suppose that $sa \in D_{\sigma;\tau}$. Since σ and τ are both reliable, the necessarily unique interaction continuing from sa either reaches a zero-choice point or enters into an infinite chatter and, in either case, there can be no response by $\sigma\,;\tau$ to sa. ■

Deterministic trace sets & reliable strategies Intuitively, the divergences of a reliable strategy are redundant. We can make this more precise. For the purposes of this little section, we refer to strategies in the sense of the previous chapter as T-strategies and strategies in the sense of this chapter as TD-strategies.

Given a deterministic T-strategy σ for arena A, we define a reliable TD-strategy $\sigma_{\lightning}$ for A by:

$$(\sigma,\ \{sa \in L_A \mid s \in \sigma \wedge sa \notin \mathsf{dom}(\sigma)\}).$$

We define a functor, written $\lightning(-)$, from the category, denoted $\mathbf{T}_d$, of deterministic T-strategies to the category, denoted **TD**, of TD-strategies.

$$\begin{array}{rcl} A & \mapsto & A \\ \sigma : A \to B & \mapsto & \langle \sigma_{\lightning} \rangle : A \to B \end{array}$$

That this is a functor follows easily from the above proposition. We can, to some extent, go back the other way: the evident forgetful functor, written $U(-)$, goes from $\mathbf{TD}_r$, the subcategory of reliable strategies, to $\mathbf{T}_d$. If σ_t is a T-strategy and σ_r is a reliable TD-strategy, we have:

$$\begin{array}{rcl} U(\lightning(\sigma_t)) & = & \sigma_t \\ \sigma_r & =^{\natural} & \lightning(U(\sigma_r)). \end{array}$$

This sets up an isomorphism between $\mathbf{T}_d$ and $\mathbf{TD}_r$. We can therefore indifferently consider the fully abstract model of IA as being in $\mathbf{T}_d$, as it's usually presented, or in $\mathbf{TD}_r$. With this is mind, we now turn to the question of factorizing out unreliability, the reliable strategies being our intended target.

Subdeterministic strategies It turns out to be useful to approach this problem by first considering the question in a restricted setting. We say that a strategy σ is ***subdeterministic*** iff T_σ is deterministic. Any reliable strategy is subdeterministic, but we also have unreliable things such as $(\{\varepsilon, q\mathtt{tt}\}, \{q\})$ on the Boolean arena.

The factorization of a subdeterministic σ for arena A proceeds by constructing a reliable $\mathsf{Rel}(\sigma)$ for $\mathbf{C} \Rightarrow A$; the canonical subdeterministic strategy, *i.e.* our "oracle", is then just an "unreliable skip" strategy, *i.e.* $(\{\varepsilon, qa\}, \{q\})$ for $\mathbf{C}$, which is prodded whenever we need to introduce unreliability.

The construction of $\mathsf{Rel}(\sigma)$ is as follows. Consider $sa \in \mathsf{dom}(\sigma)$. If $sa \in \mathsf{rp}(\sigma)$ then $\mathsf{Rel}(\sigma)$ behaves just like σ. Otherwise, assuming that the (unreliable) response of σ to sa is b, $\mathsf{Rel}(\sigma)$ plays like:

$$\begin{array}{ccc}
\mathbf{C} & \Rightarrow & A \\
 & & \vdots \\
 & & a \\
\mathsf{q} & & \\
\mathsf{a} & & \\
 & & b.
\end{array}$$

The definition of $\mathsf{Rel}(\sigma)$ is completed by stipulating its divergences to be precisely those $sa \in L_{\mathbf{C}\Rightarrow A}$ such that $s \in T_{\mathsf{Rel}(\sigma)}$ and $sa \upharpoonright A \notin \mathsf{dom}(\sigma)$.

It should be clear that plugging in the unreliable skip strategy gives back a strategy for A equivalent to σ. It needn't be equal to σ since the above definition causes $\mathsf{Rel}(\sigma)$ to interrogate the oracle for each $sa \in \mathsf{dom}(\sigma)$ such that $sa \notin \mathsf{rp}(\sigma)$. This has the effect of throwing in all extensions of divergences in σ, even if they weren't in the original strategy. For example, composing $\mathsf{Rel}((\{\varepsilon, \mathsf{qq}, \mathsf{qqaa}\}, \{\mathsf{q}\}))$ with the unreliable skip strategy yields $(\{\varepsilon, \mathsf{qq}, \mathsf{qqaa}\}, \{\mathsf{q}, \mathsf{qqa}\})$.

$$\begin{array}{ccccccc}
\mathbf{1} & \Rightarrow & \mathbf{C} & \Rightarrow & (\mathbf{C} & \Rightarrow & \mathbf{C}) \\
 & & & & & & \mathsf{q} \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
 & & & & \mathsf{q} & & \\
 & & & & \mathsf{a} & & \\
 & & \mathsf{q} & & & & \\
 & & \mathsf{a} & & & & \\
 & & & & & & \mathsf{a}
\end{array}$$

4.6.2 Reliable deterministic factorization

The above sketch of a factorization from subdeterministic to reliable strategies is intended to illustrate the general mechanism by which the "choice" of divergence can be delegated to an oracle strategy: certain behaviours (in that case, all one of them) of the oracle are unreliable.

Now, in the above subdeterministic case, if some $sa \in \mathsf{dom}(\sigma)$ is reliable then we have no need of the oracle at all. In the general case, where σ's trace set may be nondeterministic, we might still need the oracle in order to factor out some nondeterminism—presumably via some mechanism analogous to that described in the previous chapter—even if σ's behaviour is completely reliable at this point. It's therefore essential that some behaviours of the oracle are reliable, otherwise we might accidentally introduce new possibilities of divergence and the factorization will be incorrect.

Outline of the factorization Suppose σ is some finite-branching strategy for the arena A and let $sa \in \mathsf{dom}(\sigma)$. There are three cases we have to deal with.

- $sa \in \mathsf{rp}(\sigma)$. In this case, since σ is finite-branching, we know that $\mathsf{rng}_\sigma(sa)$ is finite. At this point, we need the oracle to *reliably* choose our next move from this finite collection of possibilities.
- $sa \notin \mathsf{rp}(\sigma)$ and $\mathsf{rng}_\sigma(sa)$ is finite. In this case, we need the oracle to do two things: it must choose our next move and do so in an *unreliable* fashion so that the factorized strategy can itself be reliable.
- $sa \notin \mathsf{rp}(\sigma)$ and $\mathsf{rng}_\sigma(sa)$ is infinite. In this final case, the oracle must pick our next move; since there are an infinite number of choices, the oracle will have to be unreliable at this point.

We can illustrate all three cases with the following (highly contrived) strategy on $\mathbf{N}$: its trace set is

$$\{\varepsilon, \mathsf{q2}, \mathsf{q3}, \mathsf{q2q2}, \mathsf{q2q3}\} \cup \{\mathsf{q2q2qn} \mid n \in \mathbf{N}\};$$

its only interesting divergences are $\mathsf{q2q}$ and $\mathsf{q2q2q}$. The domain of this strategy contains q, $\mathsf{q2q}$ and $\mathsf{q2q2q}$, which correspond exactly to the three cases above. The factorization can now proceed, in much the same way as before, via an encoding of Player's moves (in A) as numbers. The tricky point is to get a correct treatment of the first case where we must proceed cautiously to avoid accidental divergence. We first describe an oracle strategy capable of the required behaviour.

The oracle Given these requirements on its behaviour, what might the oracle strategy look like? A first thought might be that it's a strategy for $\mathbf{N} \times \mathbf{N}$ where, say, the first component is reliable and the second unreliable. This is insufficient since, in the first case, we need to inform the oracle as to *how much* reliable nondeterminism is required. This leads us to consider the oracle as a strategy for $(\mathbf{N} \Rightarrow \mathbf{N}) \times \mathbf{N}$; the first component is now a strategy that consumes a number n and *reliably* produces any number between 0 and n (say).

A closer look at the second and third cases reveals that we may as well identify them: once σ is unreliable, it doesn't matter whether we're trying to factorize a finite or infinite number of choices. We can therefore make an economy with the type of the oracle: we fix, by convention, that providing an input of 0 causes the oracle to (necessarily unreliably) be able to produce *any* natural number; so the oracle just becomes a strategy for $\mathbf{N} \Rightarrow \mathbf{N}$.

In the light of the above remarks, we now define a strategy oracle for the arena $\mathbf{N} \Rightarrow \mathbf{N}$. Its trace set is that generated by the following thread relation:

$$\{\varepsilon, \mathsf{qq}\} \cup \{\mathsf{qq0n} \mid n \in \mathbf{N}_0\} \cup \bigcup_{i \in \mathbf{N}} \{\mathsf{qqij} \mid j \leq i\};$$

its only divergent thread is $\mathsf{qq0}$.

Proposition 4.6.2 If σ is a finite-branching strategy on arena A then there exists a reliable strategy $\mathsf{det}(\sigma)$ on $(\mathbf{N} \Rightarrow \mathbf{N}) \Rightarrow A$ such that $\sigma =^{\natural} \mathsf{oracle} \,;\, \mathsf{det}(\sigma)$. If σ is compact then $\mathsf{det}(\sigma)$ can be taken compact.

Proof Let $\mathsf{code}_A(-)$ be an injective function from the even-length legal plays of A to $\mathbf{N}_0$ and suppose that $sa \in \mathsf{dom}(\sigma)$. Following the above discussion, we identify two cases, depending on whether $sa \in \mathsf{rp}(\sigma)$ or not.

If so, by the finite-branching condition, the range of σ at sa must be finite, *i.e.* $\mathsf{rng}_\sigma(sa) = \{b_1, \ldots, b_n\}$. Let m be the maximum of $\{\mathsf{code}_A(sab_1), \ldots, \mathsf{code}_A(sab_n)\}$ and assume that $\mathsf{code}_A(sab_1) < \mathsf{code}_A(sab_2) < \ldots < \mathsf{code}_A(sab_n)$. $\mathsf{det}(\sigma)$ proceeds as follows, taking care to avoid introducing any possibility of divergence.

$$
\begin{array}{ccccc}
(\mathbf{N} & \Rightarrow & \mathbf{N}) & \Rightarrow & A \\
 & & \vdots & & \\
 & & & & a \\
 & & q & & \\
q & & & & \\
m & & & & \\
 & & j & & \\
 & & & & b_i
\end{array}
$$

If $i = 1$ then $0 \leq j \leq \mathsf{code}_A(sab_1)$ or, for $1 < i \leq m$, $\mathsf{code}_A(sab_{i-1}) < j \leq \mathsf{code}_A(sab_i)$. In other words, $\mathsf{det}(\sigma)$ "consults the oracle" who responds by asking "how much nondeterminism do you want to factorize?" to which $\mathsf{det}(\sigma)$ says m. The oracle will then *reliably* provide a number between 0 and m. $\mathsf{det}(\sigma)$ divides up the numbers between 0 and m by: if we get j such that $0 \leq j \leq \mathsf{code}_A(sab_1)$, we play b_1; if $\mathsf{code}_A(sab_1) < j \leq \mathsf{code}_A(sab_2)$, we play b_2, etc.

The other case is if $sa \notin \mathsf{rp}(\sigma)$. By convention, we supply the oracle with input 0. The oracle's response to this will be to supply any natural number but with the possibility of divergence. The strategy continues in the following fashion.

$$
\begin{array}{ccccc}
(\mathbf{N} & \Rightarrow & \mathbf{N}) & \Rightarrow & A \\
 & & \vdots & & \\
 & & & & a \\
 & & q & & \\
q & & & & \\
0 & & & & \\
 & \mathsf{code}(sab_i) & & & \\
 & & & & b_i
\end{array}
$$

It's clear that $\mathsf{det}(\sigma)$ is a reliable strategy and that if σ is compact then so is $\mathsf{det}(\sigma)$. It's also easy to check that $\mathsf{oracle} \,;\, \mathsf{det}(\sigma) =^{\natural} \sigma$: the essential point is that the cautious case of the factorization, where $sa \in \mathsf{rp}(\sigma)$, ensures that no divergence can arise in $\mathsf{oracle} \,;\, \mathsf{det}(\sigma)$ until σ is unreliable, *i.e.* σ and $\mathsf{oracle} \,;\, \mathsf{det}(\sigma)$ share the same set of minimal divergences. ■

4.6.3 Preservation of constraints

Our main concern in this section is to be sure that the result of factorizing a single-threaded strategy σ is itself single-threaded. As before, we proceed by factorizing $\mathsf{WO}(\sigma)$. We therefore need to check the analogue of lemma 3.5.9(1).

Lemma 4.6.3 If σ is a single-threaded strategy then σ is reliable if, and only if, $\mathsf{WO}(\sigma)$ is reliable.

Proof The trace part is just lemma 3.5.9(1). For the left-to-right implication, it suffices to consider $sa \in \mathsf{DT}(\sigma)$. By definition, $sa \in D_\sigma$ so, by reliability of σ, $sa \notin \mathsf{dom}(\sigma)$. If there were any $tab \in T_\sigma$ such that $\lceil sa \rceil = \lceil ta \rceil$ then we'd have $\mathsf{Match}(tab, sa) \in T_\sigma$. But since we don't, we conclude that $sa \notin \mathsf{dom}(\mathsf{Rel}(\sigma))$. For the right-to-left case, let $sa \in D_\sigma$. By single-threading, we have some $d \sqsubseteq sa$ such that $\lceil d \rceil \in D_\sigma$. Applying single-threading again, we find a $d' \sqsubseteq d$ such that $d' \in D_\sigma$. Since $d' \sqsubseteq sa$ and σ is reliable, we have $d' = d = sa$. So $\lceil sa \rceil \in D_\sigma$ and hence in $\mathsf{DT}(\sigma)$. By reliability of $\mathsf{WO}(\sigma)$, $\lceil sa \rceil \notin \mathsf{dom}(\mathsf{Rel}(\sigma))$ and so, by single-threading of σ, $sa \notin \mathsf{dom}(\sigma)$. ■

So, given a single-threaded strategy σ, we define $\mathsf{Det}(\sigma)$ to be $\mathsf{ST}(\mathsf{det}(\mathsf{WO}(\sigma)))$. In order to show that this is sensible, we need to extend lemma 3.5.11.

Lemma 4.6.4 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are single-threaded strategies then $\mathsf{WO}(\sigma ; \tau) =^\sharp \sigma ; \mathsf{WO}(\tau)$.

Proof Observe that $\mathsf{WO}(\sigma) \leqslant^\natural \sigma$ and $\sigma \leqslant^\sharp \mathsf{WO}(\sigma)$. We calculate: $\sigma ; \mathsf{WO}(\tau) \leqslant^\natural \sigma ; \tau \leqslant^\sharp \mathsf{WO}(\sigma ; \tau)$ and $\mathsf{WO}(\sigma ; \tau) \leqslant^\natural \sigma ; \tau \leqslant^\sharp \sigma ; \mathsf{WO}(\tau)$. ■

Given a single-threaded σ, we calculate:

$$\begin{aligned} \mathsf{WO}(\mathsf{oracle} ; \mathsf{Det}(\sigma)) &=^\natural \mathsf{oracle} ; \mathsf{WO}(\mathsf{Det}(\sigma)) \\ &=^\natural \mathsf{oracle} ; \mathsf{det}(\mathsf{WO}(\sigma)) \\ &=^\natural \mathsf{WO}(\sigma) \end{aligned}$$

and applying $\mathsf{ST}(-)$ gives us $\sigma =^\natural \mathsf{oracle} ; \mathsf{Det}(\sigma)$ as desired.

Visibility & bracketing In the previous chapter, we defined the constraints of P-visibility and P-bracketing. We extend these notions to strategies with divergences in the evident way: σ is P-vis (resp. P-brk) iff T_σ is P-vis (resp. P-brk) in that original sense. Note that these are purely constraints on the behaviour of Player; this is why they need make no reference to the divergences of σ.

Using lemma 3.6.2, it's easy to see that, if σ satisfies one or other (or both) of these constraints then so does $\mathsf{det}(\sigma)$. Furthermore, by lemma 3.5.9, both survive the passage to the well-opened representative of a single-threaded strategy (and back again). From now on, we restrict our attention to those strategies satisfying both these constraints; we denote this category by $\mathbf{C}_{vb}$.

4.7 Full abstraction for M&M-equivalence

4.7.1 The model of Erratic IA

The model is built exactly as in the previous chapter. We have a Cartesian closed category so we can handle the λ-calculus; the basic numerals, arithmetic and conditional terms are interpreted as before, the strategies used—*e.g.* succ—all having no divergences. Since our category is CPO-enriched, we model recursion in the usual way. The additional types and terms of IA are also easily accounted for. Even the interpretation of $\mathtt{or}$ is unchanged.

