Information and Computation **160**, 1–61 (2000)
doi:10.1006/inco.1999.2845, available online at http://www.idealibrary.com on IDEAL®

Games and Full Abstraction for **FPC**

Guy McCusker

School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton BN1 9QH, United Kingdom
E-mail: guym@cogs.susx.ac.uk

A new category of games is developed which improves over existing such categories in that it interprets *sums* as well as products, function spaces, and recursive types. A model of **FPC**, a sequential functional language with just this type structure, is described and shown to be fully abstract.

1. INTRODUCTION

Since the 1970s, denotational semantics has been used to model, reason about, and describe programming languages. Typically, for a given language $\mathscr{L}$, there is an understood class of observable properties that we are interested in. This gives rise to an *observational preorder* $\underset{\sim}{\sqsubset}$ on the language, where for programs M and N the assertion $M \underset{\sim}{\sqsubset} N$ means that every observable property of M is also satisfied by N. Denotational semantics models a program M as a mathematical object $[\![M]\!]$ (usually some kind of function), and there is an order in the model, written as $\preccurlyeq$. There are two natural properties to ask for:

- $[\![M]\!] \preccurlyeq [\![N]\!] \Rightarrow M \underset{\sim}{\sqsubset} N$.
- $M \underset{\sim}{\sqsubset} N \Rightarrow [\![M]\!] \preccurlyeq [\![N]\!]$.

The first of these, *soundness*, says that we can use the model to prove properties of the language, while the second, *completeness*, says that every observable property of the language is captured by the semantics. A model which is both sound and complete is called *fully abstract* and captures exactly the behaviour of programs of $\mathscr{L}$. If such a model can be constructed abstractly, without recourse to the syntax of $\mathscr{L}$, it sheds new light on the computational concepts embodied in the language and yields deep structural information and insight.

While the construction of sound models for many kinds of language is well understood, the problem of finding fully abstract models for sequential languages,

0890-5401/00 $35.00

particularly those with higher types, has proved to be a very subtle and difficult one. The famous "Full Abstraction Problem for **PCF**," raised by Plotkin in 1977 and studied intensively ever since [Plo77, Mil77, Cur93, Ong95], remained open until very recently, when a number of solutions emerged [OR95, AJM94, AJM97, HO97, Nic94]. One successful approach was via *game semantics*, a novel kind of model which includes more intensional information than traditional domain-theoretic ones. The correspondence between the *strategies* used to model programs and the programs themselves was shown to be so strong that these results could be said to provide a definitive analysis of **PCF**. But **PCF** is a very simple language, so it is natural to want to extend this work.

An important feature of many languages is the availability of *recursive types*. First steps towards modelling such languages with games were taken in [AM95b], and in [AM95a] a fully abstract model of the lazy λ-calculus was presented, making use of this work. Equally important, however, are *sum* types. With sums and recursive types, one can construct the types of lists, trees, and much more. It was therefore a major shortcoming of the categories of games previously proposed that there was no clear way of modelling such types. Sums bring with them new possibilities, particularly a notion of partial computation or laziness: a program of type $\mathbb{B}$ (Booleans) can only ever return true or false and terminate, whereas a program of type $\mathbb{B} + \mathbb{B}$ first tells whether its output is in the left or right component and then waits to be asked for more information. In order to handle this, a whole new category of games needed to be developed.

In this paper, we present a solution to this problem. A new category, similar to that of [HO97] but different in a number of important ways, is presented and used to give a fully abstract model to the language **FPC**. **FPC** is possibly the canonical calculus for discussing recursive types, sums, products, and function spaces together. It originally arose as a metalanguage for denotational semantics [Plo85], so it could be said that any category proposed for denotational semantics should at least be able to model **FPC**. A further indication of the importance of this language is that it appears in two well-known modern textbooks [Gun92, Win93]. It has recently been studied by Fiore and Plotkin [FP94, Fio96], who provide an axiomatisation of sound domain-theoretic models of the call-by-value variant, and by Gordon [Gor95b], who develops an operationally based theory of program equivalence for it. Here we provide the first fully abstract denotational semantics of a purely sequential, functional language as rich as **FPC**.

1.1. Game Semantics: An Informal Introduction

We now give an informal presentation of some of the basic ideas of game semantics to provide the reader with some intuition as to how functional programs are interpreted. The series of examples which follows is adapted from the introductory section of a set of lecture notes written in conjunction with Abramsky [AM98c], who has kindly granted permission for the material to be used here.

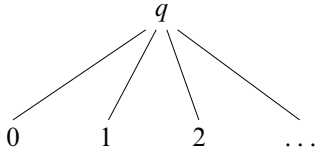
As the name suggests, game semantics models computation as the playing of a certain kind of game, with two participants, called Player (P) and Opponent (O).

P is to be thought of as representing the system under consideration, while O represents the environment. In the case of programming languages, the system corresponds to a term (a piece of program text) and the environment to the context in which the term is used.

In the games we shall consider, O always moves first—the environment sets the system going—and thereafter the two players make moves alternately. What these moves are, and when they may be played, are determined by the rules of each particular game. Since in a programming language a *type* determines the kind of computation which may take place, types will be modelled as games; a *program* of type A determines how the system behaves, so programs will be represented as *strategies* for P, that is, predetermined responses to the moves O may make.

1.1.1. Modelling Values

In standard denotational semantics, values are *atomic*: a natural number is represented simply as $n \in \mathbb{N}$. In game semantics, each number is modelled as a simple interaction: the environment starts the computation with an initial move q (a *question*: "What is the number?"), and P may respond by playing a natural number (an *answer* to the question). So the game **N** of natural numbers looks like this:



and the strategy for 3 is "When O plays q, I will play 3."

N

q O
3 P

In diagrams such as the above, time flows downwards: here O has begun by playing q, and at the next step P has responded with 3, as the strategy dictates.

1.1.2. Functions

The interactions required to model functions are a little more complex. The view taken in game semantics is that the environment of a function consumes the output and provides the input, while the function itself consumes the input and produces the output. The game $\mathbf{N} \multimap \mathbf{N}$ is therefore formed from "two copies of **N**," one for input, one for output. In the output copy, O may demand output by playing the move q and P may provide it. In the input copy, the situation is reversed: P may

demand input with the move q. Thus the O/P role of moves in the input copy is reversed. Plays of this game take the following form.

$\mathbf{N} \multimap \mathbf{N}$	
	q O
q	P
3	O
	4 P

The play above is a particular run of the strategy modelling the successor function:

> "When O asks for output, I will ask for input; when O provides input n, I will give output $n+1$."

It is important to notice that the play in each copy of $\mathbf{N}$ (that is, each column of the above diagram) is indeed a valid play of $\mathbf{N}$: it is not possible for O to begin with the third move shown above, supplying an input to the function immediately. Note also that nonstrict functions can be modelled. Here is the strategy which returns 3 without ever investigating what its argument is.

$\mathbf{N} \multimap \mathbf{N}$	
	q O
	3 P

These relatively simple ideas let us model all first-order functions. For example, a play in the strategy for addition might look like this.

$\mathbf{N} \multimap \mathbf{N} \multimap \mathbf{N}$		
		q O
q		P
3		O
	q	P
	2	O
		5 P

The same idea lets us form $A \multimap B$ for any games A and B: take a copy of A and a copy of B, "place them side by side," and reverse the O/P roles of the moves in A.

1.2. Higher-Order Functions

The strategy for the function $\lambda f.\ f\,0\,1$ plays as follows.

$(\mathbf{N} \multimap \mathbf{N} \multimap \mathbf{N}) \multimap \mathbf{N}$			
		q	O
	q		P
q			O
0			P
	q		O
	1		P
		n	O
		n	P

Here O plays the role of the function f in the game $(\mathbf{N} \multimap \mathbf{N} \multimap \mathbf{N})$ as well as demanding output form the rightmost **N**. P first asks for the output from f; when O asks for the first input to f, P supplies 0; when O asks for the second input, P supplies 1; and when O supplies n as output for f, P copies this as the overall output.

The choice of moves made by O in the example above is by no means the only one. For example, O could ask for the arguments to f in the other order or could neglect to ask for the arguments at all. But P's strategy would be the same regardless: answer 0 to the first input, 1 to the second, and copy the output of f as the overall output.

1.3. Roadmap

The remainder of this paper is organised as follows. In Section 2 we give detailed definitions of the games and strategies which will form our model of **FPC**, together with various operations on them, build the category which contains the fully abstract model, and show that it has all the necessary structure to interpret **FPC**. Section 3 introduces the syntax and operational semantics of **FPC**, and Section 4 defines the interpretation of **FPC** in our category of games. In Section 5, a *definability* result is proved which shows that every finite element of the model is indeed the denotation of a term of **FPC**. This yields the full abstraction theorem which is the main result of this paper. Finally, Section 6 describes related work.

2. A CATEGORY OF GAMES

In this section we develop a category of games with all the structure needed to interpret **FPC**: it is Cartesian closed, has a notion of sum, and has canonical solutions of "domain equations." This category, called the *extensional category* $\mathscr{E}$, is

built in three stages. First a symmetric monoidal category $\mathscr{G}$ is constructed. Then, using ideas from linear logic, this is used as the basis of an *intensional* Cartesian closed category $\mathscr{I}$. Finally, $\mathscr{E}$ is given by a certain quotient of $\mathscr{I}$.

The fundamental ideas of arenas, views, innocent strategies, and so on are due to Hyland and Ong [HO97]. The definitions in this paper should be seen as a streamlined presentation of theirs, making use of some of the apparatus of [AJM97]; there are a couple of important differences which are necessary to allow the interpretation of sums, which will be pointed out when they are encountered.

The organisation of this section is as follows. We first give the definitions of arenas, views, and legal positions which are fundamental to all our work. Section 2.2 introduces games and gives game interpretations of the connectives $\otimes$ and $\multimap$. We then give the definition of a strategy and show how strategies can be composed, before introducing the crucial class of *innocent* strategies. In Section 2.3 these ideas are used to build a category of games which is shown to be symmetric monoidal closed and to have products. Sections 2.4 and 2.5 build a second category, the *intensional category* $\mathscr{I}$ which is Cartesian closed and has good order-theoretic properties. This category is in turn used to construct the *extensional category* $\mathscr{E}$ in Section 2.6. Finally, the structure required to interpret sum types and recursive types is discussed in Sections 2.7 and 2.8 and shown to exist in $\mathscr{E}$.

2.1. Arenas, Views, and Legal Positions

Our notion of game is built over a primitive structure called an *arena*. One can think of an arena as mapping out a playing area for a game, while the game itself specifies what can be done within its arena.

DEFINITION. An *arena* A is specified by a structure $\langle M_A, \lambda_A, \vdash_A \rangle$ where

- M_A is a set of *moves*;
- $\lambda_A : M_A \to \{\mathsf{O}, \mathsf{P}\} \times \{\mathsf{Q}, \mathsf{A}\}$ is a *labelling* function which indicates whether a move is by Opponent (O) or Player (P) and whether it is a question (Q) or an answer (A). We write the set $\{\mathsf{O}, \mathsf{P}\} \times \{\mathsf{Q}, \mathsf{A}\}$ as $\{\mathsf{OQ}, \mathsf{OA}, \mathsf{PQ}, \mathsf{PA}\}$, and use λ_A^{OP} to mean λ_A followed by left projection, so that $\lambda_A^{\mathsf{OP}}(m) = \mathsf{O}$ if $\lambda_A(m) = \mathsf{OQ}$ or $\lambda_A(m) = \mathsf{OA}$. Define λ_A^{QA} in a similar way. Finally, $\bar{\lambda}_A$ is λ_A with the O/P part reversed, so that

$$\bar{\lambda}_A(m) = \mathsf{OQ} \Leftrightarrow \lambda_A(m) = \mathsf{PQ}$$

and so on. If $\lambda^{\mathsf{OP}}(m) = \mathsf{O}$, we call m and O-move; otherwise, m is a P-move;

- $\vdash_A$ is a relation between $M_A + \{\star\}$ (where $\star$ is just a dummy symbol) and M_A, called *enabling*, which satisfies

(e1) $\star \vdash_A m \Rightarrow \lambda_A(m) = \mathsf{OQ} \wedge [n \vdash_A m \Leftrightarrow n = \star]$

(e2) $m \vdash_A n \wedge \lambda^{\mathsf{QA}}(n) = A \Rightarrow \lambda_A^{\mathsf{QA}}(m) = \mathsf{Q}$;

(e3) $m \vdash_A n \wedge m \neq \star \Rightarrow \lambda_A^{\mathsf{OP}}(m) \neq \lambda_A^{\mathsf{OP}}(n)$.

The idea of the enabling relation is that when a game is played, a move can only be made if a move has already been made to enable it. The $\star$ enabler is special—it says which moves are enabled at the outset. A move m such that $\star \vdash_A m$ is called *initial*. Conditions (e2) and (e3) say that answers are enabled by questions and that the protagonists always enable each other's moves, never their own.

An arena can be thought of as a sort of "forest" of moves, where m is a parent of n in the tree if $m \vdash n$. (But note that the enabling relation does not necessarily give a forest in the technical sense—so these remarks should be taken with a pinch of salt). The roots of the forest are the initial moves and the O/P labelling alternates as we move down the branches.

Given an arena, we are interested in sequences of moves of a certain kind. Before defining these, let us fix our notation for operations on sequences. If s and t are sequences, we write st for their concatenation. We also write sa for the sequence s with element a appended. Sometimes we use the notation $s \cdot t$ or $s \cdot a$ when it aids legibility. The prefix ordering on sequences is denoted by $\sqsubseteq$, and ε is the empty sequence. The subsequence ordering is written as $\preccurlyeq$. The length of a sequence s is denoted by $|s|$. Finally, if s is a sequence and m is an occurrence of an element of s, then $s_{\leqslant m}$ denotes the prefix of s up to and including m, and $s_{<m}$ is the prefix of s up to, but not including, m.

DEFINITION. A *justified sequence* in an arena A is a sequence s of moves of A, together with an associated sequence of pointers: for each noninitial move m in s, there is a pointer to a move n earlier in s such that $n \vdash_A m$. We say that the move n *justifies* m. Note that the first move in any justified sequence must be initial, since it cannot possibly have a pointer to an earlier move attached to it; so by (e1), justified sequences always start with an opponent question. When drawing a justified sequence, we will use curved arrows to denote the justification pointers; in fact, we often omit the pointers altogether.