As an example of how and when divergences play a rôle in the model, consider the PCF-term $\mathbf{Y}_{\mathtt{Nat}}(\lambda x.\ \mathtt{0}\ \mathtt{or}\ \mathtt{succ}\ x)$. As we've already seen, this *may* converge to any numeral; but it also *may* diverge if it never jumps to the left. We build an ω-chain modelling the "finite unwindings" of this term

$$(\{\varepsilon\}, \{q\}) \leqslant^{\natural} (\{\varepsilon, q0\}, \{q\}) \leqslant^{\natural} (\{\varepsilon, q0, q1\}, \{q\}) \leqslant^{\natural} \cdots$$

and, since every strategy in this ω-chain has q as a divergence, so does its lub:

$$[\![\mathbf{Y}_{\mathtt{Nat}}(\lambda x.\ \mathtt{0}\ \mathtt{or}\ \mathtt{succ}\ x)]\!] = (\top_{\mathbf{N}}, \{q\}).$$

4.7.2 Soundness

The trace part of the model is identical to that of chapter 3. In particular, the consistency and adequacy results from there are still valid for reasoning about the operational semantics; the only thing to note is that the interpretation $[\![s]\!]$ of a Γ-store s has to be taken as the image under $\lightning(-)$ of its interpretation in chapter 3, *i.e.* we have a divergence for a given variable iff s is undefined for that variable.

We have the usual substitution lemma.

Lemma 4.7.1 If $\Gamma, x : T \vdash M : U$ and $\Gamma \vdash N : T$ then $[\![M[N/x]]\!] = \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![N]\!]\rangle \,;\, [\![M]\!]$.

Proof By induction on the structure of M. ■

As before, we also have a little lemma reflecting the fact that there is no stateful or nondeterministic behaviour at higher types.

Lemma 4.7.2 If $\Gamma \vdash M : T$ where T is either of the form $T_1 \to T_2$ or $\mathtt{Var}$ and $\langle s, M\rangle \Downarrow \langle s, V\rangle$ then $[\![M]\!] = [\![V]\!]$.

Proof By induction on the derivation that $\langle s, M\rangle \Downarrow \langle s, V\rangle$. ■

We now proceed to a consistency result showing that if a program, *i.e.* closed term of type $\mathtt{Com}$, M must converge then we don't have a divergence q in $[\![M]\!]$.

Lemma 4.7.3 Let $\Gamma \vdash M : T$ be a term where T is some base type, s be a Γ-store and suppose that $\langle s, M\rangle \Downarrow^{\mathrm{must}}_{\Gamma}$. If T is $\mathtt{Com}$ or $\mathtt{Nat}$ then $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$. If T is $\mathtt{Var}$ then $\mathsf{i} \in D_{[\![s]\!];[\![M]\!]}$ if, and only if, $\mathsf{i} \in D_{[\![s]\!];[\![V]\!]}$ for the V such that $\langle s, M\rangle \Downarrow \langle s, V\rangle$.

Proof By induction on the derivation that $\langle s, M\rangle \Downarrow^{\mathrm{must}}$. The base cases, where M is a canonical form, are all trivial. We illustrate the argument for the key cases below.

If M is of the form '$M_1 M_2$' then we have that $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$. So $\langle s, M_1\rangle \Downarrow \langle s, V\rangle$ for a unique canonical form V of the form $\lambda x.\, M'$. By hypothesis, $\langle s, M'[M_2/x]\rangle \Downarrow^{\mathrm{must}}$. If T is $\mathtt{Com}$ or $\mathtt{Nat}$, we apply the inductive hypothesis, yielding $\mathsf{q} \notin D_{[\![s]\!];[\![M'[M_2/x]]\!]}$ and, since $[\![M_1 M_2]\!] = [\![M'[M_2/x]]\!]$, we have $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$ as desired. If T is $\mathtt{Var}$, the inductive hypothesis gives us: $\mathsf{i} \in D_{[\![s]\!];[\![M'[M_2/x]]\!]}$ if, and only if, $\mathsf{i} \in D_{[\![s']\!];[\![V]\!]}$ for some V such that $\langle s, M'[M_2/x]\rangle \Downarrow \langle s', V\rangle$. But $\mathsf{i} \in D_{[\![s]\!];[\![M]\!]}$ if, and only if, $\mathsf{i} \in D_{[\![s]\!];[\![M'[M_2/x]]\!]}$ and $\langle s, M'[M_2/x]\rangle \Downarrow \langle s', V\rangle$ if, and only if, $\langle s, M\rangle \Downarrow \langle s', V\rangle$ so we're done.

We illustrate the "arithmetic" cases with nondeterministic choice: if M is of the form 'M_1 $\mathtt{or}$ M_2' then we know that $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$ and $\langle s, M_2\rangle \Downarrow^{\mathrm{must}}$. By the inductive hypothesis, we have $\mathsf{q} \notin D_{[\![s]\!];[\![M_1]\!]}$ and $\mathsf{q} \notin D_{[\![s]\!];[\![M_2]\!]}$. But then, by definition of $[\![M]\!]$, $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$ as desired.

If M is of the form '$\mathtt{seq}_{\mathtt{Nat}}$ M_1 M_2' then, according to the definition of $\Downarrow^{\mathrm{must}}$, we have $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$. By the inductive hypothesis, $\mathsf{q} \notin D_{[\![s]\!];[\![M_1]\!]}$. We are therefore guaranteed to at least reach the following juncture:

$$
\begin{array}{ccccccccc}
\mathbf{1} & \Rightarrow & [\![\Gamma]\!] & \Rightarrow & (\mathbf{C} & \times & \mathbf{N}) & \Rightarrow & \mathbf{N} \\
 & & & & & & & & \mathsf{q} \\
 & & & & \mathsf{q} & & & & \\
 & & \mathsf{t} & & & & & & \\
 & & & & \mathsf{a} & & & & \\
 & & & & & & \mathsf{q} & &
\end{array}
$$

So $\mathsf{qta} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M_1]\!]})$ and, by adequacy for traces, we have $\langle s, M_1\rangle \Downarrow \langle s', \mathtt{skip}\rangle$ for the state s' left by t. But then, by hypothesis, we have that $\langle s', M_2\rangle \Downarrow^{\mathrm{must}}$ and, by the inductive hypothesis, $\mathsf{q} \notin D_{[\![s']\!];[\![M_2]\!]}$. Hence $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$.

If M is of the form '$\mathtt{deref}$ M_1' then we have $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$. By the inductive hypothesis, we have $\mathsf{rd} \in D_{[\![s]\!];[\![M_1]\!]}$ if, and only if, $\mathsf{rd} \in D_{[\![s']\!];[\![V]\!]}$ for the V such that $\langle s, M_1\rangle \Downarrow \langle s, V\rangle$. But if $\langle s, M_1\rangle \Downarrow \langle s, \nu\rangle$ then, by hypothesis, $s(\nu)\downarrow$ and so $\mathsf{rd} \notin D_{[\![s]\!];[\![V]\!]}$. And if $\langle s, M_1\rangle \Downarrow \langle s, \mathtt{mkvar}\ L\ P\rangle$ then, by hypothesis, $\langle s, P\rangle \Downarrow^{\mathrm{must}}$. By the inductive hypothesis, $\mathsf{q} \notin D_{[\![s]\!];[\![P]\!]}$ and hence $\mathsf{rd} \notin D_{[\![s]\!];[\![V]\!]}$. Therefore $\mathsf{rd} \notin D_{[\![s]\!];[\![V]\!]}$ so $\mathsf{rd} \notin D_{[\![s]\!];[\![M_1]\!]}$ and hence $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$.

If M is of the form '$\mathtt{assign}$ M_1 M_2' then we have $\langle s, M_2\rangle \Downarrow^{\mathrm{must}}$ and, by the inductive hypothesis, $\mathsf{q} \notin D_{[\![s]\!];[\![M_2]\!]}$. So any interaction is guaranteed to reach at least the following point.

$$
\begin{array}{ccccccccc}
\mathbf{1} & \Rightarrow & [\![\Gamma]\!] & \Rightarrow & (\mathbf{Var} & \times & \mathbf{N}) & \Rightarrow & \mathbf{C} \\
 & & & & & & & & \mathsf{q} \\
 & & & & & & \mathsf{q} & & \\
 & & \mathsf{t} & & & & & & \\
 & & & & & & \mathsf{k} & & \\
 & & & & \mathsf{wr_k} & & & &
\end{array}
$$

So, if t leaves state s', we have $\langle s, M_2\rangle \Downarrow \langle s', \mathtt{k}\rangle$. But then, by hypothesis, we have $\langle s', M_1\rangle \Downarrow^{\text{must}}$ and $\langle s', M_1\rangle \Downarrow \langle s'', \mathtt{mkvar}\ L\ P\rangle \Rightarrow \langle s'', L\mathtt{k}\rangle \Downarrow^{\text{must}}$. By the inductive hypothesis, $\mathsf{wr_k} \in D_{[\![s']\!];[\![M_1]\!]}$ if, and only if, $\mathsf{wr_k} \in D_{[\![s',V]\!]}$ for the V such that $\langle s', M_1\rangle \Downarrow \langle s', V\rangle$. If $\langle s', M_1\rangle \Downarrow \langle s', v\rangle$ then $\mathsf{wr_k} \notin D_{[\![s']\!];[\![v]\!]}$ since $\mathsf{wr_k} \notin D_{[\![s]\!]}$ for any k or s. If $\langle s', M_1\rangle \Downarrow \langle s', \mathtt{mkvar}\ L\ P\rangle$ then we know that $\langle s', L\mathtt{k}\rangle \Downarrow^{\text{must}}$ and, by the inductive hypothesis, $\mathsf{q} \notin D_{[\![s']\!];[\![L\mathtt{k}]\!]}$. Hence $\mathsf{wr_k} \notin D_{[\![s']\!];[\![V]\!]}$ and so $\mathsf{wr_k}$ cannot be a divergence of $[\![s']\!]\ ;\ [\![M_1]\!]$ and so $\mathsf{q} \notin [\![s]\!]\ ;\ [\![M]\!]$. ■

Collating this result with consistency for traces from chapter 3, we obtain:

Corollary 4.7.4 (consistency) If M is a closed term of type `Com` then if $M \Downarrow^{\text{may}}$ then $\mathsf{qa} \in T_{[\![M]\!]}$ and if $M \Downarrow^{\text{must}}$ then $\mathsf{q} \notin D_{[\![M]\!]}$.

We now wish to establish the converse to consistency: if a program M doesn't have $\mathsf{q} \in D_{[\![M]\!]}$ then $M \Downarrow^{\text{must}}$. As usual, this involves defining a suitable predicate of computability and showing that all terms satisfy it.

Definition All we have to do is amend the two clauses for computability of semi-closed terms of type `Nat` and `Com` to take account of must-convergence.

- $\Gamma \mid - \vdash M : \mathtt{Com}$ is computable iff
  - if $\mathsf{q}t\mathsf{a} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M]\!]})$ where t leaves state s' then $\langle s, M\rangle \Downarrow \langle s', \mathtt{skip}\rangle$;
  - if $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$ then $\langle s, M\rangle \Downarrow^{\text{must}}$.
- $\Gamma \mid - \vdash M : \mathtt{Nat}$ is computable iff
  - if $\mathsf{q}t\mathsf{n} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M]\!]})$ where t leaves state s' then $\langle s, M\rangle \Downarrow \langle s', \mathtt{n}\rangle$;
  - if $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$ then $\langle s, M\rangle \Downarrow^{\text{must}}$.

Lemma 4.7.5 All split terms are computable.

Proof By induction on the structure of the term. The cases of application and abstraction are standard. For $\Gamma \mid - \vdash v : \mathtt{Var}$, it's easy to show that '$\mathtt{assign}\ v\ \mathtt{k}$' is computable for any k; and if $\mathsf{q} \notin D_{[\![s]\!];[\![\mathtt{deref}\ v]\!]}$ then we can deduce that $s(v)\downarrow$ and hence $\langle s, \mathtt{deref}\ v\rangle \Downarrow^{\text{must}}$. We give a sample of the remaining cases.

If $\Gamma \mid - \vdash M : \mathtt{Nat}$ has the form 'M_1 $\mathtt{or}$ M_2', suppose that $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$. By the definition of $[\![M]\!]$, we have $\mathsf{q} \notin D_{[\![s]\!];[\![M_1]\!]}$ and $\mathsf{q} \notin D_{[\![s]\!];[\![M_2]\!]}$. By the inductive hypothesis, M_1 and M_2 are both computable so $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$ and $\langle s, M_2\rangle \Downarrow^{\mathrm{must}}$. But we can now apply the $\Downarrow^{\mathrm{must}}$ rule for $\mathtt{or}$ so we have $\langle s, M\rangle \Downarrow^{\mathrm{must}}$ as desired.

If $\Gamma \mid - \vdash M : \mathtt{Nat}$ has the form '$\mathtt{seq}$ M_1 M_2' then suppose that $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$. This means that $\mathsf{q} \notin D_{[\![s]\!];[\![M_1]\!]}$ and, by the inductive hypothesis, $\langle s, M_1\rangle \Downarrow^{\mathrm{must}}$. This is the first precedent for the $\Downarrow^{\mathrm{must}}$ rule for $\mathtt{seq}$. The second precedent is the implication: $\langle s, M_1\rangle \Downarrow \langle s', \mathtt{skip}\rangle \Rightarrow \langle s', M_2\rangle \Downarrow^{\mathrm{must}}$. If $\langle s, M_1\rangle \Downarrow \langle s', \mathtt{skip}\rangle$ then, by consistency, we have a trace $\mathsf{q}t\mathsf{a} \in T_{[\![s]\!]} \parallel \mathsf{Rel}([\![M_1]\!])$ where t leaves state s'. But, since $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$, we must therefore have $\mathsf{q} \notin D_{[\![s']\!];[\![M_2]\!]}$. We apply the inductive hypothesis again, yielding $\langle s', M_2\rangle \Downarrow^{\mathrm{must}}$ so $\langle s, M\rangle \Downarrow^{\mathrm{must}}$ as desired.

If M has the form $\mathtt{deref}$ M', the result follows from the inductive hypothesis. If $\Gamma \mid - \vdash M : \mathtt{Com}$ has the form '$\mathtt{assign}$ M_1 M_2' then suppose that $\mathsf{q}t\mathsf{a} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M]\!]})$. This trace of $[\![M]\!]$ must be witnessed by a interaction of the form

$$\begin{array}{ccccccccc}
\mathbf{1} & \Rightarrow & [\![\Gamma]\!] & \Rightarrow & (\mathbf{Var} & \times & \mathbf{N}) & \Rightarrow & \mathbf{C} \\
 & & & & & & & & \mathsf{q} \\
 & & & & & & \mathsf{q} & & \\
 & & t' & & & & & & \\
 & & & & & & \mathsf{k} & & \\
 & & & & \mathsf{wr}_{\mathsf{k}} & & & & \\
 & & t'' & & & & & & \\
 & & & & \mathsf{ok} & & & & \\
 & & & & & & & & \mathsf{a}
\end{array}$$

where $t = t't''$. We therefore have $\mathsf{q}t'\mathsf{k} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M_2]\!]})$ and, by the inductive hypothesis, M_2 is computable so that $\langle s, M_2\rangle \Downarrow \langle s', \mathtt{k}\rangle$ for state s' left by trace t'. By the inductive hypothesis again, M_1 is computable so '$\mathtt{assign}$ M_1 $\mathtt{k}$' is computable. Now $\mathsf{q}t''\mathsf{a} \in T_{[\![s']\!]} \parallel \mathsf{Rel}(T_{[\![\mathtt{assign}\ M_1\ \mathtt{k}]\!]})$ so we know that $\langle s', \mathtt{assign}\ M_1\ \mathtt{k}\rangle \Downarrow \langle s'', \mathtt{skip}\rangle$ for state s'' left by trace t'' and, from this, it's easy to see that $\langle s, \mathtt{assign}\ M_1\ M_2\rangle \Downarrow \langle s'', \mathtt{skip}\rangle$ as desired. Suppose now that $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$. It must certainly then be the case that $\mathsf{q} \notin D_{[\![s]\!];[\![M_2]\!]}$ and, by the inductive hypothesis, $\langle s, M_2\rangle \Downarrow^{\mathrm{must}}$. We now need to establish the large implication which is the other precedent for $\langle s, \mathtt{assign}\ M_1\ M_2\rangle \Downarrow^{\mathrm{must}}$. To this end, suppose that $\langle s, M_2\rangle \Downarrow \langle s', \mathtt{k}\rangle$. By consistency, we have a trace $\mathsf{q}t\mathsf{k} \in T_{[\![s]\!]} \parallel \mathsf{Rel}(T_{[\![M_2]\!]})$ where t leaves state s'. Since $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$, it's also the case that $\mathsf{q} \notin D_{[\![s']\!];[\![\mathtt{assign}\ M_1\ \mathtt{k}]\!]}$. By the inductive hypothesis, M_1 is computable and so '$\mathtt{assign}$ M_1 $\mathtt{k}$' is too. Hence $\langle s', \mathtt{assign}\ M_1\ \mathtt{k}\rangle \Downarrow^{\mathrm{must}}$ and, since $\langle s', \mathtt{k}\rangle \Downarrow \langle s', \mathtt{k}\rangle$, we have: $\langle s', M_1\rangle \Downarrow^{\mathrm{must}}_{\Gamma}$ and $(\langle s', M_1\rangle \Downarrow \langle s'', \mathtt{mkvar}\ L\ P\rangle \Rightarrow \langle s'', L\mathtt{k}\rangle \Downarrow^{\mathrm{must}}_{\Gamma})$. So the antecedent of the implication holds for any possible reduction $\langle s, M_2\rangle \Downarrow \langle s', \mathtt{k}\rangle$ and we conclude that $\mathtt{assign}$ M_1 $M_2 \Downarrow^{\mathrm{must}}$.