Given a justified sequence s, define the *player view* $\ulcorner s \urcorner$ and *opponent view* $\llcorner s \lrcorner$ of s by induction on $|s|$, as follows.

$$
\begin{aligned}
\ulcorner \varepsilon \urcorner &= \varepsilon. \\
\ulcorner s \cdot m \urcorner &= \ulcorner s \urcorner m, && \text{if } m \text{ is a P-move.} \\
\ulcorner s \cdot m \urcorner &= m, && \text{if } \star \vdash m. \\
\ulcorner s \cdot \overset{\frown}{m} \cdot t \cdot n \urcorner &= \ulcorner s \urcorner \cdot \overset{\frown}{m} \cdot n, && \text{if } n \text{ is an O-move.} \\
\llcorner \varepsilon \lrcorner &= \varepsilon. \\
\llcorner s \cdot m \lrcorner &= \llcorner s \lrcorner m, && \text{if } m \text{ is an O-move.} \\
\llcorner s \cdot \overset{\frown}{m} \cdot t \cdot n \lrcorner &= \llcorner s \lrcorner \cdot \overset{\frown}{m} \cdot n, && \text{if } n \text{ is a P-move.}
\end{aligned}
$$

Note that the view of a justified sequence need not itself be justified: the appearance of a move m in the view does not guarantee the appearance of its justifier. This will be rectified when we impose the *visibility condition*, to follow.

The key differences between our definitions and those of Hyland and Ong [HO97] are that we allow questions to be justified by answers and that the definition of view makes no reference to the question–answer classification of moves. The

former is of the greatest importance to this work: it is required for the interpretation of sums to follow. The latter is less fundamental for the results of the present paper, but in more recent work, Laird has fruitfully exploited the decoupling of the notion of view from the question–answer structure to model languages with nonlocal control operators [Lai97].

A justified sequence s is *well formed* if it satisfies

(w1) Players alternate: if $s = s_1 mns_2$ then $\lambda^{\mathsf{OP}}(m) \neq \lambda^{\mathsf{OP}}(n)$.

(w2) The *bracketing condition.* We say that a question q in s is *answered* by a later answer a in s if q justifies a. The bracketing condition is satisfied by s if for each prefix $tqua$ of s with q answered by a, the last unanswered question in tqu is q; in other words, when an answer is given, it is always to the most recent question which has not been answered—the *pending question.*

A useful intuition is to think of questions as left parentheses, (, and answers as right parentheses,). In order to satisfy the bracketing condition, the string of brackets must be a prefix of a well formed string of brackets, and furthermore each) must be justified by the corresponding (. Of course this is where the name "bracketing condition" comes from. In fact, we can take this further, writing O-questions as [, P-answers as], P-questions as (, and O-answers as).

A well formed sequence s is *legal*, or is a *legal position*, if it also satisfies the following *visibility condition*:

- if $tm \sqsubseteq s$ where m is a P-move, then the justifier of m occurs in $\ulcorner t \urcorner$.
- if $tm \sqsubseteq s$ where m is a noninitial O-move, then the justifier of m occurs in $\llcorner t \lrcorner$.

We write L_A for the set of legal positions of A.

Example. The simplest arena is the *empty arena* I, given by $\langle \varnothing, \varnothing, \varnothing \rangle$. A slightly more interesting arena is that for the natural numbers, $\mathbf{N} = \langle M_{\mathbf{N}}, \lambda_{\mathbf{N}}, \vdash_{\mathbf{N}} \rangle$, where

$$\begin{aligned} M_{\mathbf{N}} &= \{q\} \cup \{n \mid n \in \mathbb{N}\} \\ \lambda_{\mathbf{N}}(q) &= \mathsf{OQ} \\ \lambda_{\mathbf{N}}(n) &= \mathsf{PA} && \text{for all } n \\ \star &\vdash_{\mathbf{N}} q \\ q &\vdash_{\mathbf{N}} n && \text{for all } n. \end{aligned}$$

The only legal position of I is ε, because there are no moves. The legal positions of $\mathbf{N}$ have the form $qn_1qn_2q...$, with each answer n_i justified by the immediately preceding question. Note that the O-view of any such position is the whole position, whereas the P-view of $qn_1 \cdots qn_iq$ is just the final q.

Views are fundamental to the game semantics we present. As such, it is necessary to reason about their properties. A series of helpful lemmas now follows, starting with one that shows how views cooperate with the bracketing condition.

LEMMA 2.1. *Let $s \in L_A$. If* P *is to move, then the pending question of s is the same as that of $\ulcorner s \urcorner$. If* O *is to move, then if there is a pending question, it is the same as that of $\llcorner s \lrcorner$, and if not, then there is no pending question in $\llcorner s \lrcorner$.*

Proof. By induction on $|s|$. For the base case, note that at ε, O is to move and there is no pending question; but nor is there in $\llcorner \varepsilon \lrcorner = \varepsilon$. For the inductive step, consider the string sm. There are three cases.

- P to move. If m is initial then m is a question and is therefore the pending question of sm; but $\ulcorner sm \urcorner = m$, whose pending question is obviously m, as claimed. Otherwise, sm has the form $s_1 n s_2 m$ where n justifies m. Then $\ulcorner sm \urcorner = \ulcorner s_1 \urcorner nm$. If m is a question, it is obviously the pending question of both sm and $\ulcorner sm \urcorner$. If it is an answer then it answers n, and by the bracketing condition the pending question of sm is that os s_1. Applying the inductive hypothesis tells us that this is the pending question of $\ulcorner s_1 \urcorner$, and hence of $\ulcorner s_1 \urcorner nm = \ulcorner sm \urcorner$, as required.
- O to move, no pending question. Let $sm = s_1 n s_2 m$, where n justifies m. Since there is no pending question, m must be an answer, and by the bracketing condition there cannot be a pending question in s_1. So the inductive hypothesis tells us that there is no pending question in $\llcorner s_1 \lrcorner$ and hence none in $\llcorner s_1 \lrcorner nm = \llcorner sm \lrcorner$.
- O to move, and there is a pending question in sm. Again, if m is a question the result follows trivially. If not, let $s = s_1 n s_2 m$ where n is the question justifying answer m. By the bracketing condition, the pending question of sm is that of s_1, which by the inductive hypothesis is that of $\llcorner s_1 \lrcorner$, and hence of $\llcorner sm \lrcorner$, as required. ∎

Views also work in harmony with legal positions, as the next few results show. The proofs are straightforward verifications.

LEMMA 2.2.

1. *If $s \in L_A$ and $\ulcorner s \urcorner a \in L_A$ then $sa \in L_A$, for any* P*-move a.*
2. *If $s \in L_A$ and $\llcorner s \lrcorner a \in L_A$ then $sa \in L_A$, for any* O*-move a.*

LEMMA 2.3. *If $s \in L_A$ then both $\ulcorner s \urcorner \in L_A$ and $\llcorner s \lrcorner \in L_A$.*

Consider a legal position in an arena. There may be several initial moves, each of which starts a fresh "thread" of discussion. The following three lemmas establish a useful property of such treads, that each thread or group of threads itself forms a legal position.

DEFINITION. Let s be a legal position of an arena A and let m be a move in s. We say that m is *hereditarily justified* by an occurrence of a move n in s if the chain of justification pointers leading back from m ends at n, i.e., m is justified by some move m_1, which is in turn justified by m_2 and so on until some m_k is justified by an initial move n. We write $s \upharpoonright n$ for the subsequence of s containing all moves hereditarily justified by n. This notation is slightly ambiguous, because it confuses the move n with a particular occurrence of n; however, no difficulty will arise in practice. We similarly define $s \upharpoonright I$ for a set I of (occurrences of) initial moves in s to be the subsequence of s consisting of all moves hereditarily justified by a move of I.

DEFINITION. A segment $m \cdots n$ of a justified sequence s in which m justifies n will be called an O-*segment* if m is an O-move and a P-segment otherwise.

LEMMA 2.4. *Let s be a legal position and $m \cdots n$ be an* O- *or* P-*segment in s. Let I and J be sets of (occurrences of) initial moves of s which partition the set of all initial moves in s. If m (and hence also n) is hereditarily justified by a move of I, then*

$$m \cdots n \upharpoonright J = \theta_1 \cdots \theta_k$$

for some sequence of O-*segments $\theta_1, ..., \theta_k$.*

Proof. First note that in any legal position, the visibility condition implies that any P-move is hereditarily justified by the same initial move as the immediately preceding O-move. The visibility condition also implies that $m \cdots n$ has the form

$$m \cdots n = m \cdot \phi_1 \cdots \phi_l \cdot n$$

for some sequence of O- or P-segments $\phi_1, ..., \phi_l$. We prove the required result by induction on the length of the segment $m \cdots n$. The base case is trivial: $mn \upharpoonright J = \varepsilon$. For the inductive step, first consider the case when m is an O-move. Then each ϕ_i is a P-segment with its endpoints hereditarily justified by a move in I, so applying the inductive hypothesis, each $\phi_i \upharpoonright J$ has the required form. Since

$$m \cdots n \upharpoonright J = \phi_1 \upharpoonright J \cdots \phi_l \upharpoonright J$$

the result follows. If m is a P-move, then some of the ϕ_i may have their endpoints hereditarily justified by a move of J, so we cannot apply the inductive hypothesis as above. But in such a case, $\phi_i \upharpoonright J$ is itself an O-segment, so the result follows. ∎

LEMMA 2.5. *Let s be a legal position and I a set of initial moves in s.*

1. $\ulcorner s \urcorner \upharpoonright I \preccurlyeq \ulcorner s \upharpoonright I \urcorner$.
2. $\llcorner s \lrcorner \upharpoonright I \preccurlyeq \llcorner s \upharpoonright I \lrcorner$.

Proof. The proof of the first part is easy: all moves in $\ulcorner s \urcorner$ have the same initial justifier, so either $\ulcorner s \urcorner \upharpoonright I = \varepsilon$ or $\ulcorner s \urcorner \upharpoonright I = \ulcorner s \urcorner$. In the latter case, $s \upharpoonright I$ contains all the moves of $\ulcorner s \urcorner$, so $\ulcorner s \upharpoonright I \urcorner$ is equal to $\ulcorner s \urcorner$.

For the second part, the proof proceeds by induction on the length of s. The base case is trivial. For the inductive step, consider two separate cases.

- Sequence sm, m an O-move. We have

$$\begin{aligned} \llcorner sm \lrcorner \upharpoonright I &= \llcorner s \lrcorner \upharpoonright I \cdot m \upharpoonright I \\ &\preccurlyeq \{\text{inductive hypothesis}\} \\ &\quad \llcorner s \upharpoonright I \lrcorner \cdot m \upharpoonright I \\ &= \llcorner sm \upharpoonright I \lrcorner . \end{aligned}$$

• Sequence $smtn$, n a P-move justified by m. Now we make use of Lemma 2.4. Suppose m and n are hereditarily justified by a move in I. Then

$$smtn \upharpoonright I = s \upharpoonright I \cdot m \cdot t \upharpoonright I \cdot n$$

and therefore $\lfloor smtn \upharpoonright I \rfloor = \lfloor s \upharpoonright I \rfloor mn$ which is a supersequence of $\lfloor s \rfloor \upharpoonright I \cdot mn$ by the inductive hypothesis and hence contains $\lfloor smtn \rfloor \upharpoonright I$ as a subsequence. If m and n are not hereditarily justified in I, let J be the set containing all initial moves of s which are not in I. Then m and n are moves in $s \upharpoonright J$ with m justifying n, so by Lemma 2.4, the subsequence $m \cdots n \upharpoonright I$ has the form

$$m_1^- \cdots m_1 m_2^- \cdots m_2 \cdots m_i^- \cdots m_i,$$

where each m_i is a P-move justified by m_i^-. So

$$\lfloor smtn \upharpoonright I \rfloor = \lfloor s \upharpoonright I \rfloor \cdot m_1^- m_1 \cdot m_2^- m_2 \cdots m_i^- m_i,$$

which is a supersequence of $\lfloor s \rfloor \upharpoonright I$ by the inductive hypothesis; but $\lfloor smtn \rfloor \upharpoonright I = \lfloor s \rfloor \upharpoonright I$ in this case, so the proof is complete. ▮

LEMMA 2.6. *Let s be a legal position and I a set of initial moves in s. Then $s \upharpoonright I$ is legal.*

Proof. It is easy to see that $s \upharpoonright I$ is a justified sequence. To see that it is alternating, first note that if $t_1 mnt_2$ is legal and m is an O-move, then by the visibility condition m must have the same hereditary justifier as n. Let m_1 and m_2 be any two moves of $s \upharpoonright I$ by the same player, with m_1 occurring first; we will show that they are not consecutive. So suppose m_1 and m_2 are O-moves in $s \upharpoonright I$. Then m_1 is followed in s by some P-move n, and n must be hereditarily justified by the same move as m_1, so n appears in $s \upharpoonright I$, so m_1 and m_2 are not consecutive. If m_1 and m_2 are two P-moves in s, then m_2 is immediately preceded in s by some O-move n, and n must have the same hereditary justifier as m_2, so n appears in $s \upharpoonright I$, so that m_1 and m_2 are not consecutive. Therefore $s \upharpoonright I$ is alternating.

For the bracketing condition, note that if a question in s has been answered, then it appears in $s \upharpoonright I$ if and only if its answer does. Therefore if t is any segment such that all questions asked in t are answered in t, then $t \upharpoonright I$ also has this property. So if $t_1 \upharpoonright I \cdot q \cdot t_2 \upharpoonright I \cdot a$ is a prefix of $s \upharpoonright I$, corresponding to a prefix $t_1 q t_2 a$ of s with a answering q, then because q was pending at $t_1 q t_2$, it is also pending at $t_1 \upharpoonright I \cdot q \cdot t_2 \upharpoonright I$. For the visibility condition, suppose that tm is a prefix of s such that $t \upharpoonright I \cdot m$ is a prefix of $s \upharpoonright I$. The justifier of m appears in $\lceil t \rceil$ or $\lfloor t \rfloor$, according to whether m is a P-move or an O-move. But then it must appear in $\lceil t \rceil \upharpoonright I$ or $\lfloor t \rfloor \upharpoonright I$, and hence in $\lceil t \upharpoonright I \rceil$ or $\lfloor t \upharpoonright I \rfloor$, by Lemma 2.5, so the visibility condition is satisfied. ▮

2.2. Games and Strategies

DEFINITION. A *game* A is specified by a structure $\langle M_A, \lambda_A, \vdash_A, P_A \rangle$ where

- $\langle M_A, \lambda_A, \vdash_A \rangle$ is an arena.
- P_A is a nonempty, prefix-closed subset of L_A, called the *valid positions*, and satisfying

$$\text{if } s \in P_A \text{ and } I \text{ is a set of initial moves of } s \text{ then } s \upharpoonright I \in P_A.$$

We sometimes refer to the valid positions of a game A as *plays* of the game. The condition on P_A reinforces the idea that threads of play are separate. The main use made of this condition is to ensure that for any initial move m and valid position s, the string $s \upharpoonright m$ is also valid, which is important in the definition of the ! connective.

EXAMPLE. There is a unique game built over the empty arena, namely the *empty game*, which we also write as I. Its set of valid positions is just $\{\varepsilon\}$.