We lift this result to the full language similarly to before. If M is a semi-closed term of type $\mathtt{Com}$, we define M_k in the usual way. If $\mathsf{q} \notin D_{[\![s]\!];[\![M]\!]}$ then, for some $k \in \mathbf{N}_0$, we have $\mathsf{q} \notin D_{[\![s]\!];[\![M_k]\!]}$. Since M_k is computable, we have $\langle s, M_k\rangle \Downarrow^{\mathrm{must}}$ and, by lemma 2.3.4, $\langle s, M\rangle \Downarrow^{\mathrm{must}}$. ■

As for consistency, we can put this result together with the earlier result for traces, obtaining adequacy and soundness.

Corollary 4.7.6 (adequacy) If M is a closed term of type $\texttt{Com}$ then if $\mathsf{qa} \in T_{[\![M]\!]}$ then $M \Downarrow^{\text{may}}$ and if $\mathsf{q} \notin D_{[\![M]\!]}$ then $M \Downarrow^{\text{must}}$.

Theorem 4.7.7 (soundness) If M and N are closed terms of EIA such that $[\![M]\!] \leqslant^{\natural} [\![N]\!]$ then $M \lesssim_{\text{m\&m}} N$.

Proof The argument for $\Downarrow^{\text{may}}$ follows from Theorem 3.8.7. For the $\Downarrow^{\text{must}}$ part, let $C[-]$ be a program context. If $C[M] \Downarrow^{\text{must}}$ then, by consistency, $\mathsf{q} \notin D_{[\![C[M]]\!]}$. Therefore $\mathsf{q} \notin D_{[\![C[N]]\!]}$ and, by adequacy, $C[N] \Downarrow^{\text{must}}$. ∎

4.7.3 Full abstraction

The passage to full abstraction is now fairly standard. First we have a definability result for the new model.

Proposition 4.7.8 If $\sigma : [\![\Gamma]\!] \Rightarrow [\![T]\!]$ is a compact strategy satisfying P-visibility and P-bracketing then σ is definable in EIA.

Proof We apply the reliable factorization, yielding a compact, reliable strategy σ' for $([\![\Gamma]\!] \times (\mathbf{N} \Rightarrow \mathbf{N})) \Rightarrow [\![T]\!]$. This is definable by some $\Gamma, x : \texttt{Nat} \to \texttt{Nat} \vdash M : T$ in IA so, provided we can define the oracle strategy, we can define σ in the usual way.

The main point about the oracle is that, if it receives an input $n > 0$, it *reliably* produces any output m between 0 and n. The following term does most of the work for this: $\mho_n = \mathbf{Y}_{\texttt{Nat}\to\texttt{Nat}}(\lambda f.\, \lambda x.\, \texttt{if0}\ x\ 0\ (0\ \texttt{or}\ \texttt{succ}\, f(\texttt{pred}\, x)))$. The strategy oracle can now be seen to be definable by:

$$\lambda x.\, \texttt{new}_{\texttt{Nat}}\ (\lambda v.\, \texttt{seq}_{\texttt{Nat}}\ (\texttt{assign}\, v\, x)\ (\texttt{if0}\ (\texttt{deref}\ v)\ \mho\ (\mho_n(\texttt{deref}\ v)))).$$

∎

The intrinsic collapse We apply this construction to $\mathbf{C}_{\nu b}$, once again using $\mathbf{C}$ as the "testing object". Notice that we now have three possible outcomes to a test: definite failure; possible success, but possible failure; and definite success. This is what allows the model to distinguish between 'Ω', '$\texttt{skip}\ \texttt{or}\ \Omega$' and '$\texttt{skip}$'.

The same argument as in the previous chapter shows that the quotiented model, written $\mathcal{E}[\![-]\!]$, inherits the soundness result of §4.7.2. We put this together with definability in the usual way, obtaining our desired full abstraction result.

Theorem 4.7.9 (full abstraction) If M and N are closed terms of type $\texttt{Com}$ then $\mathcal{E}[\![M]\!] \preccurlyeq \mathcal{E}[\![N]\!]$ if, and only if, $M \lesssim_{\text{m\&m}} N$.

Chapter 5

AJM-games, self-equivalence & nondeterminism

We mentioned in the introduction that there are two extant formulations of game semantics in theoretical computing. We've seen the HO setting in some detail in the previous two chapters; we now turn our attention to the AJM-style of games. In some sense, these are more general than HO-style games: the key axiomatic structure of arenas, namely the *enabling* relation, can be seen as an abstraction of the properties of a certain class of AJM-games. This extra structure makes the HO setting the natural choice for modelling programming languages; but the simpler, more primitive nature of AJM-games makes them ideal for studying some of the basic issues of game semantics.

Our particular interest will be in examining some of the issues pertaining to nondeterminism in the AJM setting. As we've seen, there are certain complications arising from the interaction of innocence and nondeterminism in the HO setting. Therefore we might reasonably expect to see this mirrored in some way in the AJM framework. Unfortunately, before we can do this, it's necessary to introduce a whole new collection of definitions. Nevertheless, our development of AJM-style games proceeds at a quicker pace than that of HO-style games in chapter 2 as there is substantial conceptual overlap. The definitions and results of this chapter clearly owe a substantial debt to those of the original AJM paper [1]. The differences that arise are, in the main, clarifications resulting from tackling the problems associated with nondeterminism. However, it should be noted that the Question/Answer aspect of moves has been suppressed, allowing us to keep the definitions simple and the discussion focussed on the essential issues at hand.

The general development of AJM-games has always relied more heavily on ideas from linear logic than has HO-games. The presentation here is no exception to this and a passing acquaintance with the ideas of linear logic might be helpful. However, this shouldn't be considered to be a genuine prerequisite and, indeed, there are one or two places where gamey intuition deviates from the logical norm.

5.1 The basic definitions

5.1.1 Games

We begin with the definition of an AJM-game. As we're eliding any treatment of bracketing, there is only one aspect of moves to be considered, *i.e.* whether a given move is an Opponent move or a Player move. This affords us the (minor) notational advantage of eliminating the labelling function.

Definition A ***game*** G is a triple $\langle \mathsf{O}_\mathsf{G}, \mathsf{P}_\mathsf{G}, \approx_\mathsf{G} \rangle$ where

- O_G and P_G are countable sets of ***moves***; we denote their (disjoint) union by M_G. If $m \in \mathsf{O}_\mathsf{G}$, we say that m is an O-move (of G); similarly, if $m \in \mathsf{P}_\mathsf{G}$ then it's a P-move (of G). The set O_G defines ***Opponent*** in the game G; the set P_G defines ***Player*** in G.

  Let L_G be the set of all $s \in M_\mathsf{G}^\star$ satisfying: $s_i \in \mathsf{O}_\mathsf{G}$ if, and only if, i is odd. So, if $s \in L_\mathsf{G}$ then it begins with an O-move and thereafter strictly alternates between O_G and P_G; we say that s is a ***legal play*** of G.

- $\approx_\mathsf{G}$ is a (specified) partial equivalence relation (*i.e.* symmetric and transitive) on L_G satisfying

  (**e1**) if $s \approx_\mathsf{G} t$ then $|s| = |t|$

  (**e2**) if $sa \approx_\mathsf{G} ta'$ then $s \approx_\mathsf{G} t$

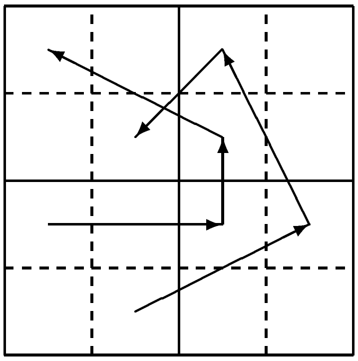
  (**e3**) if $s \approx_\mathsf{G} t$ and $sa \approx_\mathsf{G} sa$ then $\exists a' \in M_\mathsf{G}.\ sa \approx_\mathsf{G} ta'$.

Axioms (**e1**) and (**e2**) ask that $\approx_\mathsf{G}$ is length- and prefix-respecting. In particular, if $s \approx_\mathsf{G} \varepsilon$ then $s = \varepsilon$. We denote by P_G the set of ***valid plays*** (or just ***plays***) of G, defined to be the set of all $s \in L_\mathsf{G}$ such that $s \approx_\mathsf{G} s$. Note that (**e2**) implies that P_G is prefix-closed. Axiom (**e3**) is known as ***extensibility***: if we have two equivalent plays and one of them can be extended by a further move to a new valid play then so can the other—and the new plays are still equivalent.

Aside We can therefore see $\approx_\mathsf{G}$ as serving two purposes: firstly, it specifies the valid plays as a subset of L_G; and secondly, it allows us to partition the valid plays into (partial) equivalence classes. This allows us to look at a game at two levels: the level of "concrete" plays, *i.e.* just valid plays of G, and the level of "abstract" plays, *i.e.* (partial) equivalence classes of valid plays.

The following diagram is intended to illustrate this. We have four "states", each of which is subdivided into four "substates". We regard two sequences of "state transitions" as equivalent if they start in the same state and pass through the same states in the same order, regardless of the particular substates they travel through.

So here we have two (concretely) differing sequences that we would still regard as being equivalent.



Examples of games We can recast our favourite arenas as games in the following fashion. The "empty game" **1** is defined as $\langle\varnothing, \varnothing, \{(\varepsilon, \varepsilon)\}\rangle$. Given a countable set X, the ***flat game over*** X is defined as $\langle\{q\}, \{x \mid x \in X\}, \{(\varepsilon, \varepsilon), (q, q)\} \cup \{(qx, qx) \mid x \in X\}\rangle$. It's worth noting that, with the equivalence specified here, these games are *affine*: Opponent only gets one opportunity to play q. We'll see later how the bang! of linear logic allows us to recover the "repeatability" of arenas. As before, we denote by $\perp$ the flat game over $\varnothing$, by **B** the flat game over $\{\mathtt{tt}, \mathtt{ff}\}$ and by **N** the flat game over $\{0, 1, 2, \ldots\}$.

For a more interesting example, consider a game X where $O_X = \{q_1, q_2\}$, $P_X = \{a_1, a_2, a_3\}$ and the equivalence $\approx_X$ is generated by $q_1 a_1 \approx_X q_2 a_2$ and $q_1 a_3 \approx_X q_2 a_3$. Morally, this game has just one Opponent move (since $q_1 \approx_X q_2$) and two Player responses ($q_1 a_1 \approx_X q_2 a_2$ gives one; $q_1 a_3 \approx_X q_2 a_3$ gives the other). So X is, in some sense, a non-standard representation of the flat game over $\{\mathtt{tt}, \mathtt{ff}\}$.

5.1.2 Multiplicative constructors

The natural primitive constructors on AJM-games are the multiplicatives—tensor $\otimes$ product and linear $\multimap$ arrow—of "intuitionistic" linear logic.

Times We begin by defining the tensor. Given games G and H, we define $G \otimes H$ (read "G times H" or "G tensor H") as follows.

- $O_{G\otimes H} = O_G + O_H$
- $P_{G\otimes H} = P_G + P_H$
- If $s, t \in L_{G\otimes H}$ then $s \approx_{G\otimes H} t$ iff

$$(s_i \in M_G \iff t_i \in M_G) \wedge (s \restriction M_G \approx_G t \restriction M_G) \wedge (s \restriction M_H \approx_H t \restriction M_H).$$

In other words, we build the new Opponent out of the two old Opponents and the new Player is similarly constructed. The interesting point is in the definition of the equivalence. The first conjunct just requires that two equivalent plays are interleaved in the same way. The remaining conjuncts insist that the restrictions to each constituent are equivalent in the old games. An immediate consequence of this is that, given any $s \in P_{G\otimes H}$, we have $s \restriction M_G \in P_G$ and $s \restriction M_H \in P_H$. These are commonly known as ***projection*** conditions and have *no* counterpart in HO-style arenas: if s is a legal play in $A \times B$, this doesn't necessarily mean that $s \restriction A \in L_A$ and $s \restriction B \in L_B$.

Entails Given games G and H, we define $G \multimap H$ (read "G entails H" or "G linear H") in the following fashion.

- $O_{G\multimap H} = P_G + O_H$
- $P_{G\multimap H} = O_G + P_H$
- If $s, t \in L_{G\multimap H}$ then $s \approx_{G\multimap H} t$ iff

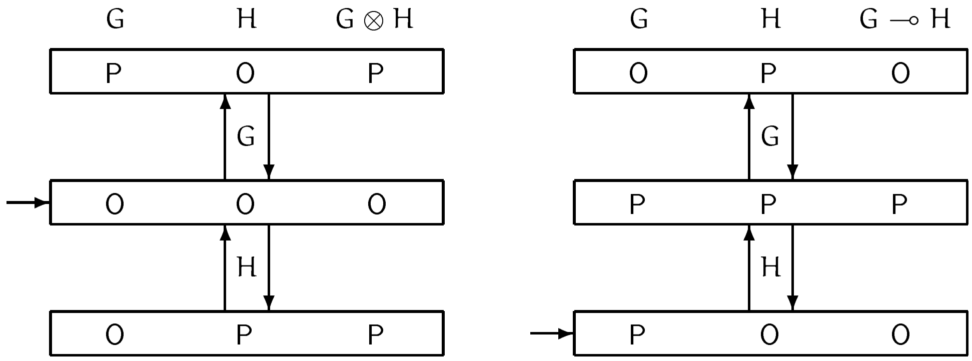
$$(s_i \in M_G \iff t_i \in M_G) \wedge (s \restriction G \approx_G t \restriction G) \wedge (s \restriction H \approx_H t \restriction H).$$

The definitions of the multiplicatives differ only in how they construct Opponent and Player; the equivalence is the same in both cases. This contrasts with the definitions of $\times$ and $\Rightarrow$ on arenas where the enabling relations are quite different. Similar remarks apply to $G \multimap H$ as regards projection conditions: if $s \in P_{G\multimap H}$ then $s \restriction M_G \in P_G$ and $s \restriction M_H \in P_H$.

Nil While not really a multiplicative connective, the linear negation of A, written $A^\perp$, is a useful thing to have around. It is *defined* to be simply $A \multimap \perp$. The effect is to reverse the rôles of Player and Opponent but with an extra "dummy move" added to the beginning of all plays. Doing this twice results in two dummy moves, one for each protagonist, which makes $A^{\perp\perp}$ resemble *lifting* from Chapter 3.

Switching conditions Since all valid plays are required to be alternating, these projection conditions conspire to constrain the shapes of valid plays of $G \otimes H$ and $G \multimap H$. For example, if Opponent starts off with a move of G in $G \otimes H$, we know that the next move overall must be a Player move. Since it's Opponent's go in H, the next move can only be in G. On the other hand, in $G \multimap H$ Opponent is obliged to start in H since any Opponent move of $G \multimap H$ which used to be a move of G was in fact a Player move of G. But after this initial move, Player has a choice: a Player move of $G \multimap H$ is either an Opponent move of G or a Player move of H and either option is available. But now, Opponent is obliged to continue in whichever side Player chooses.

The easiest way to see why is by consideration of the following state diagrams. They're in the same spirit as those for legal interactions but this time we're keeping track of whose turn it is in each constituent game and overall.



So, for example, consider the OPO state of $G \multimap H$. It's Opponent's go overall but, as it's Player's turn in H, we can only play in G. These state diagrams can be summed up in the following slogan: *only Opponent can switch between the constituent games of a tensor; and only Player can switch between the constituent games of a linear arrow.*

5.1.3 Strategies

A ***strategy*** σ on a game G is a non-empty set of even length plays of G satisfying:

$$\text{if } sab \in \sigma \text{ then } s \in \sigma.$$

If σ is a strategy on a game G, we often write $\sigma : G$. All other relevant notation and terminology is inherited directly from the previous chapters. There's an evident notion of determinism for strategies: σ is ***deterministic*** iff

$$(sab \in \sigma \wedge sac \in \sigma) \Rightarrow b = c.$$

In the absence of (explicit) justification information, there is no general analogue to P-visibility or innocence and, as we've left out Questions and Answers at the axiomatic level, there's also nothing corresponding to P-bracketing. However, we do have the following new constraints.

History-freedom A history-free strategy has the property that Player's response to a given move by Opponent depends only on that move—it is completely independent of the "context", *i.e.* of the preceding history. Such strategies are sometimes also known as *memory-less* strategies and can be informally likened to the apocryphal story of the goldfish that spends its life swimming around in circles thinking "what a nice rock that is", "what a nice rock that is"...

This constraint is expressed formally with the following condition. A strategy σ on a game G is ***history-free*** iff

$$(sab \in \sigma \wedge t \in \sigma \wedge ta \in P_G) \Rightarrow tab \in \sigma.$$

Note that part of the force of this condition is that $tab \in P_G$. History-freedom is, just like innocence and single-threading, an example of a *liveness* constraint.