The game for the natural numbers, again written $\mathbf{N}$, has arena $\mathbf{N}$, and set of valid positions $\{\varepsilon, q\} \cup \{qn \mid n \in \mathbb{N}\}$, so that a play of the game "computes" a single natural number.

2.2.1. Multiplicatives

Given games A and B, define new games $A \otimes B$ and $A \multimap B$ as follows.

$$M_{A \otimes B} = M_A + M_B.$$

$$\lambda_{A \otimes B} = [\lambda_A, \lambda_B].$$

$$\star \vdash_{A \otimes B} n \Leftrightarrow \star \vdash_A n \vee \star \vdash_B n.$$

$$m \vdash_{A \otimes B} n \Leftrightarrow m \vdash_A n \vee m \vdash_B n.$$

$$P_{A \otimes B} = \{s \in L_{A \otimes B} \mid s \upharpoonright A \in P_A \wedge s \upharpoonright B \in P_B\}.$$

$$M_{A \multimap B} = M_A + M_B.$$

$$\lambda_{A \multimap B} = [\bar{\lambda}_A, \lambda_B].$$

$$\star \vdash_{A \multimap B} m \Leftrightarrow \star \vdash_B m$$

$$m \vdash_{A \multimap B} n \Leftrightarrow m \vdash_A n \vee m \vdash_B n \vee [\star \vdash_B m \wedge \star \vdash_A n] \quad \text{for} \quad m \neq \star.$$

$$P_{A \multimap B} = \{s \in L_{A \multimap B} \mid s \upharpoonright A \in P_A \wedge s \upharpoonright B \in P_B\}.$$

In the above, $s \upharpoonright A$ denotes the subsequence of s consisting of all moves from M_A; $s \upharpoonright B$ is analogous. The conflict with the previously introduced notation $s \upharpoonright I$ should

not cause any confusion. The condition on valid plays of $A \otimes B$ and $A \multimap B$, that the play in each component should itself be valid, will be referred to as the *projection condition*.

EXAMPLE. In the game $\mathbf{N} \otimes \mathbf{N}$, all valid positions are prefixes of positions of the form shown below.

$\mathbf{N} \otimes \mathbf{N}$	or	$\mathbf{N} \otimes \mathbf{N}$
q		q
n		n
q		q
n'		n'

In $\mathbf{N} \multimap \mathbf{N}$, O can only begin in the right-hand component. P may choose to make use of the left-hand component or not.

$\mathbf{N} \multimap \mathbf{N}$	or	$\mathbf{N} \multimap \mathbf{N}$
q		q
n		q
		n
		n'

LEMMA 2.7 (Switching condition). *If $sab \in P_{A \otimes B}$, with a and b in different components* (*i.e.*, *either a is a move from A and b is a move from B or vice versa*), *then b is an* O*-move. If $sab \in P_{A \multimap B}$ with a and b in different components, then b is a* P*-move.*

Proof. The simplest proof is to consider a notion of "state" for the compound game $A \otimes B$, which consists of three pieces of information: which player is to play in the A-component, which player is to play in the B-component, and which player is to play overall. The argument is best understood pictorially: see Fig. 1. At the outset, the state is (O, O), with O to play overall. So a move can be made by Opponent in either A or B, leading to the state (P, O) or (O, P), with P to play overall. At the next move, the projection condition forces P to play in the same component that O just played, because the other component is expecting an O-move, and the state then returns to the initial one. So we see that P can never switch components.

The argument for $A \multimap B$ is much the same: the starting state is (P, O), with O to play, because of the reversal of roles in the A-component. So initially O must play in B, giving state (P, P), with P to play. There are now two possibilities: if P plays in B, we return to the initial state, and if P plays in A we get to state (O, P) with O to play. Opponent is then forced to play in A, and we return to the intermediate state. Therefore O can never switch components. ▮

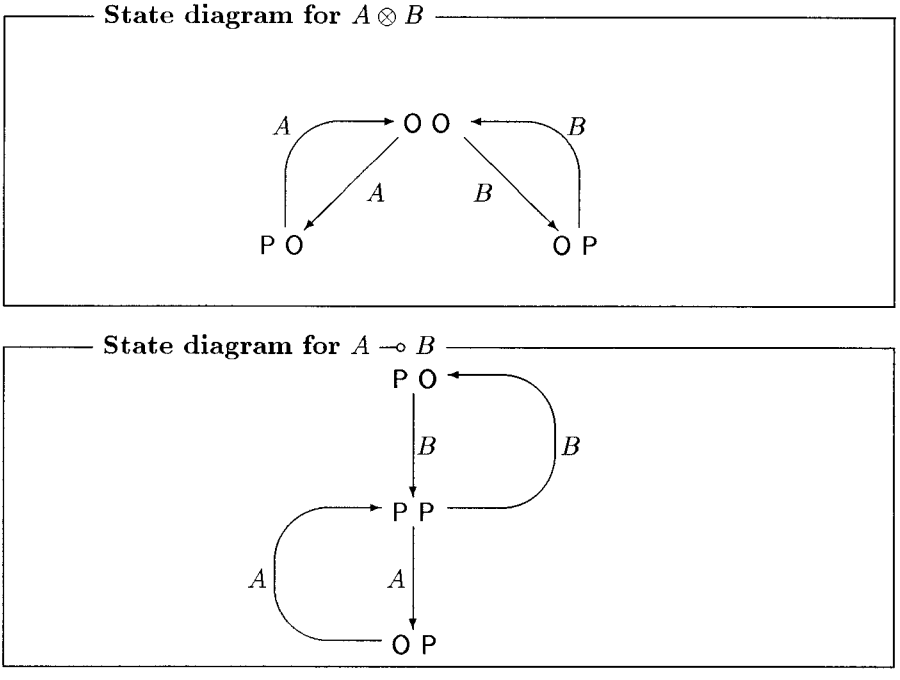


FIG. 1. The switching condition.

LEMMA 2.8 (O-view projection). *Let $s \in P_{A \multimap B}$. If s ends with a move in A then $\lfloor s \rfloor \upharpoonright A = \ulcorner s \upharpoonright A \urcorner$, and if s ends with a move in B then $\lfloor s \rfloor = \lfloor s \upharpoonright B \rfloor$.*

Proof. A routine induction on the length of s. We give one case of the inductive step for illustration. Suppose $s = s'ms''n$ where n is a P-move in A, justified by m, which is also in A. Then we calculate

$$\begin{aligned} \lfloor s \rfloor \upharpoonright A &= \lfloor s' \rfloor \upharpoonright A \cdot mn \\ &= \ulcorner s' \upharpoonright A \urcorner mn \qquad \text{by inductive hypotheses} \\ &= \ulcorner s'ms''n \upharpoonright A \urcorner, \end{aligned}$$

where the last step is justified by the change of O/P polarity when we restrict to A. ∎

2.2.2. *Strategies*

DEFINITION. A *strategy* σ for a game A is a nonempty set of even-length positions from P_A, satisfying

(s1) $sab \in \sigma \Rightarrow s \in \sigma$.

(s2) $sab, sac \in \sigma \Rightarrow b = c$, and the justifier of b is the same as that of c. In other words, the justified sequences sab and sac are identical. In the future, in circumstances such as these, we will omit mention of the justification pointers when it is clear how they should work: so in this case, when we say that $b = c$, it should be understood that the justification pointers are also the same.