Injectivity Our second constraint is, in some sense, complementary to history-freedom. In a history-free strategy, an (occurence of an) O-move precisely determines what P-moves may follow. In an injective strategy, (an occurence of) a P-move precisely determines its *local context, i.e.* the preceding O-move. Formally speaking, we say that a strategy σ is ***injective*** iff

$$(sab \in \sigma \wedge ta'b \in \sigma) \Rightarrow a = a'.$$

Relational representation of history-free strategies We can find a connection between the history-free strategies on a game G and a certain class of relations between O_G and P_G analogous to the correspondence between single-threaded strategies and well-opened strategies noted in chapter 3. We outline this below, highlighting the novel aspects but leaving out most of the details, which barely differ from the single-threaded/well-opened case.

Given strategy σ for game G, we say that it's ***well-threaded*** iff

$$(sab \in \sigma \wedge ta \in P_G) \Rightarrow tab \in P_G.$$

For any well-threaded strategy σ, we define

$$\mathsf{Rel}(\sigma) = \{(a, b) \mid \exists s \in \sigma.\ s = tab\}.$$

Now, let R be a relation between O_G and P_G. We inductively define:

$$\begin{aligned} T_0(R) &= \{\varepsilon\} \\ T_{i+1}(R) &= \{sab \in L_G \mid s \in T_i(R) \wedge sa \in P_G \wedge (a, b) \in R\} \\ \mathsf{Traces}(R) &= \bigcup_{i \in \mathbf{N}_0} T_i(R) \end{aligned}$$

We say that R is ***safe*** iff $\mathsf{Traces}(R) \subseteq P_G$ and ***saturated*** iff every entry is useful:

$$(a, b) \in R \Longleftrightarrow \exists s \in \mathsf{Traces}(R).\ s = tab.$$

It's clear that, if $\sigma \subseteq \tau$ and $R \subseteq S$, then we have $\mathsf{Rel}(\sigma) \subseteq \mathsf{Rel}(\tau)$ and $\mathsf{Traces}(R) \subseteq \mathsf{Traces}(S)$. It's easy to see that $\mathsf{Rel}(\mathsf{Traces}(R)) \subseteq R$ and a simple induction yields $\sigma \subseteq \mathsf{Traces}(\mathsf{Rel}(\sigma))$. It's also easy to see that $\mathsf{Rel}(\mathsf{Traces}(\mathsf{Rel}(\sigma))) \subseteq \mathsf{Rel}(\sigma)$ and another little induction yields $\mathsf{Traces}(R) \subseteq \mathsf{Traces}(\mathsf{Rel}(\mathsf{Traces}(R)))$.

For any well-threaded strategy σ, the associated relation $\mathsf{Rel}(\sigma)$ is always safe and saturated, safety necessarily relying on well-threading. If R is a safe relation, the associated trace set $\mathsf{Traces}(R)$ is a history-free strategy. It follows from these and the above remarks that $\mathsf{Traces}(\mathsf{Rel}(-))$ is a closure operator on the well-threaded strategies, sending any well-threaded σ to the smallest history-free strategy containing it, and also that there's a one-to-one correspondence between the history-free strategies for G and safe, saturated relations on $O_G \times P_G$.

5.2 Composition & ordering

5.2.1 Composition of strategies

The informal intuition of composition is no different in AJM-games than in HO-games; we proceed by first defining the *parallel composition* and subsequently *hiding* the interaction in the "middle game".

Given $\sigma : G \multimap H$ and $\tau : H \multimap J$, we define their ***parallel composition*** to be

$$\sigma \parallel \tau =_{\mathrm{df}} \{u \in (M_G + M_H + M_J)^\star \mid u \restriction G, H \in \sigma \wedge u \restriction H, J \in \tau\}$$

where $u \restriction G, H$ is an abbreviation for $u \restriction (M_G + M_H)$. Notice how much simpler this is than the corresponding definition in the HO framework. There's no need at all to set up the machinery of "legal interactions" or suchlike.

The definition of composition now follows in the usual way.

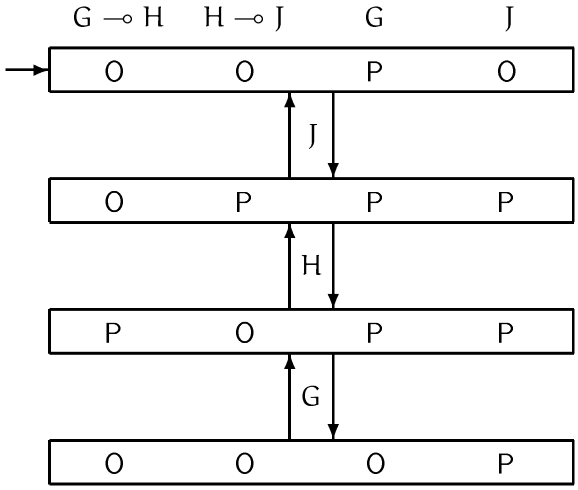
$$\sigma ; \tau =_{\mathrm{df}} \{u \restriction G, J \mid u \in \sigma \parallel \tau\}$$

We need to check that composition is well-defined and associative.

Proposition 5.2.1 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ then $\sigma ; \tau : G \multimap J$.

Proof We have to check that $\sigma ; \tau$ is a set of valid plays, all of which have even length, which also contains ε and is even-length prefix-closed.

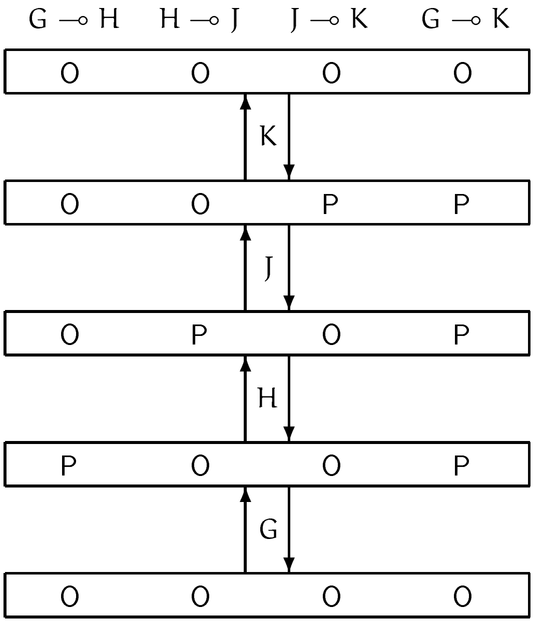
Consider the following state diagram. Note that we have an extra state compared to the analogous diagram for legal interactions. This is due to the switching condition in the linear arrow which prevents Opponent from switching from, say, G to J. It's easy to see from this that any $s \in \sigma ; \tau$ must be legal.



To see that s is a valid play, note that it comes from a witness $u \in \sigma \parallel \tau$ such that $u \restriction G, H \in \sigma$ and $u \restriction H, J \in \tau$. So $s \restriction G = u \restriction G \in P_G$ and similarly $s \restriction J \in P_J$ so that $s \in P_{G \multimap J}$. The other parts follow in the same way as in the HO setting. ∎

Proposition 5.2.2 If $\sigma : G \multimap H$, $\tau : H \multimap J$ and $\upsilon : J \multimap K$ then $(\sigma ; \tau) ; \upsilon = \sigma ; (\tau ; \upsilon)$.

Proof The proof is analogous to the one in the HO setting. It relies on the following state diagram which is a straightforward amendment of the diagram (in chapter 2) for legal interactions of four arenas.



Given $s \in (\sigma ; \tau) ; \upsilon$, we build a witness $w \in (M_G + M_H + M_J + M_K)^\star$ in the usual way. This satisfies $w \restriction G, H \in \sigma$, $w \restriction H, J \in \tau$ and $w \restriction J, K \in \upsilon$. The state diagram now tells us that $w \restriction H, K \in L_{H \multimap K}$ and the rest follows. ∎

Identities As usual, the copycat strategies are the identities for composition: for a game G, we define

$$\mathrm{id}_G = \{s \in P^{\mathrm{even}}_{G \multimap G} \mid \forall s' \sqsubseteq^{\mathrm{even}} s.\ s' \restriction G_\ell = s' \restriction G_r\}.$$

This strategy is always deterministic, history-free and injective.

5.2.2 Robustness of constraints

We should now check that the above-mentioned constraints are closed under composition. We begin with history-freedom.

Proposition 5.2.3 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are both history-free strategies then so is their composite $\sigma\,;\tau$.

Proof Suppose $sab \in \sigma$, $t \in \sigma$ and $ta \in P_G$. There must be a witness $u \in \sigma \parallel \tau$ for $sab \in \sigma\,;\tau$ taking the form $u = u'am_1 \cdots m_k b$ where u' witnesses s and all the m_is are in H. We must also have a witness $v \in \sigma \parallel \tau$ for $t \in \sigma\,;\tau$. Suppose $a \in M_J$ (the other case is analogous). So we have $(u' \restriction H, J)am_1 \in \tau$, $v \restriction H, J \in \tau$ and $(v \restriction H, J)a \in P_{H \multimap J}$. We invoke history-freedom of τ to get $(v \restriction H, J)am_1 \in \tau$. Now we know that $(u' \restriction G, H)m_1 m_2 \in \sigma$, $v \restriction G, H \in \sigma$ and $(v \restriction G, H)m_1 \in P_{G \multimap H}$. This time, history-freedom of σ yields $(v \restriction G, H)m_1 m_2 \in \sigma$... and we iterate this process to build a witness $vam_1 \cdots m_k b \in \sigma \parallel \tau$ for $tab \in \sigma\,;\tau$. ∎

It's clear that determinism is preserved by composition. Another property enjoyed by deterministic strategies is that, if we compose deterministic σ and τ, any play $s \in \sigma\,;\tau$ has a *unique* witness. This is proved by an easy induction on the length of s. This "unique witness" property turns out to be quite useful but is clearly not going to hold for arbitrary nondeterministic strategies. However, it does hold for injective strategies. (Indeed, this is the whole point of injectivity.) But first we must show that injectivity is robust.

Proposition 5.2.4 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are both injective strategies then so is their composite $\sigma\,;\tau$.

Proof If we have sab and $ta'b$ in $\sigma\,;\tau$, we can find witnesses $u = u'am_1 \cdots m_k b$ and $v = v'a'm'_1 \cdots m'_\ell b$ for them. Suppose WLOG that $b \in M_G$. Applying injectivity of σ, we find that $m_k = m'_\ell$. Now we iterate, much as for history-freedom but going backwards, yielding $m_{k-1} = m'_{\ell-1}$... until we're forced to conclude that $k = \ell$ and $a = a'$. ∎

The point of the above proof is that the P-move b uniquely determines the entire *interaction, i.e.* $am_1 \cdots m_k b$, giving rise to it. Armed with this observation, it's an easy induction to show that, for injective σ and τ, if $s \in \sigma\,;\tau$ then s has a unique witness in $\sigma \parallel \tau$.

5.2.3 Ordering strategies

Recall that, in the basic HO setting, we order strategies by subset inclusion. The analogous notion in AJM-games is "subset inclusion up to equivalence". More formally, given strategies σ and τ on some game G, we say that σ ***approximates*** τ, written $\sigma \precsim \tau$, iff

$$s \in \sigma \Rightarrow \exists t \in \tau.\, s \approx_G t.$$

It's easy to see that this is a reflexive and transitive relation. If $\sigma \precsim \tau$ and $\tau \precsim \sigma$, we write $\sigma \sim \tau$ and say that σ and τ are ***equivalent***. This is, by construction, an equivalence relation; we denote by $\langle\sigma\rangle$ the equivalence class containing σ.

Aside In the HO-setting, the assumption of well-threading guarantees that the well-opened representation of a strategy is itself a set of legal plays. In the AJM-setting, this assumption rules out certain pathological strategies, such as

$$\sigma = \{\varepsilon, q_1q_1, q_2q_1, q_1q_1a_1a_1, q_2q_1a_1a_2\}$$

on $X \multimap X$. We have (a_1, a_1) and (a_1, a_2) in $\mathsf{Rel}(\sigma)$, allowing us to build $q_1q_1a_1a_2$ and $q_2q_1a_1a_1$ in $\mathsf{Traces}(\mathsf{Rel}(\sigma))$.

This seems to relate to a mysterious aspect of the interaction between history-freedom and equivalence: it's possible to find σ and τ on some game such that τ is history-free, σ isn't but, nonetheless, $\sigma \sim \tau$. A simple example is:

$$\tau = \{\varepsilon, q_1q_1, q_1q_1a_1a_1\}.$$

It's interesting to note that, in all such examples known to the author, the non-history-free strategy σ violates injectivity. However, we must leave this (somewhat arcane) point open for future work.

5.3 Monotonicity & saturation

If σ and τ are equivalent then we think of them as being "essentially the same". This leads us to consider a categorical framework where objects are games and arrows are equivalence classes of strategies.

Now, we already know that a good notion of composition can be defined which has identities (the copycats) and is associative. All that we additionally need then is for equivalence to be suitably propagated by composition, *i.e.* for appropriately typed σ, σ', τ and τ', if $\sigma \sim \sigma'$ and $\tau \sim \tau'$ then $\sigma ; \tau \sim \sigma' ; \tau'$, *i.e.* $\sim$ is a *congruence*. Better still, we could show that composition is monotone with respect to the approximation ordering: if $\sigma \precsim \sigma'$ and $\tau \precsim \tau'$ then $\sigma ; \tau \precsim \sigma' ; \tau'$.

Composition isn't monotone... Unfortunately, this enterprise soon runs into trouble. Recall the game X that we defined earlier as a "non-standard" version of the flat Boolean game. We can think of the strategies $\{\varepsilon, q_1a_1\}$ and $\{\varepsilon, q_2a_2\}$ as two distinct representations of "true" (these strategies are obviously equivalent) and the strategies $\{\varepsilon, q_1a_3\}$ and $\{\varepsilon, q_2a_3\}$ as the representatives of "false".

If we compose our two representations of "true" (call them t_1 and t_2 respectively, considered as strategies on $\mathbf{1} \multimap X$) with a (partial) "negation" strategy (call it neg) on $X \multimap X$ defined as the set $\{\varepsilon, q_2q_2, q_2q_2a_2a_3\}$, we find that $t_1 \,;\, \mathsf{neg} = \{\varepsilon\}$ but $t_2 \,;\, \mathsf{neg} = \{\varepsilon, q_2a_3\}$. So, although $t_1 \sim t_2$, they behave quite differently in composition—in this case, the problem is that our "negation" strategy only works with the q_2a_2 representation of "true".

A question now arises: is this the fault of the "negation" strategy for not taking account of the alternative representation of "true"; or is it the fault of composition requiring a "strict match" in the middle game? The answer is that we can look at it either way: we may proceed either by slackening composition or by picking out suitable "canonical" choices of strategy for each $\sim$-equivalence class.

5.3.1 Saturated strategies

The reason for the above failure of monotonicity is that a strategy can contain a play s and yet not contain another play s' equivalent to s. In [11], Danos *et al.* introduced the notion of *saturated* strategies: a strategy σ on a game G is ***saturated*** iff

$$(s \in \sigma \wedge s' \approx_G s) \Rightarrow s' \in \sigma.$$

Given any old σ for G, we define its ***saturation***, written $[\sigma]$, to be

$$\{s \in P_G \mid \exists s' \in \sigma.\, s \approx_G s'\}.$$

Clearly σ is saturated if, and only if, $\sigma = [\sigma]$. Is $[\sigma]$ always a well-defined strategy? Well, it's certainly a set of valid plays and it contains ε because σ does. Suppose that $sab \in [\sigma]$. So there must be some $s'a'b' \in \sigma$ such that $sab \approx s'a'b'$. But then $s \approx s'$ by two applications of (**e2**) and, since σ is a strategy, $s' \in \sigma$ and hence $s \in [\sigma]$. In other words, a saturated strategy is obliged to take account of all possible concrete representatives of a given $\approx$-equivalence class of plays—and to do so in a "uniform" fashion.

The class of saturated strategies is readily seen to be robust.

Proposition 5.3.1 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are both saturated strategies then so is their composite $\sigma \,;\, \tau$.

Proof Suppose $s \in \sigma \,;\, \tau$. This must have a witness $u \in \sigma \parallel \tau$. Now, if some $s' \in P_{G \multimap J}$ such that $s \approx_{G \multimap J} s'$ then, by definition, $s \restriction G \approx_G s' \restriction G$ and $s \restriction J \approx_J s' \restriction J$. Using u as a template, we can therefore find $u' \in (M_G + M_H + M_J)^\star$ such that

$u' \restriction G = s' \restriction G$, $u' \restriction J = s' \restriction J'$, $u' \restriction H = u \restriction H$, $u' \restriction G, H \approx_{G \multimap H} u \restriction G, H$ and $u' \restriction H, J \approx_{H \multimap J} u \restriction H, J$. Since σ and τ are both saturated, $u' \restriction G, H \in \sigma$ and $u' \restriction H, J \in \tau$, witnessing $s' \in \sigma ; \tau$. ■

This result offers some evidence that the saturated strategies are good candidates to be concrete representatives of our equivalence classes of strategies.