Think of σ as a crib-book telling Player how to play in a give position: if $sab \in \sigma$, then at position sa, Player should play b. Condition (s2) above guarantees that there can be at most one entry for each position, so σ is *deterministic*. Note that σ may have no entry corresponding to certain positions; in that case, playing according to σ, Player makes no response. If σ is a strategy for a game A, we write $\sigma : A$.

The condition that σ be nonempty is equivalent to requiring that $\varepsilon \in \sigma$, so the smallest possible strategy for any game is the *empty strategy* $\{\varepsilon\}$. This strategy has no response to any O-move and so represents an "undefined" element of the type represented by a game; it is therefore denoted by $\bot$.

EXAMPLE. There is only one strategy for the empty game I, namely the empty strategy $\bot$.

For $\mathbf{N}$, there is the empty strategy, and also one strategy for each natural number n, namely $\{\varepsilon, qn\}$. In the game $\mathbf{N} \multimap \mathbf{N}$, there are many strategies: the empty strategy; the "constant" strategy for each n, given by $\{\varepsilon, qn\}$, which immediately answers the opening question with the number n; and for each partial function $f: \mathbb{N} \rightharpoonup \mathbb{N}$, a strategy which plays thus

$$\begin{array}{ll} \mathbf{N} \multimap & \mathbf{N} \\ & q \\ q & \\ n & \\ & f(n) \end{array}$$

the last move being played only if $f(n)$ is defined. Note that the identity function gives rise to a strategy which simply copies moves from one side of the arrow to the other. This can be extended to all games as follows.

The *identity* strategy for a game A is a strategy for $A \multimap A$ defined by

$$\mathsf{id}_A = \{s \in P_{A_1 \multimap A_2} \mid \forall t \sqsubseteq^{\text{even}} s.(t \upharpoonright A_1 = t \upharpoonright A_2)\}.$$

We use subscripts to distinguish the two occurrences of A and write $t \sqsubseteq^{\text{even}} s$ to mean that t is an even-length prefix of s.

All that id_A does is to copy the move made by Opponent in one copy of A to the other copy of A. The justifier for Player's move is the copy of the justifier of Opponent's move. It is easy to check that this does indeed define a strategy.

This sort of strategy, which merely copies moves between one component and another, is extremely common in game semantics. Indeed, in [AJ94], Abramsky and Jagadeesan show that all the strategies in their model of multiplicative linear logic have this "copycat" behaviour. We will see many more strategies of this kind; to borrow a phrase from Mac Lane, the slogan is "Copycat strategies arise everywhere."

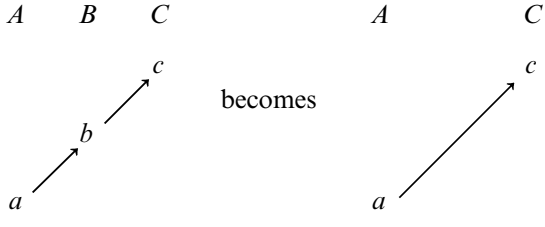
2.2.3. Composition

The categories we will work in have games as objects and strategies as morphisms. Therefore, given strategies $\sigma: A \multimap B$ and $\tau: B \multimap C$, we would like to compose them to form a strategy $\sigma; \tau: A \multimap C$. First, some auxiliary definitions are necessary.

DEFINITION. Let u be a sequence of moves from games A, B, and C together with justification pointers from all moves except those initial in C. Define $u \upharpoonright B, C$ to be the subsequence of u consisting of all moves from B and C; if a pointer from one of these points to a move of A, delete that pointer (this case will never arise in the sequences of interest). Similarly define $u \upharpoonright A, B$. We say that u is an *interaction sequence* of A, B, and C if $u \upharpoonright A, B \in P_{A \multimap B}$ and $u \upharpoonright B, C \in P_{B \multimap C}$. The set of all such sequences is written as $\mathsf{int}(A, B, C)$.

An important observation is that in an interaction sequence, switching between A, B, and C is *local*: there cannot be two consecutive moves one of which is in A and the other in C. Some moves of B must intervene. This can be proved in the same way as the switching condition, Lemma 2.7.

Suppose $u \in \mathsf{int}(A, B, C)$. A pointer from a C-move must be to another C-move, and a pointer from an A-move a must be either to another A-move or to an *initial* B-move, b, which in turn must have a pointer to an initial C-move, c. Define $u \upharpoonright A, C$ to be the subsequence of u consisting of all moves from A and C, except that in the case outlined above, the pointer from a is changed to point to c.



Given strategies $\sigma: A \multimap B$ and $\tau: B \multimap C$, define $\sigma \parallel \tau$ to be

$$\{u \in \mathsf{int}(A, B, C) \mid u \upharpoonright A, B \in \sigma \wedge u \upharpoonright B, C \in \tau\}.$$

So $\sigma \parallel \tau$ consists of sequences generated by playing σ and τ in parallel, communicating via B. When σ plays a move in B, it becomes a stimulus for τ to move and *vice versa*.

We extend the definition of interaction sequence to four games, forming the set $\mathsf{int}(A, B, C, D)$ in the obvious way, and define $\sigma \parallel \tau \parallel v$ similarly to above, for $\sigma: A \multimap B$, $\tau: B \multimap C$, and $v: C \multimap D$.

We are now ready to define the composite of two strategies.

DEFINITION. If $\sigma: A \multimap B$ and $\tau: B \multimap C$, define $\sigma; \tau: A \multimap C$ by

$$\sigma; \tau = \{u \upharpoonright A, C \mid u \in \sigma \parallel \tau\}.$$

If $s \in \sigma; \tau$, we refer to the $u \in \mathsf{int}(A, B, C)$ from which it arose as the *witness* of s and note that this witness is unique.

We now show that composition is well defined and associative. First, a definition which will be of use in the proofs to follow.

DEFINITION. Let $u \in \mathsf{int}(A, B, C)$. We make the following definitions.

- The *core* of u, written $\bar{u}$, is defined to be the subsequence of u obtained by removing all segments $m \cdots n$ from u, where m is an O-move in A or C, n is a P-move in A or C, all the intervening moves are in B, and neither m nor n appears in $\ulcorner u \upharpoonright A, C \urcorner$.

The core of a sequence can be defined recursively as follows.

$$\begin{aligned} \overline{ub} &= \bar{u}b, && \text{if } b \text{ is in } B \text{ or is a P-move in } A \text{ or } C \\ \overline{uc} &= c, && \text{if } c \text{ in an initial move in } C \\ \overline{u_1 m u_2 n} &= \overline{u_1} mn, && \text{if } n \text{ is an O-move in } A \text{ or } C \text{ justified by } m. \end{aligned}$$

- Let m be any move of u. Then the *component* of m is A, B if m is in A and B, C if m is in C. If m is in B, the component of m is A, B if m is an O-move in $u \upharpoonright A, B$, and is B, C otherwise. We will often use X to denote a component in this sense, and when we do so, Y denotes the other component.

- A *generalized* O-*move* of u is a move m which is an O-move in $u \upharpoonright A, C$, or is in B (so that it can be seen as an O-move in one component.)

The intuition behind the definition of component is that the component of a move m tells us which game is "active" when m is played. If the component is A, B then the $A \multimap B$ part is active and σ is the strategy with control. Otherwise $B \multimap C$ is active and τ has control. Observe that the component of a move in B is always different to that of the previous move. For if m is in B with component A, B, then it is an O-move when considered as being in $A \multimap B$. By the switching condition on $A \multimap B$, the previous move cannot be in A. If it is in B, then it is a P-move in $A \multimap B$ and hence has component B, C. If it is in C then it also has component B, C. This corresponds to the intuition that moves in B transfer control from one strategy to the other. Similarly, moves in A or C do not transfer control.