The next little lemma is trivial to prove but provides us with a useful characterization of the approximation ordering on strategies. It also guarantees that every $\sim$-equivalence class contains exactly one saturated strategy and that composition of saturated strategies is monotone.

Lemma 5.3.2 If σ and τ are strategies for G then $\sigma \precsim \tau$ if, and only if, $[\sigma] \subseteq [\tau]$.

Proof Immediate from the definitions. ■

Aside One way of thinking about saturated strategies is that they operate purely at the level of $\approx$-equivalence classes of plays: we can freely ignore the underlying concrete plays, instead concentrating on the abstract plays. Viewed in this light, abstract plays and saturated strategies closely resemble (HO) legal plays and HO-strategies respectively; see [26] for more in this connection.

5.3.2 Saturated composition

An alternative approach to this problem uses a *blurred* composition, permitting strategies to communicate "up to equivalence" rather than the strict synchronization demanded by the usual definition.

The formal definition is as follows. Consider two strategies $\sigma : G \multimap H$ and $\tau : H \multimap J$. We define $\sigma \parallel\!| \tau$ to be the set of all $u \in (M_G + M_H + M_H + M_J)^\star$ satisfying

$$u \restriction G, H_\ell \in \sigma \wedge u \restriction H_r, J \in \tau \wedge \forall u' \sqsubseteq^{\mathsf{odd}} u.\ u' \restriction H_\ell \approx_H u' \restriction H_r$$

where the ℓ and r tags serve to distinguish the left and right occurences of H and $u' \sqsubseteq^{\mathsf{odd}} u$ means that u' is an odd-length prefix of u. The definition is completed, as is usual, by hiding the middle game.

$$\sigma \fatsemi \tau =_{\mathrm{df}} \{u \restriction G, J \mid u \in \sigma \parallel\!| \tau\}$$

It's not too hard to adapt the proofs of prositions 5.2.1 and 5.2.2 to show that this ***saturated composition*** is well-defined and associative. The usual copycat strategies are identities for saturated composition—up to equivalence.

Proposition 5.3.3 If σ is a strategy on $G \multimap H$ then $\mathsf{id}_G \fatsemi \sigma \sim \sigma \sim \sigma \fatsemi \mathsf{id}_H$.

Proof We consider the case of $\sigma \sim \sigma \fatsemi \mathsf{id}_H$; the other is just the same. First of all, $\sigma \precsim \sigma \fatsemi \mathsf{id}_H$ since $\sigma \subseteq \sigma \fatsemi \mathsf{id}_H$. For the other half, let $s \in \sigma \fatsemi \mathsf{id}_H$. This has a witness $u \in \sigma \parallel\!| \mathsf{id}_H$ and it's easy to see that $s = u \restriction G, H \approx_{G \multimap H} u \restriction G, H_\ell \in \sigma$. ■

So, on the one hand we have the saturated strategies which are closed under all possible permutations of equivalence; and, on the other, we have a saturated composition which allows strategies to communicate up to equivalence. Perhaps unsurprisingly, it turns out that these approaches are precisely equivalent, as is shown by the following result.

Proposition 5.3.4 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ then $\sigma \fatsemi \tau \sim [\sigma] \,;\, [\tau]$.

Proof If $s \in \sigma \fatsemi \tau$ then this has a witness $u \in \sigma \parallel\!| \tau$. By definition, $u \upharpoonright H_r, J \in \tau$ and so $u \upharpoonright H_\ell, J \in [\tau]$. We therefore have $u \upharpoonright G, H_\ell, J \in \sigma \parallel [\tau] \subseteq [\sigma] \parallel [\tau]$, witnessing $s \in [\sigma] \,;\, [\tau]$.

If $s \in [\sigma] \,;\, [\tau]$ then it has a witness $u \in [\sigma] \parallel [\tau]$. Since $u \upharpoonright G, H \in [\sigma]$ and $u \upharpoonright H, J \in [\tau]$, we must have some $s' \in \sigma$ such that $s' \approx_{G \multimap H} u \upharpoonright G, H$ and some $t' \in \tau$ such that $t' \approx_{H \multimap J} u \upharpoonright H, J$. Therefore $s' \upharpoonright H \approx_H u \upharpoonright H \approx_H t' \upharpoonright H$ and, using u as a template, we can find $u' \in \sigma \parallel\!| \tau$ witnessing $t \in \sigma \fatsemi \tau$ such that $s \approx_{G \multimap J} t$. ■

It follows from this that, for any saturated σ on $G \multimap H$, $[\mathsf{id}_G] \,;\, \sigma = \sigma = \sigma \,;\, [\mathsf{id}_H]$, *i.e.* saturated strategies form a category. We can also deduce monotonicity of the saturated composition with respect to $\precsim$ from this: if we have $\sigma, \sigma' : G \multimap H$ and $\tau, \tau' : H \multimap J$ such that $\sigma \precsim \sigma'$ and $\tau \precsim \tau'$ then $\sigma \fatsemi \tau \sim [\sigma] \,;\, [\tau] \subseteq [\sigma'] \,;\, [\tau'] \sim \sigma' \fatsemi \tau'$.

In summary, we can equally restrict to saturated strategies while maintaining the usual notion of composition or we can instead work with $\sim$-equivalence classes of strategies, defining composition by $\langle \sigma \rangle \fatsemi \langle \tau \rangle = \langle \sigma \fatsemi \tau \rangle$.

5.3.3 Constraints

Earlier on, we introduced three constraints that can be applied to AJM-strategies: determinism, history-freedom and injectivity. How do these properties interact with saturation? Well, there's no chance that many saturated strategies are going to be deterministic. For a simple example, consider $\mathsf{id}_X : X \multimap X$. This contains, amongst other things, the traces $\mathsf{q}_1\mathsf{q}_1$ and $\mathsf{q}_2\mathsf{q}_2$. According to the definition of $\approx_{X \multimap X}$, we have $\mathsf{q}_1\mathsf{q}_1 \approx \mathsf{q}_1\mathsf{q}_2$ so that $[\mathsf{id}_X]$ manifestly violates determinism.

Determinism up to equivalence Although we've seen that $[\mathsf{id}_X]$ violates determinism, the manner of this violation doesn't seem overly pathological—after all, the two plays mentioned above *are* equivalent. This leads us to consider whether there might be an alternative formulation of determinism more suitable for the saturated setting.

The obvious idea is to say that, if $sa, s'a' \in \mathsf{dom}(\sigma)$ and $sa \approx s'a'$ then all of σ's responses to sa and $s'a'$ must be equivalent to one another. Formally, we say that a strategy σ is ***deterministic up to equivalence*** (or just ***deteq***) iff

$$(sab \in \sigma \wedge s'a'b' \in \sigma \wedge sa \approx s'a') \Rightarrow sab \approx s'a'b'.$$

We'll have more to say about this condition later on. For the time being, we simply note that the class of deteq strategies is closed under composition.

Proposition 5.3.5 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are both deteq then so is $\sigma ; \tau$.

Proof First of all, consider any $s_1, s_2 \in \sigma ; \tau$ such that $s_1 \approx s_2$ and let $u_1, u_2 \in \sigma \parallel \tau$ witness s_1 and s_2 respectively. We claim that $u_1 \upharpoonright G, H \approx u_2 \upharpoonright G, H$ and $u_1 \upharpoonright H, J \approx u_2 \upharpoonright H, J$. The proof is by induction on $|u_1|$; the base case, where $u_1 = \varepsilon = u_2$ is trivial. Otherwise, we have $u_1 = u_1' a m_1 \cdots m_k b$ and $u_2 = u_2' a' m_1' \cdots m_\ell' b'$ witnessing $s_1 = s_1' ab$ and $s_2 = s_2' a' b'$. Since $s_1' \approx s_2'$, we have $u_1' \upharpoonright G, H \approx u_2' \upharpoonright G, H$ and $u_1' \upharpoonright H, J \approx u_2' \upharpoonright H, J$. Clearly $u_1' a \upharpoonright G, H \approx u_2' a' \upharpoonright G, H$ and $u_1' a \upharpoonright H, J \approx u_2' a' \upharpoonright H, J$ and applying deteq alternately in σ and τ now yields the desired result. In the case at hand, suppose $sab, s'a'b' \in \sigma ; \tau$ such that $sa \approx s'a'$. We apply the above to show that any witnesses of s and s' are "equivalent" in this sense and the same sort of reasoning easily establishes that $sab \approx s'a'b'$. ■

History-freedom & injectivity? The interaction of these two constraints with saturation is rather more problematic: rather as for determinism, we find that very few saturated strategies are history-free or injective (again, $[\mathsf{id}_X]$ is a good example); but the situation is further complicated by the fact that both of these constraints, *i.e.*

$$\begin{aligned} sab, t \in \sigma \wedge ta \in P_G &\Rightarrow tab \in \sigma \\ sab, ta'b \in \sigma &\Rightarrow a = a', \end{aligned}$$

involve plays s and t that are entirely independent of one another. This makes it unclear how to express any reasonable notion of "history-free up to equivalence" or "injective up to equivalence".

This problem is perhaps not so surprising when we remember that determinism is a *coherence* condition, that is to say, it's concerned with a certain "uniformity" which makes equal sense at the level of concrete or of abstract plays. History-freedom and injectivity, on the other hand, are both intrinsically concerned with particular occurences of moves and, as such, cannot obviously be expressed at the abstract level.

Fortunately, there is an alternative to saturation. In the original work on AJM-games [1], attention was restricted to a class of "self-equivalent" strategies. It turns out that composition of such strategies *is* monotone with respect to the approximation ordering. Unfortunately, this property—rather like innocence—crucially depends on determinism! After a brief detour to introduce the bang constructor on games, we'll examine the nature of self-equivalence. This eventually leads us to a more general conception of strategy and self-equivalence which works without any assumption of determinism.

5.4 Bang!

It's well-known that the forbidden operations of weakening and contraction are reintroduced to linear logic, in annotated form, via the use of the exponential(s) 'bang' (and 'why not'). In terms of games, bang is a constructor allowing us to "play the game" as many times as we like. (We've already noted the "affine" nature of games; the rôle of bang here is really just to reintroduce contraction.) More formally, given a game G, we define $!G$ (read "bang (of) G") by:

- $O_{!G} = \mathbf{N} \times O_G$ and $P_{!G} = \mathbf{N} \times P_G$. Given a move $m = (i, m')$, we define projections $\mathsf{index}(m) = i$ and $\mathsf{move}(m) = m'$.

  Given $s \in L_{!G}$ and $i \in \mathbf{N}$, define $s \upharpoonright i$ to be the subsequence of moves m such that $\mathsf{index}(m) = i$.

- $s \approx_{!G} t$ iff there exists a permutation π on $\mathbf{N}_0$ such that, for all $i \in \mathbf{N}_0$,

  $$s \upharpoonright i \approx_G t \upharpoonright \pi(i),$$

  i.e. each "thread" of play in s has an equivalent counterpart in t and, moreover, these threads are interleaved in the "same way":

  $$\pi(\mathsf{index}(s_j)) = \mathsf{index}(t_j), \text{ for } 1 \leq j \leq |s|.$$

What this definition says is that $!G$ allows us to play countably many "copies" of G, interleaved in any way that obeys the global axioms on plays. Different copies are distinguished by the natural number tags on moves and two plays should be considered equivalent if they are the same—upto these choices of indices. This can be thought of, intuitively, as an "infinite symmetric tensor" and, indeed, there is a switching condition analogous to that of the tensor above: only Opponent may move between different copies of G, *i.e.* change index.

Dereliction For any game G and natural number i, we have a (deterministic) history-free strategy $\mathsf{d}_G^i : !G \multimap G$ which copycats between the RHS and thread i of the LHS:

$$\begin{array}{rcl} \mathsf{inr}(m) & \mapsto & \mathsf{inl}(i, m) \\ \mathsf{inl}(i, m) & \mapsto & m. \end{array}$$

Clearly, for any natural numbers i and j, we have $\mathsf{d}_G^i \sim \mathsf{d}_G^j$.

Weakening For any game G, we have a unique strategy $!_G : G \multimap \mathbf{1}$, *i.e.* $\{\varepsilon\}$. In other words, the tensor unit is terminal and our category is "affine". We therefore trivially have weakening arrows: for a game G, we set $\mathsf{w}_G = !_{!G}$.

Contraction Unlike the situation with HO-games, we don't have diagonal arrows for all objects. However, we can define diagonals for games of the form $!G$. First of all, let $t : \mathbf{N}_0 + \mathbf{N}_0 \rightarrowtail \mathbf{N}_0$ be an injective *tagging* function, ***e.g.*** $t(\mathsf{inl}(n)) = 2n$, $t(\mathsf{inr}(n)) = 2n+1$. For a game G, we define a (deterministic) history-free strategy $c_G : !G \multimap !G \otimes !G$ by:

$$\begin{aligned} \mathsf{inr}(\mathsf{inl}(i, m)) &\mapsto \mathsf{inl}(t(\mathsf{inl}(i)), m) \\ \mathsf{inr}(\mathsf{inr}(i, m)) &\mapsto \mathsf{inl}(t(\mathsf{inr}(i)), m) \\ \mathsf{inl}(i, m) &\mapsto \mathsf{inr}(\mathsf{inl}(j, m)), \text{ if } t(\mathsf{inl}(j)) = i \\ \mathsf{inl}(i, m) &\mapsto \mathsf{inr}(\mathsf{inr}(j, m)), \text{ if } t(\mathsf{inr}(j)) = i. \end{aligned}$$

This (rather hideous-looking) definition simply says that we interleave the play in the two RHS copies of $!G$ into the LHS copy, using the tagging function t to keep track of "where we are and where we're going". The particular choice of tagging function is unimportant: for two tagging functions t and t', we have $c_G^t \sim c_G^{t'}$.

5.5 Self-equivalence: the deterministic case

5.5.1 Self-equivalence, robustness & monotonicity

The idea of self-equivalence is that it's a kind of "weak saturation" property which insists that, if a strategy σ responds with b to some $sa \in \mathsf{dom}(\sigma)$, it must also have an "equivalent response" b' to any $ta' \in \mathsf{dom}(\sigma)$ such that $sa \approx ta'$. In other words, the strategy is saturated from Opponent's point of view but not from Player's. This idea is expressed formally in the following definition.

A deterministic strategy σ on a game G is ***self-equivalent*** iff

$$(sab \in \sigma \wedge t \in \sigma \wedge sa \approx_G ta') \Rightarrow \exists!\, ta'b' \in \sigma.\ sab \approx_G ta'b'.$$

We have two major obligations to fulfil: to demonstrate that the class of self-equivalent strategies is closed under composition; and to show that composition (of self-equivalent strategies) respects the approximation ordering.

Robustness The next lemma is the vital link for the first of these. Its proof is reminiscent of that of the robustness of deteq; the reasons for this should become clearer a little later. The reliance on determinism also recalls the similar lemma in the HO-setting used to establish the robustness of innocence.

Lemma 5.5.1 If $s, t \in \sigma ; \tau$ for deterministic, self-equivalent strategies $\sigma : G \multimap H$ and $\tau : H \multimap J$ with (necessarily unique) witnesses $u, v \in \sigma \parallel \tau$ then, if $s \approx t$, $u \restriction G, H \approx v \restriction G, H$ and $u \restriction H, J \approx v \restriction H, J$.

Proof We go by induction on the length of s. The base case, as is usually the case, is completely trivial. Otherwise, suppose we have $s = s'ab$ and $t = t'a'b'$ with witnesses $u = u'am_1 \cdots m_k b$ and $v = v'a'm'_1 \cdots m'_\ell b'$. By the inductive hypothesis, $u' \restriction G, H \approx v' \restriction G, H$ and $u' \restriction H, J \approx v' \restriction H, J$. WLOG, suppose that $a \in M_J$. We know that $sa \approx ta'$, so $u'a \restriction H, J \approx v'a' \restriction H, J$. Now we invoke self-equivalence of τ, yielding $(v' \restriction H, J)a'm''_1 \in \tau$ such that $(u' \restriction H, J)am_1 \approx (v' \restriction H, J)a'm''_1$. But, since τ is also deterministic, $m'_1 = m''_1$. Proceeding in this way, we eventually establish that $u \restriction G, H \approx v \restriction G, H$ and $u \restriction H, J \approx v \restriction H, J$ as desired. ■

Proposition 5.5.2 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are deterministic, self-equivalent strategies then so is their composite $\sigma ; \tau$.