The following technical lemma establishes an important property relating the view of part of the core of u to that of u itself.

LEMMA 2.9. *Let $u \in \mathsf{int}(A, B, C)$ and let m be a generalised* O-*move of u with component X. If m is not initial in X, write n for the justifier of m in X and n^- for the move immediately before n.*

1. *If m is not initial in X, the move n^- is a generalised* O-*move with component X.*
2. *If m appears in $\bar{u}$ and m is not initial in X then both n and n^- appear in $\bar{u}$.*
3. *If m appears in $\bar{u}$ then $\ulcorner \bar{u}_{\leqslant m} \upharpoonright X \urcorner = \ulcorner u_{\leqslant m} \upharpoonright X \urcorner$.*

Proof. The first part is easily checked using the remarks above. We prove the remaining parts by induction on the length of $u_{\leqslant m}$. The case where m is initial in X is trivial and encompasses the base case. If m is not initial, there are two cases. If m is in A or C, it appears in $\bar{u} \upharpoonright A, C = \ulcorner u \upharpoonright A, C \urcorner$ by definition, and by the definition of view the justifier of each O-move in $\ulcorner u \upharpoonright A, C \urcorner$ is also in $\ulcorner u \upharpoonright A, C \urcorner$, so n is in $\bar{u}$. It is then easy to see that n^- is also in $\bar{u}$. If m is in B, it is an O-move in component X and hence a P-move in component Y. Therefore its justifier appears in $\ulcorner u_{<m} \upharpoonright Y \urcorner$ since $u \upharpoonright Y$ is legal. But the last move of $u_{<m}$ has component Y and appears in $\bar{u}$ trivially, so by the inductive hypothesis, $\ulcorner u_{<m} \upharpoonright Y \urcorner = \ulcorner \bar{u}_{<m} \upharpoonright Y \urcorner$. Therefore n is in $\bar{u}$ as required, and again it is easy to show that n^- is also in $\bar{u}$. Finally, we have

$$\begin{aligned} \ulcorner u_{\leqslant m} \upharpoonright X \urcorner &= \ulcorner u_{\leqslant n^-} \upharpoonright X \urcorner nm \\ &= \ulcorner \bar{u}_{\leqslant n^-} \upharpoonright X \urcorner nm && \text{by first part and IH} \\ &= \ulcorner \bar{u}_{\leqslant m} \upharpoonright X \urcorner && \text{by the above.} \end{aligned}$$

∎

We can now prove the result which ensures that composition is well defined.

LEMMA 2.10. *Let* $u \in \mathsf{int}(A, B, C)$. *Then* $u \upharpoonright A, C \in P_{A \multimap C}$.

Proof. It is clear that $u \upharpoonright A, C$ is a justified sequence, and it is not hard to show that it is alternating and satisfies the bracketing condition. The projection conditions are satisfied because $u \upharpoonright A, C \upharpoonright A = u \upharpoonright A, B \upharpoonright A$ and $u \upharpoonright A, B \in P_{A \multimap B}$ (and similarly for the projection onto C). So we just need to verify the visibility condition. Let m be a noninitial move of $u \upharpoonright A, C$. If m is an O-move in A, its justifier appears in $\llcorner u_{<m} \upharpoonright A, B \lrcorner \upharpoonright A$, and by Lemma 2.8,

$$\llcorner u_{<m} \upharpoonright A, B \lrcorner \upharpoonright A = \ulcorner u_{<m} \upharpoonright A, B \upharpoonright A \urcorner = \ulcorner u_{<m} \upharpoonright A, C \upharpoonright A \urcorner = \llcorner u_{<m} \upharpoonright A, C \lrcorner \upharpoonright A$$

so the visibility condition is satisfied at m. The case when m is an O-move in C is similar. If m is a P-move not initial in A, its justifier n appears in $\ulcorner u_{<m} \upharpoonright X \urcorner$ (where X is the component of m). But the last move of $u_{<m}$ is a generalised O-move in component X, so by the previous lemma, $\ulcorner u_{<m} \upharpoonright X \urcorner = \ulcorner \overline{u_{<m}} \upharpoonright X \urcorner$. So n is in $\overline{u_{<m}} \upharpoonright A, C = \ulcorner u_{<m} \upharpoonright A, C \urcorner$ as required. The only remaining case is when m is initial in A. It therefore has a pointer to a move n initial in B, and by the same argument as for the previous case, n appears in $\ulcorner \overline{u_{<m}} \upharpoonright A, B \urcorner$. We can similarly show that the justifier of n, an initial move in C, also appears in $\overline{u_{<m}}$ and hence in $\ulcorner u_{<m} \upharpoonright A, C \urcorner$; but this move is the justifier of m in $u \upharpoonright A, C$, so the proof is complete. ∎

PROPOSITION 2.11. *Composition is well defined; that is, if* $\sigma : A \multimap B$ *and* $\tau : B \multimap C$, *then* $\sigma ; \tau$ *is a strategy for* $A \multimap C$.

Proof. By Lemma 2.10, $\sigma ; \tau$ is a set of valid positions of $A \multimap C$, and it is easy to see that they are all of even length. It is also easy to check that conditions (s1) and (s2) hold for $\sigma ; \tau$, because they hold for σ and τ. So $\sigma ; \tau$ is a valid strategy. ∎

PROPOSITION 2.12. *Let* $\sigma: A \multimap B$, $\tau: B \multimap C$, *and* $v: C \multimap D$. *Then* $\sigma;(\tau ; v) = (\sigma;\tau); v$.

Proof. Let $s \in (\sigma;\tau); v$. Then by definition, $s = u \upharpoonright A, D$ for some $u \in (\sigma;\tau) \parallel v$. This means that $u \upharpoonright A, C \in \sigma;\tau$, so $u \upharpoonright A, C = v \upharpoonright A, C$ for some $v \in \sigma \parallel \tau$. Putting u and v together in the obvious way given us a sequence $w \in \sigma \parallel \tau \parallel v$ such that $w \upharpoonright A, B, C = v$, $w \upharpoonright C, D = u \upharpoonright C, D$, and $w \upharpoonright A, D = s$. Then $w \upharpoonright A, B, D \in \sigma \parallel (\tau ; v)$ (using Lemma 2.10 to guarantee that $w \upharpoonright B, D \in P_{B \multimap D}$) and hence $s = w \upharpoonright A, D \in \sigma;(\tau ; v)$. This establishes that $(\sigma;\tau); v \subseteq \sigma;(\tau ; v)$. The other inclusion is proved by a similar argument. ∎

Furthermore, the identity strategy really is an identity for composition, as can easily be checked.

LEMMA 2.13. *Let* $\sigma: A \multimap B$. *Then* $\mathsf{id}_A;\sigma = \sigma = \sigma;\mathsf{id}_B$.

2.2.4. Innocent Strategies

Recall that the view of a position is supposed to represent the currently relevant subsequence of moves. It follows that a strategy for Player should only be interested in the view of the position. We now introduce the class of strategies for which this is the case, namely the *innocent* ones. From now on we will only concern ourselves with innocent strategies.

DEFINITION. Given positions sab, $ta \in L_A$, where sab has even length and $\ulcorner sa \urcorner = \ulcorner ta \urcorner$, there is a unique extension of ta by the move b together with a justification pointer in such a way that $\ulcorner sab \urcorner = \ulcorner tab \urcorner$. Call this extension $\mathsf{match}(sab, ta)$. A strategy $\sigma: A$ is *innocent* if and only if it satisfies

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

In other words, the move and pointer played by an innocent strategy σ at a position sa is determined by the P-view $\ulcorner sa \urcorner$.