Proof We already know that $\sigma ; \tau$ is deterministic. So, suppose that $sab, t \in \sigma ; \tau$ such that $sa \approx ta'$; let $uam_1 \cdots m_k b$ witness sab and v witness t. By the above lemma, $u \restriction G, H \approx v \restriction G, H$ and $u \restriction H, J \approx v \restriction H, J$. WLOG, suppose $a \in M_J$ so, by self-equivalence of τ, we have (a unique) $(v \restriction H, J)a'm'_1 \in \tau$ such that $(u \restriction H, J)am_1 \approx (v \restriction H, J)a'm'_1$. Hence $(u \restriction G, H)m_1 \approx (v \restriction G, H)m'_1$ and, by self-equivalence of σ, we have (a unique) $(v \restriction G, H)m'_1 m'_2 \in \sigma$ such that $(u \restriction G, H)m_1 m_2 \approx (v \restriction G, H)m'_1 m'_2$. Hence, $(u \restriction H, J)am_1 m_2 \approx (v \restriction H, J)a'm'_1 m'_2$ and, by self-equivalence of τ... ■

Monotonicity First of all, recall the definition of $\sigma \mathbin{|||} \tau$ from the earlier subsection on saturated composition. If $u \in \sigma \mathbin{|||} \tau$, we say that it's a *lax witness*. If we have self-equivalent strategies $\sigma, \sigma' : G \multimap H$ and $\tau, \tau' : H \multimap J$ where $\sigma \precsim \sigma'$ and $\tau \precsim \tau'$, we want, for any $u \in \sigma \parallel \tau$, some "equivalent" witness in $\sigma' \parallel \tau'$.

Lemma 5.5.3 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are deterministic, self-equivalent strategies and $u \in \sigma \mathbin{|||} \tau$, *i.e.* it's a lax witness, then there's some $v \in \sigma \parallel \tau$ such that $u \restriction G, H_\ell \approx v \restriction G, H$ and $u \restriction H_r, J \approx v \restriction H, J$.

Proof We go by induction on the length of u. The base case is trivial. Otherwise, suppose that $u = u'am_1 n_1 \cdots b$ where $a \in O_{G \multimap J}$. So $u' \in \sigma \mathbin{|||} \tau$ and, by the inductive hypothesis, we have some $v' \in \sigma \parallel \tau$ such that $u' \restriction G, H_\ell \approx v' \restriction G, H$ and $u' \restriction H_r, J \approx v' \restriction H, J$. Consider the case where a is a move of J and b is a move of G so we have $u = u'am_1 n_1 n_2 m_2 \cdots m_k n_k b$, where $u' \restriction H_r \cdot m_1 \cdots m_k \approx u' \restriction H_\ell \cdot n_1 \cdots n_k$. So, we know that $(u' \restriction H_r, J)am_1 \in \tau$, $v' \restriction H, J \in \tau$ and, by (**e3**), we have $(u' \restriction H_r, J)a \approx (v' \restriction H, J)a'$ for some a'. Applying self-equivalence of τ, we find some $(v' \restriction H, J)a'm'_1 \in \tau$ such that $(v' \restriction H, J)a'm'_1 \approx (u' \restriction H_r, J)am_1$. Since $u' \restriction H_r \cdot m_1 \approx u' \restriction H_\ell \cdot n_1$, we have $(v' \restriction G, H)m'_1 \approx (u' \restriction G, H_\ell)n_1$. So, by self-equivalence of σ, we can find $(v' \restriction G, H)m'_1 m'_2 \in \sigma$ such that $(v' \restriction G, H)m'_1 m'_2 \approx (u' \restriction G, H_\ell)n_1 n_2$ and we can iterate this process until we've built an appropriate $v'a'm'_1 \cdots m'_k b' \in \sigma \parallel \tau$. ■

Proposition 5.5.4 If we have self-equivalent strategies $\sigma, \sigma' : G \multimap H$ and $\tau, \tau' : H \multimap J$ where $\sigma \sqsubseteq_{\sim} \sigma'$ and $\tau \sqsubseteq_{\sim} \tau'$ then $\sigma ; \tau \sqsubseteq_{\sim} \sigma' ; \tau'$.

Proof If $s \in \sigma ; \tau$, it has some witness $u \in \sigma \parallel \tau$. Applying $\sigma \sqsubseteq_{\sim} \sigma'$ and $\tau \sqsubseteq_{\sim} \tau'$, we build from u some $v \in \sigma' \parallel\!| \tau'$ such that $u \restriction G, J \approx v \restriction G, J$. By the previous lemma, we have some $v' \in \sigma' \parallel \tau$ such that $s \approx v \restriction G, J \approx v' \restriction G, J \in \sigma' ; \tau'$. ■

In other words, composition up to equivalence is no "weaker" than the usual strict composition for self-equivalent strategies.

A basic category We can put all this together to build a category where objects are games and an arrow from G to H is a $\sim$-equivalence class of self-equivalent strategies for $G \multimap H$. We define composition by: $\langle\sigma\rangle ; \langle\tau\rangle = \langle\sigma ; \tau\rangle$. This is well-defined by propositions 5.2.1, 5.5.2 and 5.5.4; it's associative by proposition 5.2.2; and has copycats as identities as usual. We denote this category by $\mathcal{G}$.

The $\otimes$ and $\multimap$ operations on games lift up, analogously to $\times$ and $\Rightarrow$ in the HO-setting, to bifunctors on this category. Furthermore, the evident analogues of the (symmetric) monoidal isomorphisms also exist here, so $\mathcal{G}$ is an SMCC.

5.5.2 Historical aside

The above presentation of strategies and self-equivalence substantially differs from the original formulation of [1]. In that work, the fundamental definition was of the *simulation* ordering: we say that σ ***simulates*** τ, written $\sigma \sqsubseteq_{\approx} \tau$, iff

$$(sab \in \sigma \wedge t \in \tau \wedge sa \approx ta') \Rightarrow \exists ta'b' \in \tau.\ sab \approx ta'b'.$$

Modulo the uniqueness requirement, $\sigma \sqsubseteq_{\approx} \sigma$ if, and only if, σ is self-equivalent.

In fact, the uniqueness requirement is intimately tied in with the issue of determinism. The typical kind of strategy that approximates itself in the simulation ordering but which isn't self-equivalent is $d^2_G \cup d^3_G$, *i.e.* it has "pointless" (deteq) nondeterminism. If we rule out these kinds of things, we have:

Lemma 5.5.5 A self-equivalent strategy is deterministic if, and only if, it's deteq.

Proof If σ is deteq and $sab, sac \in \sigma$ then $sab \approx sac$ and, by self-equivalence, we have $b = c$. Conversely, if $sab, s'a'b' \in \sigma$ and $sa \approx s'a'$ then, by self-equivalence, there's a unique $s'a'b'' \in \sigma$ equivalent to sab. But, by determinism, $b' = b''$. ■

Better still, for self-equivalent strategies, the approximation ordering coincides with the simulation ordering so it makes more sense to take self-equivalence as the fundamental notion and use the simpler ordering.

Lemma 5.5.6 If $\sigma \sqsubseteq_{\approx} \tau$ then $\sigma \sqsubseteq_{\sim} \tau$. If τ is self-equivalent and $\sigma \sqsubseteq_{\sim} \tau$ then $\sigma \sqsubseteq_{\approx} \tau$.

Proof For the first part, we induct on the length of $s \in \sigma$; the base case is trivial and, for $sab \in \sigma$, the inductive hypothesis guarantees $t \in \tau$ equivalent to s. Axiom (**e3**) provides ta' equivalent to sa and the rest follows from $\sigma \mathrel{\underset{\approx}{\sqsubseteq}} \tau$. For the "converse", if $sab \in \sigma$, $t \in \tau$ and $sa \approx ta'$ then, by $\sigma \mathrel{\underset{\sim}{\sqsubseteq}} \tau$, we have $t'cd \in \tau$ equivalent to sab and self-equivalence of τ gives us $ta'b' \in \tau$ such that $ta'b' \approx t'cd \approx sab$. ■

5.5.3 Bang as a comonad

Let $p : \mathbf{N}_0 \times \mathbf{N}_0 \rightarrowtail \mathbf{N}_0$ be an injective *pairing* function. For each $i \in \mathbf{N}_0$, we write p_i for the (injective) function from $\mathbf{N}_0$ to itself defined by fixing the first argument of p as i, *i.e.* by currying in **Set** and applying to i. We write $\mathsf{rng}(p_i)$ for the range of p_i; if $i \neq j$ then the ranges of p_i and p_j are obviously disjoint.

Given a self-equivalent strategy $\sigma : !G \multimap H$, we define its ***promotion*** with respect to pairing function p, written $\sigma^!_p : !G \multimap !H$, by:

$$\sigma^!_p = \{s \in P_{!G \multimap !H} \mid \forall i \in \mathbf{N}_0.\ s \restriction i \in \sigma\},$$

where $s \restriction i \in P_{!G \multimap H}$ is defined inductively as:

$$\begin{aligned}
\varepsilon \restriction i &= \varepsilon \\
s \cdot \mathsf{inl}(a) \restriction i &= s \restriction i, && \text{if } \mathsf{index}(a) \notin \mathsf{rng}(p_i) \\
s \cdot \mathsf{inl}(a) \restriction i &= (s \restriction i) \cdot (j, \mathsf{move}(a)), && \text{if } \mathsf{index}(a) = p_i(j) \\
s \cdot \mathsf{inr}(a) \restriction i &= s \restriction i, && \text{if } \mathsf{index}(a) \neq i \\
s \cdot \mathsf{inr}(a) \restriction i &= (s \restriction i) \cdot a, && \text{if } \mathsf{index}(a) = i.
\end{aligned}$$

In other words, it's the ith thread of the RHS and the "associated" threads, that is to say $\mathsf{rng}(p_i)$, of the LHS. It can be shown that the particular choice of pairing function is irrelevant; if p and p' are pairing functions, $\sigma^!_p \sim \sigma^!_{p'}$.

It's not too hard to see that, for $\sigma : !G \multimap H$ and $\tau : !H \multimap J$, we have:

$$\begin{aligned}
\sigma^! ; d_H &\sim \sigma \\
{d_G}^! ; \sigma &\sim \sigma \\
\sigma^! ; \tau^! &\sim (\sigma^! ; \tau)^!.
\end{aligned}$$

This means that bang extends to be a comonad on $\mathcal{G}$. We could take this much further, showing that every object of the form $!G$ has a cocommutative comonoid structure associated with it, defining $G \mathbin{\&} H$, the "with" of linear logic, showing the Seely isomorphism $!(G \mathbin{\&} H) \cong !G \otimes !H$ and hence that the Kleisli category for bang is Cartesian closed. While obviously important, none of this is particularly pertinent to our interests here; we shall content ourselves with one or two brief remarks before turning to the question of nondeterministic self-equivalence.

Some AJM arcana It's interesting to note the difference between the apparently similar constraints of self-equivalence (in AJM-games) and single-threading (in HO-games). Whereas single-threading rules out behaviours such as the "counter" and the "one-off" strategies on (the arena) $\mathbf{N}$, we find that the AJM analogues to these *are* self-equivalent strategies for (the game) $!\mathbf{N}$. Since such strategies *are* single-threaded for the lifted arena $\mathbf{N}_\perp$, this is suggestive of a connection between lifting and bang. And indeed, a natural definition of lifting in AJM-games is:

$$\mathsf{G}_\perp = (!\mathsf{G})^{\perp\perp}.$$

In other words, we can get the extra single-threaded behaviours bestowed by lifting in HO-games just by using bang in AJM-games. This means that not every strategy $\sigma : !\mathsf{G} \multimap !\mathsf{H}$ is the promotion of something. (Indeed, neither the counter nor a one-off strategy is.) However, this property *is* true for history-free σ; this is the vital *Bang lemma* of [1].

The second point concerns the nature of flat games. To keep things simple, let's consider the Booleans. The usual flat game $\mathbf{B}$ can be written as $(\perp \otimes \perp) \multimap \perp$. Given this, it might make sense to consider a similarly "internalized" form of the game when we use the Kleisli category for bang. This suggests the alternative Boolean game $(!\perp \otimes !\perp) \multimap \perp$. The ($\sim$-equivalence classes of) strategies for this game form the usual "flat domain" when ordered by $\precsim$. The utility of this alternate game is that it allows us to model PCF, including its conditional branchings, using only injective strategies. This is of vital importance in the next section.

5.6 Self-equivalence: the nondeterministic case

Preamble Recall the discussion of nondeterministic innocent strategies in §3.7 where we identified insufficient "branching time" information as the reason why they fail to be closed under composition. Consider the following strategy:

$$\begin{array}{ccccc}
(!\mathbf{B} & \otimes & !\mathbf{B}) & \multimap & \mathbf{B} \\
 & & & & \mathsf{q} \\
0\mathsf{q} & & & & \\
0\mathsf{tt} & & & & \\
 & & & & \mathsf{tt} \\
\hline
 & & & & \mathsf{q} \\
 & & 0\mathsf{q} & & \\
 & & 0\mathsf{tt} & & \\
 & & & & \mathsf{ff}
\end{array}$$

This strategy is obviously nondeterministic, but it does satisfy the liveness condition for self-equivalence. However, if we compose it with the contraction strategy for $\mathbf{B}$, we get the same problem as with nondeterministic innocence.

The following interactions are typical of this composition. Just as with innocence, problems arise when two equivalent plays in the composite—in this case, $\mathsf{q} \cdot 0\mathsf{q}$ and $\mathsf{q} \cdot 1\mathsf{q}$—have “inequivalent” witnesses.

$$
\begin{array}{ccccccc}
!\mathbf{B} & \multimap & !\mathbf{B} & \otimes & !\mathbf{B} & \multimap & !\mathbf{B} \\
 & & & & & & \mathsf{q} \\
 & & 0\mathsf{q} & & & & \\
0\mathsf{q} & & & & & & \\
0\mathsf{tt} & & & & & & \\
 & & 0\mathsf{tt} & & & & \\
 & & & & & & \mathsf{tt} \\
\hline
 & & & & & & \mathsf{q} \\
 & & & & 0\mathsf{q} & & \\
1\mathsf{q} & & & & & & \\
1\mathsf{tt} & & & & & & \\
 & & & & 0\mathsf{tt} & & \\
 & & & & & & \mathsf{ff}
\end{array}
$$

The resulting composite strategy isn't self-equivalent. In order to fix this, what we need is some way of saying that, although $\mathsf{q} \cdot 0\mathsf{q}$ and $\mathsf{q} \cdot 1\mathsf{q}$ are equivalent in the underlying game, they *aren't* equivalent in this strategy because they have “inequivalent” witnesses. Once we have this, we can weaken self-equivalence in such a way that we no longer rely on determinism.

5.6.1 Strategies & composition

The basic idea is to add to a strategy an equivalence relation saying which plays of the strategy are *really* equivalent, as far as that strategy is concerned.

Definition A ***strategy*** σ for a game G is a couple $(T_\sigma, \approx_\sigma)$ where T_σ is a strategy in the usual sense, with the additional requirement of injectivity:

(**t3**) if $sab \in T_\sigma$ and $ta'b \in T_\sigma$ then $a = a'$;

and $\approx_\sigma$ is an equivalence relation on T_σ satisfying

(**r1**) if $s \approx_\sigma t$ then $s \approx_G t$;

(**r2**) if $sab \approx_\sigma ta'b'$ then $s \approx_\sigma t$.

If $s \approx_\sigma t$ then we say that s and t are ***really equivalent*** in σ. Axiom (**r1**) guarantees that $\approx_\sigma$ only *refines* the equivalence of the underlying game G, *i.e.* it makes no new identifications. Axiom (**r2**) is a straightforward sanity condition.

Self-equivalence & composition A strategy σ for a game G is ***self-equivalent*** iff

$$(sab \in T_\sigma \wedge s \approx_\sigma t \wedge sa \approx_G ta') \Rightarrow \exists!\, b' \in M_G.\ sab \approx_\sigma ta'b'.$$

This is the key definition. The extra condition—namely that $s \approx_\sigma t$—means that we only need the matching $ta'b'$ in σ if s and t are "in the same branch", *i.e.* they're really equivalent in σ.

Since we've imposed injectivity as an additional constraint on trace sets, if we define composition of trace sets in the usual way, we'll have a unique witness property. So, given $\sigma : G \multimap H$ and $\tau : H \multimap J$, we define $T_{\sigma;\tau}$ to be $T_\sigma ; T_\tau$. If $s \in T_{\sigma;\tau}$, we write $\mathsf{wit}(s)$ for the (necessarily unique) witness of s in $T_\sigma \parallel T_\tau$. We can now define $\approx_{\sigma;\tau}$ by:

$$s \approx_{\sigma;\tau} t \ \text{ iff } \ \mathsf{wit}(s) \upharpoonright G, H \approx_\sigma \mathsf{wit}(t) \upharpoonright G, H \ \wedge \ \mathsf{wit}(s) \upharpoonright H, J \approx_\tau \mathsf{wit}(t) \upharpoonright H, J.$$

This captures the idea that "really equivalent plays in the composite should come from really equivalent witnesses". In particular, note that, in the above example, the equivalent plays $q \cdot 0q$ and $q \cdot 1q$ cannot be really equivalent in the composite strategy (since their witnesses are not even equivalent).

Proposition 5.6.1 If $\sigma : G \multimap H$ and $\tau : H \multimap J$ are self-equivalent strategies then so is $\sigma ; \tau : G \multimap H$.

Proof It's clear that $T_{\sigma;\tau}$ is a valid trace set. It's also easy to see that $\approx_{\sigma;\tau}$ is an equivalence relation on $T_{\sigma;\tau}$ and that it respects (**r1**). For (**r2**), if $sab \approx_{\sigma;\tau} ta'b'$ then $s = \mathsf{wit}(sab)_{<a} \upharpoonright G, J$ and, similarly, $t = \mathsf{wit}(ta'b')_{<a'} \upharpoonright G, J$ so that $s \approx_{\sigma;\tau} t$. For self-equivalence, suppose that $sab \in T_{\sigma;\tau}$, $s \approx_{\sigma;\tau} t$ and $sa \approx_{G \multimap J} ta'$. WLOG, suppose that $a \in M_J$. Since $s \approx_{\sigma;\tau} t$, we have that $\mathsf{wit}(s) \upharpoonright H, J \approx_\tau \mathsf{wit}(t) \upharpoonright H, J$. Consider $\mathsf{wit}(sab)$. It must have the form $\mathsf{wit}(s) \cdot am_1 \cdots m_k b$. We now apply self-equivalence of τ, yielding a unique m_1' such that $(\mathsf{wit}(s) \upharpoonright H, J) \cdot am_1 \approx_\tau (\mathsf{wit}(t) \upharpoonright H, J) \cdot a'm_1'$. We therefore have $(\mathsf{wit}(s) \upharpoonright G, H) \cdot a \approx_{G \multimap H} (\mathsf{wit}(t) \upharpoonright G, H) \cdot a'$. Proceeding in this way, we build a unique witness $\mathsf{wit}(ta'b') = \mathsf{wit}(t) \cdot a'm_1' \cdots m_k' b'$ such that $\mathsf{wit}(sab) \upharpoonright G, H \approx_\sigma \mathsf{wit}(ta'b') \upharpoonright G, H$ and $\mathsf{wit}(sab) \upharpoonright H, J \approx_\tau \mathsf{wit}(ta'b') \upharpoonright H, J$. Hence $sab \approx_{\sigma;\tau} ta'b'$. ■

At this point, it's worth noting that, if σ is self-equivalent and deterministic (*i.e.* T_σ is deterministic) then $\approx_\sigma$ makes no more distinctions than the equivalence in the underlying game.

Lemma 5.6.2 Let σ be a self-equivalent strategy for the game G such that T_σ is deterministic. If $s, t \in T_\sigma$ and $s \approx_G t$ then $s \approx_\sigma t$.

Proof We induct on the length of s; the base case is trivial. Let $s = s'ab$ and $t = t'a'b'$. So $s' \approx_G t'$ and, by the inductive hypothesis, $s' \approx_\sigma t'$. By self-equivalence, we have a unique b'' such that $sab \approx_\sigma ta'b''$ and, by determinism, $b' = b''$. ■

5.6.2 A basic category of games & self-equivalent strategies

We now build a category where the objects are games and an arrow from G to H is a self-equivalent strategy for $G \multimap H$. We already know that composition is well-defined and that, if $\sigma : G \multimap H$, $\tau : H \multimap J$ and $\upsilon : J \multimap K$ are (self-equivalent) strategies, then $T_{(\sigma;\tau);\upsilon} = T_{\sigma;(\tau;\upsilon)}$. It's easy to see that

$$s \approx_{(\sigma;\tau);\upsilon} t \iff s \approx_{\sigma;(\tau;\upsilon)} t$$

so that composition is associative. For each game G, the strategy Id_G defined as $(\mathsf{id}_G, \approx_G)$ acts as the identity arrow.

A crude ordering Given two (self-equivalent) strategies, σ and τ, for the game G, we define:

$$\sigma \sqsubseteq_G \tau \text{ iff } s \approx_\sigma t \Rightarrow s \approx_\tau t.$$

Clearly, if $\sigma \sqsubseteq_G \tau$ then $T_\sigma \subseteq T_\tau$. It's easy to see that $\sqsubseteq_G$ defines a partial ordering on the set of all strategies for G. It also has the following useful property.

Lemma 5.6.3 If σ and τ are self-equivalent strategies for the game G such that $\sigma \sqsubseteq \tau$ then, if $s \approx_\tau t$ and $s \in T_\sigma$ then $s \approx_\sigma t$.

Proof By induction on $|s|$; the base case is trivial. Let $s = s'ab$ and $t = t'a'b'$. By (**r2**), we have $s' \approx_\tau t'$ and, since $s' \in T_\sigma$, applying the inductive hypothesis yields $s' \approx_\sigma t'$. Since σ is self-equivalent, we have a unique b'' such that $sab \approx_\sigma ta'b''$. But then $sab \approx_\tau ta'b''$ and, by self-equivalence of τ, $b' = b''$. ∎

Roughly speaking, this lemma says that each block (with respect to $\approx_\sigma$) in σ is unchanged in τ: any play of τ that's not also a play of σ must be in a block unique to τ. Armed with this, we can characterize $\sqsubseteq_G$ with a (far more complex!) condition strongly reminiscent of the simulation ordering from [1].

Lemma 5.6.4 If σ and τ are self-equivalent strategies for the game G then $\sigma \sqsubseteq_G \tau$ if, and only if,

$$(sab \in T_\sigma \wedge s \approx_\tau t \wedge sa \approx_G ta') \Rightarrow \exists!\, b'.\ sab \approx_\tau ta'b'.$$

Proof $(\Rightarrow)$. Suppose that $sab \in T_\sigma$, $s \approx_\tau t$ and $sa \approx_G ta'$. By the previous lemma, we have $s \approx_\sigma t$ and applying self-equivalence in σ gives a unique b' such that $sab \approx_\sigma ta'b'$. Since $\sigma \sqsubseteq_G \tau$, we have $sab \approx_\tau ta'b'$ and, by the previous lemma and the fact that $sab \in T_\sigma$, it must be unique in τ.

$(\Leftarrow)$. This goes by induction on the length of s; the base case is trivial. So, let $s = s'ab$ and $t = t'a'b'$ and suppose that $s \approx_\sigma t$. By (**r2**), we have $s' \approx_\sigma t'$ and applying the inductive hypothesis gives $s' \approx_\tau t'$. We now invoke the "simulation" condition, yielding a unique b'' such that $s'ab \approx_\tau ta'b''$. Applying the previous lemma, we get $s'ab \approx_\sigma ta'b''$ and, by self-equivalence of σ, we have $b' = b''$. ∎

Algebraicity Given a $\sqsubseteq_G$-directed set Δ of self-equivalent strategies, we define:

$$\bigsqcup \Delta = (\bigcup_{\sigma \in \Delta} \sigma, \bigcup_{\sigma \in \Delta} \approx_\sigma).$$

Lemma 5.6.5 $\bigsqcup \Delta$ is a self-equivalent strategy and is the least upper bound of Δ.

Proof It's easy to see that it's a well-defined strategy. For self-equivalence, let $\mathtt{sab} \in T_{\bigsqcup \Delta}$, $\mathtt{s} \approx_{\bigsqcup \Delta} \mathtt{t}$ and $\mathtt{sa} \approx_G \mathtt{ta'}$. By definition, we have some $\sigma_1 \in \Delta$ such that $\mathtt{s} \approx_{\sigma_1} \mathtt{t}$ and some $\sigma_2 \in \Delta$ such that $\mathtt{sab} \in T_{\sigma_2}$. By directedness, we have an upper bound for these, σ_3, and by self-equivalence, there's a unique $\mathtt{b'}$ such that $\mathtt{sab} \approx_{\sigma_3} \mathtt{ta'b'}$. Therefore $\mathtt{sab} \approx_{\bigsqcup \Delta} \mathtt{ta'b'}$. If $\mathtt{sab} \approx_{\bigsqcup \Delta} \mathtt{ta'b''}$ then we can find some $\sigma_4 \in \Delta$ that caused this and, by directedness again, we have $\sigma_5 \in \Delta$ such that $\sigma_3 \sqsubseteq_G \sigma_5$ and $\sigma_4 \sqsubseteq_G \sigma_5$. Since σ_5 is self-equivalent, we have $\mathtt{b'} = \mathtt{b''}$. So $\bigsqcup \Delta$ is self-equivalent. It's easy to see that $\bigsqcup \Delta$ is an upper bound of Δ. To see that it's the least upper bound, let τ be any upper bound of Δ and let $\mathtt{s} \approx_{\bigsqcup \Delta} \mathtt{t}$. By definition, we have some $\sigma \in \Delta$ such that $\mathtt{s} \approx_\sigma \mathtt{t}$. But then, we also have $\mathtt{s} \approx_\tau \mathtt{t}$ so that $\bigsqcup \Delta \sqsubseteq_G \tau$. ■

Given a strategy σ, we write $\sigma/\approx_\sigma$ for the quotient of T_σ by $\approx_\sigma$. We've already seen, in lemma 5.6.3, that $\sqsubseteq_G$ is well-behaved with respect to the blocks of $\sigma/\approx_\sigma$. This allows us to establish the following characterization of the compact strategies.

Lemma 5.6.6 If σ is a self-equivalent strategy for the game G then σ is compact if, and only if, $\sigma/\approx_\sigma$ is finite.

Proof To see that any strategy with finite set of blocks is compact, suppose that $\sigma \sqsubseteq_G \bigsqcup \Delta$ for some $\sqsubseteq$-directed set Δ. Each block B of σ is a block of $\bigsqcup \Delta$ and hence, by lemma 5.6.3, comes from some $\tau_B \in \Delta$. By directedness, these τ_Bs have an upper bound τ in Δ and since, if $\mathtt{s} \approx_\sigma \mathtt{t}$ in block B then $\mathtt{s} \approx_{\tau_B} \mathtt{t}$ and so $\mathtt{s} \approx_\tau \mathtt{t}$, we have $\sigma \sqsubseteq_G \tau$.

For the converse, consider the set $\Delta_\sigma = \{\sigma' \mid \sigma' \sqsubseteq_G \sigma \wedge \sigma'/\approx_{\sigma'} \text{ is finite}\}$. This set is directed since, by lemma 5.6.3, if $\sigma_1, \sigma_2 \in \Delta_\sigma$ then $\sigma_1 \vee \sigma_2 = (T_{\sigma_1} \cup T_{\sigma_2}, \approx_{\sigma_1} \cup \approx_{\sigma_2})$ is a self-equivalent strategy, $\sigma_1 \vee \sigma_2 \sqsubseteq_G \sigma$ and clearly it only has a finite number of blocks so $\sigma_1 \vee \sigma_2 \in \Delta_\sigma$. Obviously σ is an upper bound for Δ_σ. We claim that it's the least upper bound. Let $\mathtt{s} \approx_\sigma \mathtt{t}$. So $\mathtt{s}$ and $\mathtt{t}$ both live in some block, call it B, of σ. Let $\sigma_B = \{\mathtt{s'} \in P_G \mid \exists \mathtt{s} \in B.\, \mathtt{s'} \sqsubseteq^{\mathsf{even}} \mathtt{s}\}$. This is a self-equivalent strategy and clearly lives in Δ_σ. But then we must have $\mathtt{s} \approx_{\bigsqcup \Delta_\sigma} \mathtt{t}$ so that $\sigma \sqsubseteq_G \bigsqcup \Delta_\sigma$. So any strategy can be written as the lub of a directed collection of strategies with a finite number of blocks and it follows that no strategy with an infinite number of blocks can be compact. ■

So, any strategy can be seen as the lub of a directed set of strategies with a finite number of blocks:

Corollary 5.6.7 $\sqsubseteq_G$ is an algebraic CPO.

Monotonicity & continuity It's not hard to check that composition is monotone with respect to $\sqsubseteq$, *i.e.* if $\sigma \sqsubseteq_{G \multimap H} \sigma'$ and $\tau \sqsubseteq_{H \multimap J} \tau'$ then $\sigma ; \tau \sqsubseteq_{G \multimap J} \sigma' ; \tau'$. Consider now a directed set Δ of strategies for $H \multimap J$. Let $s \approx_{\sigma ; \bigsqcup \Delta} t$ with witnesses $u, v \in T_\sigma \parallel T_{\bigsqcup \Delta}$. We have $u \upharpoonright H, J \approx_{\bigsqcup \Delta} v \upharpoonright H, J$ and so, for some $\tau \in \Delta$, $u \upharpoonright H, J \approx_\tau v \upharpoonright H, J$. But then $s \approx_{\sigma ; \tau} t$ and so $s \approx_{\bigsqcup(\sigma ; \Delta)} t$, establishing that $\sigma ; \bigsqcup \Delta \sqsubseteq_{G \multimap J} \bigsqcup(\sigma ; \Delta)$. The opposite "inclusion" holds by monotonicity, so composition is continuous with respect to $\sqsubseteq$.

5.6.3 Resumé

The main purpose of this chapter has been to take a closer look at the problems associated with nondeterministic innocence encountered in chapter 3 from the alternate perspective afforded by AJM-games. We've seen that similar-looking problems arise here but also that they manifest themselves differently: whereas before we had the simple failure of innocence being preserved by composition, we find here a failure of monotonicity. Two different, independently discovered, but equivalent formulations of *saturation* were then shown to fix this problem.

But we also saw that this was only a partial solution, since most of the constraints on strategies that we're interested in seem not to cooperate with saturation (in any of its guises). We then saw how the notion of *self-equivalence* from [1], while cooperating well with all the constraints, encounters another form of the monotonicity problem: failure of self-equivalence being preserved by composition.

Trees, multisets & self-equivalence In the last couple of sections, we've made some progress by isolating a new notion of strategy and self-equivalence that doesn't rely in any way on the assumption of determinism. Moreover, this is a "conservative" adaptation of the old notions of strategy and self-equivalence in that it perfectly coincides with the latter as far as deterministic strategies are concerned (*i.e.* in those places where the latter notions actually work). Tellingly, this new formulation has a strong flavour of considering strategies as trees—as was suggested by the "indirect definition of innocence" of §3.7.2—but does so without fully moving away from the usual stance that views strategies as trace sets.

However, there's another way of regarding these new strategies. Consider, for example, $(\{\varepsilon, q \cdot 0q, q \cdot 1q\}, \mathsf{id})$ on $!\bot \multimap \bot$. This strategy may usefully be thought of as a *multiset* where the bang indices serve to tag two occurrences of the "same" element of some underlying set. This perspective motivates a notion of equivalence that would regard the above strategy as equivalent, not only to $(\{\varepsilon, q \cdot 2q, q \cdot 3q\}, \mathsf{id})$ say, but also to $(\{\varepsilon, q \cdot 5q\}, \mathsf{id})$ and so on. In the following (final) section, we describe such a notion of equivalence and proceed to sketch out an approach to bridging the gap between the rather crude category constructed above and the kind of category required to model, say, erratic PCF.

5.6.4 Towards a fully abstract model of Erratic PCF

The remarks in this final section are still somewhat tentative: the precise technical details are still a little amorphous but we hope to be able to report on this work in detail in the near future. Our main aim here is to give some hint as to how we can turn the above idea into something workable.

A refined ordering Let σ and τ be self-equivalent strategies for some game G. What does it mean for τ to improve σ? The basic idea is that, for each point $sa \in \mathsf{dom}(\sigma)$, there should be an equivalent point $ta' \in \mathsf{dom}(\tau)$ with (at least) the same options as σ—up to equivalence. More precisely, $\sigma \precsim_G \tau$ iff

$$\begin{aligned} sa \in \mathsf{dom}(\sigma) \quad \Rightarrow \quad & \exists ta' \in \mathsf{dom}(\tau).\, sa \approx_G ta' \;\wedge \\ & (sab \in T_\sigma \Rightarrow \exists b'.\, (ta'b' \in T_\tau \wedge sab \approx_G ta'b')). \end{aligned}$$

It's fairly straightforward to see that this defines a preorder on the set of all (self-equivalent) strategies for G. The overloading of the symbol $\precsim$ is justified by the following lemma which says that, for deterministic strategies, the above ordering coincides with the original ordering defined in §5.2.3. The proof is a cinch.

Lemma 5.6.8 If σ and τ are both deterministic and self-equivalent, $\sigma \precsim \tau$ if, and only if, $s \in T_\sigma \Rightarrow \exists t \in T_\tau.\, s \approx t$.

With a little reworking, the argument of proposition 5.5.4 can be used to show that composition is monotone with respect to $\precsim$. However, we do have a caveat to this. The above definition of the ordering is very clumsy to work with in practice; an alternative characterization—perhaps some kind of "intrinsic" characterization along the lines of §3.8.4—would be of real value. Such a characterization could also help in the proof, at a more general level, of results such as rationality of the quotient category—a property which, at the present time, can only be verified by tedious, *ad hoc* analysis.

History-freedom The most important constraint on the AJM model of PCF is undoubtedly history-freedom; this is what guarantees applicative behaviour. This constraint lifts to our refined setting without any problems. A strategy σ for a game G is history-free iff T_σ is history-free in the usual sense:

$$(sab \in T_\sigma \wedge t \in T_\sigma \wedge ta \in P_G) \Rightarrow tab \in T_\sigma.$$

Notice that we've completely side-stepped the problem we had with saturation, *i.e.* of finding some notion of "history-freedom up to equivalence"—a notion that wasn't particularly forthcoming. In this new framework, the traces and the equivalence have been deliberately separated, thus allowing conditions that only need to talk about the traces to do just that.

The main undischarged commitment in this work is to check all the technical details that ensure that this notion of ordering—and, in particular, the associated notion of equivalence—cooperate with the bang constructor. This is the only substantial remaining obstacle to building a (fully abstract) model of erratic PCF. As was mentioned above, this is work in progress and we hope to report on it before too long.

Chapter 6

Concluding remarks

6.1 Conclusions

In this thesis, we've explored the axis of determinacy/indeterminacy in the spirit of Abramsky's "semantic cube" methodology [9]. The main technical conclusions to be drawn follow immediately from the full abstraction results of chapters 3 and 4: the "observable" may-convergent behaviour of a process can be precisely described by its *traces*; whereas the more discriminating may/must behaviour of a (finitely branching) process can be characterized in terms of its traces and *divergences*. Furthermore, we see that the divergences of a determinate process are entirely determined by its traces so that the passage to indeterminate processes motivates us to render explicitly a hitherto implicit aspect of computation.

This is an instance of the more general point that, with the relaxation of semantic constraints possible with the machinery of game semantics, previously implicit properties of strategies/processes can become visible. In the case of nondeterminism, the property in question is that of divergence; but this is not without precedent. The work on game semantics for ground type references [5, 7, 8] tells a similar story: in the call-by-name case at least, a sharp distinction is drawn between those languages with active expressions and those where expressions are passive.

It seems then that the various "case studies" that have been carried out in game semantics in recent years are helping to clarify some well-known issues and controversies in computation. Of course, much more remains to be done; but, in this author's opinion at least, a basic foundation has now been laid [3, 5, 7, 8, 42, 56, 63] for a semantic assault on more "advanced" topics such as object-orientation and concurrency, as found in theoretical languages such as IPA [17] and in modern programming languages like ML and Java.

6.2 Work in progress & future directions

Applicative nondeterminism As we noted in chapter 3, a total reformulation of strategies as trees may be the best way to tackle this problem. The work in chapter 5 takes a step in this direction without completely abandoning the trace-based nature of strategies. It remains to be seen which approach is preferable—but we hope to report on this in the near future.

Probabilistic strategies Our work on divergences illustrates one way of getting at liveness information with game semantics. A contrasting approach is to use *probabilistic* strategies, as alluded to in chapter 4. In ongoing work with Vincent Danos, we've isolated a robust notion of probabilistic strategies and composition, leading to a category that's capable of modelling many classical probabilistic experiments and algorithms. Several lines of research follow naturally from this.

- The construction of a (fully abstract?) model of a probabilistic variant of Idealized Algol, *e.g.* IA extended with fair coin tossing.
- A probabilistic model of EIA. We already know that probabilistic strategies give a different, more optimistic, theory of must-convergence than strategies with divergences. Intuitively, there appears to be a connection to the question of *fairness*, but this remains to be properly formalized.
- Another fascinating direction is to consider *adaptive* algorithms. Many such algorithms, such as those used in neural nets, are probabilistic and it's natural to ask whether or not our semantic ideas can be applied.

Progress is currently being made on all three fronts—particularly the first which is largely just a reworking of the results of chapter 4, with a few technical twists—with some intriguing connections to arbitrary precision real arithmetic emerging.

Reliable countable nondeterminism As is remarked in [10], the assumption of *fair scheduling* in a simple imperative programming language with an operator for the interleaving of command sequences necessarily means that the language can express countably branching processes; the classic example is

$$\mathtt{x} := \mathtt{true};\ \mathtt{y} := 0;\ \mathtt{while}\ \mathtt{x}\ \mathtt{do}\ (\mathtt{y} := \mathtt{y} + 1 \parallel \mathtt{x} := \mathtt{false})$$

which, intuitively, ought to be observationally equivalent to '$\mathtt{x} := \mathtt{false} \parallel \mathtt{y} := \mho$' where $\mho$ may converge to any numeral with no possibility of divergence.

Since this assumption of fairness is of primary importance in reasoning about the properties of concurrent systems—and, bluntly, is a fundamental part of our intuition of how such systems should operate—we are motivated to an attempt at extending the results of chapter 4 to such countably branching processes.

This is well-known to be problematic for at least the following two reasons.

- *failure of continuity.* If S_n behaves like the "successor" function only for inputs less than n then $S_n(\mho)$ may diverge for any $n \in \mathbf{N}_0$ and yet $\mathsf{Succ}(\mho)$ must converge.
- *failure of associativity of hiding.* Recall the example from chapter 4. The essence of the problem was that no distinction could be made between a process that could keep running "forever" and one that had a "secret upper bound" on its behaviour.

The first of these problems is, in some sense, inevitable [10]. However, this need not be an obstacle to building an adequate (and hopefully fully abstract) model of, say, IA extended with $\mho$ since the work of Panangaden & Russell [75] shows that this failure of continuity can be isolated to a single "abstraction" mapping from a (continuous) intensional model to a fully abstract model—at least in the case of a countably nondeterministic `while` language.

As for the second problem, Roscoe's work on infinite traces in CSP [87] inspires a further refinement of our notion of strategy: in addition to the trace set and divergences, we have a third component of *infinite traces*. The idea is that, for the two processes described above, the strategy (on $\mathbf{C} \Rightarrow \mathbf{C}$) corresponding to the potentially infinite process has the infinite trace $\mathtt{qqaqa}\cdots$ while the finite but unbounded process is represented by a strategy with the same traces and divergences, but lacking the infinite trace.

This allows the failure of associativity of composition to be solved: a category of such strategies can be constructed but this work remains at too preliminary a stage to be reported on here.

Ambiguous choice? In this thesis, we've only studied erratic nondeterminism. The essential characterizing property of this is that once a choice has been made, we're stuck with it—even if it turns out to have been a "bad choice", *e.g.* leading to "unnecessary" divergence.

By means of contrast, *ambiguous* nondeterminism—as described by McCarthy's `amb` operator [62]—adopts a stronger position in this regard: the observable behaviour of 'M `amb` N' is no different to that of 'M `or` N' so, in particular, IA+`or` and IA+`amb` have the same theory of may contextual equivalence; the difference with `amb` is that it can only diverge if *both* its operands can. This inevitably leads to countably branching processes, *e.g.* '$\mathtt{9}\ \mathtt{amb}\ \mathbf{Y}_{\mathtt{Nat}}(\lambda x.\ \mathtt{0}\ \mathtt{amb}\ x+1)$', so a satisfactory account of countable nondeterminism is an essential prerequisite to a model of M&M contextual equivalence in IA+`amb`. It's unclear at this stage just how much more is needed for such a model; this must remain a topic for future research.

Bibliography

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. To appear in Information and Computation, 199?

[2] S. Abramsky. Semantics of interaction. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*. Cambridge University Press, 1997. Based on lectures given at the CLICS-II Summer School on Semantics and Logics of Computation, Isaac Newton Institute for Mathematical Sciences, Cambridge UK, September 1995.

[3] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Proceedings, Thirteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1998.

[4] S. Abramsky and R. Jagadeesan. New foundations for the geometry of interaction. *Information and Computation*, 111(1):53–119, 1994.

[5] S. Abramsky and G. McCusker. Full abstraction for Idealized Algol with passive expressions. To appear in Theoretical Computer Science, 199?

[6] S. Abramsky and G. McCusker. Games and full abstraction for the lazy λ-calculus. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 234–243. IEEE Computer Society Press, 1995.

[7] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions. In P. W. O'Hearn and R. D. Tennent, editors, *Algol-like Languages*, pages 297–329 of volume 2. Birkhaüser, 1997.

[8] S. Abramsky and G. McCusker. Call-by-value games. In M. Nielsen and W. Thomas, editors, *Computer Science Logic: 11th International Workshop Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[9] S. Abramsky and G. McCusker. Game semantics. Lecture notes to accompany Samson Abramsky's lectures at the 1997 Marktoberdorf summer school. To appear, 1998.

[10] K. Apt and G. Plotkin. Countable nondeterminism and random assignment. *Journal of the ACM*, 33(4):724–767, October 1986.

[11] P. Baillot, V. Danos, T. Ehrhard, and L. Régnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science* [52].

[12] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, revised edition edition, 1984.

[13] G. Berry. *Modeles completement adequats et stable de lambda-calculs*. PhD thesis, Universite Paris VII, 1979.

[14] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[15] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1985.

[16] S. D. Brookes. Full abstraction for a shared variable parallel language. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science* [50], pages 98–109.

[17] S. D. Brookes. The essence of Parallel Algol. In *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science* [51], pages 164–173.

[18] A. Bucciarelli and T. Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–291, 1993.

[19] A. Bucciarelli and T. Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110:265–296, 1994.

[20] A. Bucciarelli. *Sequential models of PCF: some contributions to the domain-theoretic approach to full abstraction*. PhD thesis, dipartimento di informatica, universita di pisa, 1993.

[21] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proc. POPL*, pages 328–342. ACM Press, 1992.

[22] E. Crank and M. Felleisen. Parameter-passing and the lambda calculus. In *Proc. POPL*. ACM Press, 1991.

[23] P.-L. Curien. Observable sequential algorithms on concrete data structures. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 432–443. IEEE Computer Science Press, 1992.

[24] P.-L. Curien. Sequentiality and full abstraction. In P. T. J. M. Fourman and A. M. Pitts, editors, *Applications of Categories in Computer Science*, pages 66–94. Cambridge University Press, 1992.

[25] V. Danos. *Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du λ-calcul)*. PhD thesis, Université Paris VII, June 1990.

[26] V. Danos, H. Herbelin, and L. Régnier. Game semantics and abstract machines. In *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science* [51], pages 394–405.

[27] V. Danos and L. Régnier. Machina ex deo. Unpublished manuscript, 1990.

[28] V. Danos and L. Régnier. Local and asynchronous beta-reduction (an analysis of Girard's execution formula). In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science* [50], pages 296–306.

[29] V. Danos and L. Régnier. Reversible, irreversible and optimal lambda-machines. In *Proceedings of 1996 Workshop on Linear Logic*, volume 3 of *Electronic notes in Theoretical Computer Science*. Elsevier, 1996.

[30] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[31] E. Dijkstra. *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[32] T. Ehrhard. Projecting sequential algorithms on the strongly stable functions. To appear in Annals of Pure and Applied Logic vol. 77 No. 3.

[33] M. P. Fiore, A. Jung, E. Moggi, P. O'Hearn, J. Riecke, G. Rosolini, and I. Stark. Domains and denotational semantics: History, accomplishments and open problems. Technical Report CSR-96-2, University of Birmingham, School of Computer Science, January 1996.

[34] P. J. Freyd, P. W. O'Hearn, A. J. Power, M. Takeyama, and R. D. Tennent. Bireflectivity. In *Mathematical Foundations of Programming Semantics, Eleventh Annual Conference, Tulane University, New Orleans, LA, March 29 - April 1, 1995*. Elsevier, 1995. Electronic Notes in Theoretical Computer Science 1.

[35] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[36] J.-Y. Girard. Geometry of interaction 2: Deadlock-free algorithms. In P. Martin-Löf and G. Mints, editors, *International Conference on Computer Logic, COLOG 88*, pages 76–93. Springer-Verlag, 1988. Lecture Notes in Computer Science 417.

[37] J.-Y. Girard. Geometry of interaction 1: Interpretation of System F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic Colloquium 88*, pages 221–260. North Holland, 1989.

[38] J.-Y. Girard. Towards a geometry of interaction. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 69–108. American Mathematical Society, 1989.

[39] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[40] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proceedings of ACM Symposium Principles of Programming Languages*, pages 15–26, January 1992.

[41] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Foundations of Computing. MIT Press, 1992.

[42] R. S. Harmer and G. A. McCusker. A fully abstract game semantics for finite non-determinism. In *Proceedings, Fourteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1999.

[43] M. C. B. Hennessy. The semantics of call-by-value and call-by-name in a nondeterministic environment. *SIAM J. Comput.*, 9(1):67–84, 1980.

[44] M. C. B. Hennessy and E. A. Ashcroft. Parameter-passing mechanisms and non-determinism. In *Proceedings, Ninth ACM Symposium on the Theory of Computing*, 1977.

[45] M. C. B. Hennessy and E. A. Ashcroft. A mathematical semantics for a non-deterministic typed λ-calculus. *Theoretical Computer Science*, 11:227–245, 1980.

[46] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[47] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. To appear in Information and Computation, 199?

[48] J. M. E. Hyland. A syntactic characterization of the equality in some models of the lambda calculus. *J. London Math. Soc. (2)*, 12:361–370, 1976.

[49] J. M. E. Hyland. Game semantics. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*. Cambridge University Press, 1997. Based on lectures given at the CLICS-II Summer School on Semantics and Logics of Computation, Isaac Newton Institute for Mathematical Sciences, Cambridge UK, September 1995.

[50] IEEE Computer Society Press. *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, 1993.

[51] IEEE Computer Society Press. *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science*, 1996.

[52] IEEE Computer Society Press. *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science*, 1997.

[53] B. Jacobs. Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69:73–106, 1994.

[54] G. Kahn and G. Plotkin. Domaines concrets. Technical Report 336, IRIA-Laboria, 1978.

[55] J. Laird. Full abstraction for functional languages with control. In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science* [52].

[56] J. Laird. *A Semantic Analysis of Control.* PhD thesis, University of Edinburgh, 1998. In preparation.

[57] P. J. Landin. A correspondence between ALGOL 60 and Church's lambda notation. *Communications of the ACM*, 8:89–101,158–165, 1965.

[58] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism.* PhD thesis, Department of Computer Science, University of Aarhus, February 1998.

[59] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, Berlin, 1971.

[60] P. Malacaria. Dalle macchine a ambienti alla geometria dell'interazione. Unpublished manuscript, 1993.

[61] P. Malacaria and L. Regnier. Some results on the interpretation of λ-calculus in operator algebras. In *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 63–72. IEEE Computer Society Press, 1991.

[62] J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 33–69. North Holland, 1963.

[63] G. McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types.* PhD thesis, Department of Computing, Imperial College, University of London, 1996.

[64] C. McLarty. *Elementary Categories, Elementary Toposes*, volume 21 of *Oxford Logic Guides.* Oxford University Press, 1991.

[65] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[66] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[67] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[68] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, July 1991.

[69] A. K. Moran. Natural Semantics for Non-Determinism. Licentiate Thesis, Chalmers University of Technology and University of Göteborg, Sweden, May 1994.

[70] A. K. Moran. *Call-by-name, Call-by-need, and McCarthy's Amb*. PhD thesis, Department of Computing Science, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden, September 1998.

[71] J. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.

[72] H. Nickau. Hereditarily sequential functionals. In *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg*, Lecture notes in Computer Science. Springer, 1994.

[73] P. W. O'Hearn. Note on Algol and conservatively extending functional programming. *Journal of Functional Programming*, 6(1):171–180, January 1996.

[74] C.-H. L. Ong. Correspondence between operational and denotational semantics. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol 4*, pages 269–356. Oxford University Press, 1995.

[75] P. Panangaden and J. R. Russell. A category-theoretic semantics for unbounded indeterminacy. In *Proceedings, Fifth Conference on Mathematical Foundations of Programming Semantics*. LNCS 442, 1989.

[76] D. M. Park. On the semantics of fair parallelism. In *Abstract Software Specifications*. Springer-Verlag, 1980. Lecture Notes in Computer Science Vol. 86.

[77] D. M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, Berlin, 1981. Springer-Verlag. Lecture Notes in Computer Science Vol. 104.

[78] D. M. Park. The "fairness" problem and nondeterministic computing networks. In *Foundations of Computer Science IV*. Mathematisch Centrum, 1983.

[79] C. S. Pitcher. *Functional Programming and Erratic Non-Determinism*. PhD thesis, Programming Research Group, University of Oxford, Oxford, United Kingdom, 1999. To appear.

[80] G. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5:452–487, 1976.

[81] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[82] L. Régnier. *Lambda-Calcul et Réseaux*. PhD thesis, Université Paris VII, January 1992.

[83] L. Régnier. Une équivalence sur les lambda-termes. *Theoretical Computer Science*, 126, 1994.

[84] J. C. Reynolds. The essence of Algol. In *Proceedings of the 1981 International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, 1981.

[85] J. Riecke. Fully abstract translations between functional languages. *Mathematical Structures in Computer Science*, 11, 1993.

[86] A. W. Roscoe. An alternative order for the failures model. In 'Two papers on CSP', Technical monograph PRG-67, Oxford University Computing Laboratory, 1989.

[87] A. W. Roscoe. Unbounded non-determinism in CSP. *Journal of Logic and Computation*, 3(2):131–172, April 1993.

[88] D. S. Scott. A type-theoretic alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440, 1993.

[89] M. B. Smyth. Powerdomains. *Journal of Computer and Systems Sciences*, 16(1):23–36, February 1978.

[90] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, volume 1 of *The MIT Press Series in Computer Science*. The MIT Press, 1977.

[91] A. M. Turing. Computing machinery and intelligence. In D. R. Hofstadter and D. C. Dennett, editors, *The Mind's I*. Basic Books (New York) and Harvester (Brighton), 1981.

[92] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the lambda calaulus. *SIAM Journal on Computing*, 5(3):488–521, September 1976.

[93] G. Winskel. *The Formal Semantics of Programming Languages*. Foundations of Computing. The MIT Press, Cambridge, Massachusetts, 1993.