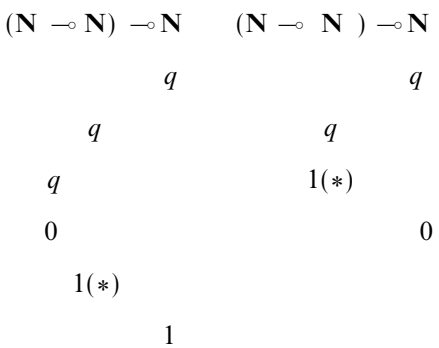
Some noninnocent strategies. The reason for imposing the condition of innocence is of course to eliminate certain unwanted strategies from consideration. Consider the following two plays in the game $\mathbf{N} \otimes \mathbf{N}$.

$\mathbf{N} \otimes \mathbf{N}$		$\mathbf{N} \otimes \mathbf{N}$	
$q(*)$			q
0			0
	q	$q(*)$	
	1	1	

In each case P responds with 0 to the first question asked by O and with 1 to the next question, regardless of which question comes first. No innocent strategy can

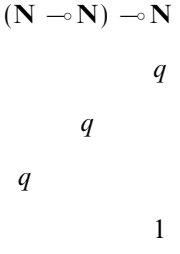
have this behaviour, because the views at the moves marked $(*)$ are identical—they each consist only of the left-hand q—but P's responses are different. Thus a simple form of "state-dependent" behaviour, in which the invocation of one copy of $\mathbf{N}$ influences the later behaviour in the other copy, is outlawed by innocence.

For a second example, consider the following two plays in the game $(\mathbf{N} \multimap \mathbf{N}) \multimap \mathbf{N}$.



Here P's final output depends on whether O chooses to interrogate the leftmost $\mathbf{N}$, that is, on whether the function argument is *strict* or not. No innocent strategy can contain these two plays, because the views at the moves marked $(*)$ are identical $(q \cdot q \cdot 1)$, but the responses are different. Thus an innocent strategy cannot have the behavior of such a "strictness test". This is a good thing: no program in a pure functional language has this behaviour either, so to obtain a definability result, and hence full abstraction, we must outlaw such strategies.

Aside: *the bracketing condition.* As a final example of a strategy which does not exist in our model, consider the following play which is also intended to provide a strictness test



Here P gives an answer to the rightmost q as soon as O interrogates the leftmost $\mathbf{N}$, that is, as soon as the input function reveals its strictness. Such a strategy does not exist in our model because it violates the *bracketing condition*: the first q is answered before both the second and the third. This is a key point a which the games model differs from the sequential algorithms model of functional programming [CCF94], which *does* allow this kind of behaviour; a full abstraction result for that model can be obtained with respect to a language extended with a control operator `catch`, which has just this form of non-well-bracketed semantics.

Composition of innocent strategies. A vital property of innocent strategies is that they are closed under composition, which we now aim to show.

Let us note first of all that the "nullary case" holds: the identity strategies are evidently innocent.

The following lemma will be of use later on.

LEMMA 2.14. *Let σ be an innocent strategy for a game A, and suppose $s \in \sigma$. Then for each initial m in s, $s \upharpoonright m \in \sigma$.*

Proof. The proof is by induction on the length of s; the base case is trivial. For the inductive step, suppose that $sab \in \sigma$ and let m be an initial move in s. If $sab \upharpoonright m = s \upharpoonright m$, the result holds by the inductive hypothesis. Otherwise, $sab \upharpoonright m = s \upharpoonright m \cdot ab$ and $s \upharpoonright m \in \sigma$ by the inductive hypothesis. Then since $s \upharpoonright m \cdot a \in P_A$, the fact that $\ulcorner s \upharpoonright m \cdot a \urcorner = \ulcorner sa \urcorner$ together with the innocence of σ give us that $s \upharpoonright m \cdot ab \in \sigma$ as required. ∎

We now set about showing that the composition of two innocent strategies is an innocent strategy. First, observe that an innocent strategy for a game A can be represented as a partial function from P-views of odd-length positions of A to P-moves, together with appropriate justification information. We refer to the function representing a strategy σ as the *view function* of σ, and we usually suppress the justification information when speaking of such functions.

We will need the following simple lemma.

LEMMA 2.15. *Let $u \in \mathsf{int}(A, B, C)$, and let m be a P-move in component X such that $um \upharpoonright X \in P_X$, where by P_X we mean $P_{A \multimap B}$ if X is A, B, and $P_{B \multimap C}$ otherwise. Then $um \upharpoonright Y \in P_Y$; in other words $um \in \mathsf{int}(A, B, C)$.*

We can now phrase the definition of composition in terms of view functions. First, some notation. Let $\sigma : A \multimap B$ and $\tau : B \multimap C$ and let $s \in \sigma ; \tau$ with witness u. For a move m of u, let X_m denote the component of m, and for a component X, let h_X denote the view function of σ if X is A, B and the view function of τ otherwise.

LEMMA 2.16. *If $sa \in P_{A \multimap C}$, then $sab \in \sigma ; \tau$ if and only if there exist $m_1, ..., m_k \in M_B$ such that*

$$h_{X_a}(\ulcorner ua \upharpoonright X_a \urcorner) = m_1$$

$$h_{X_{m_i}}(\ulcorner uam_1 \cdots m_i \upharpoonright X_{m_i} \urcorner) = m_{i+1} \qquad \text{for} \quad i = 1, ..., k-1$$

$$h_{X_k}(\ulcorner uam_1 \cdots m_k \upharpoonright X_{m_k} \urcorner) = b$$

and then the witness of sab is $uam_1 \cdots m_k b$.

Proof. Lemma 2.15 tells us that $uam_1 \cdots m_k b \in \mathsf{int}(A, B, C)$, and the rest follows. ∎

This lemma, together with Lemma 2.9(3), allows us to prove the following vital result.

LEMMA 2.17. *If* s, $t \in \sigma ; \tau$, *with witnesses* u *and* $v \in \mathsf{int}(A, B, C)$, *and* sa, $ta \in P_{A \multimap C}$ *with* $\ulcorner sa \urcorner = \ulcorner ta \urcorner$, *then* $\overline{ua} = \overline{va}$.

Proof. By induction on the length of the sequence ua. The case when a is initial, encompassing the base case, is trivial. If a is not initial, suppose that its justifier in both sa and ta is m, and let the move immediately preceding m in both sa and ta be n. (These moves are the same because $\ulcorner sa \urcorner = \ulcorner ta \urcorner$.) Then $ua = u_{\leqslant n} \cdot w_u \cdot m \cdots a$ and $va = v_{\leqslant n} \cdot w_v \cdot m \cdots a$, where w_u and w_v are sequences of moves in B. By the inductive hypothesis, $\overline{u_{\leqslant n}} = \overline{v_{\leqslant n}}$. By the definition of core, $\overline{ua} = \overline{u_{\leqslant n}} \cdot w_u \cdot ma$ and $\overline{va} = \overline{v_{\leqslant n}} \cdot w_v \cdot ma$, so we just need to show that $w_u = w_v$. But this follows from the previous lemma together with Lemma 2.9(3). ∎

We can at last show that the composition of two innocent strategies is innocent.

PROPOSITION 2.18. *If* $\sigma : A \multimap B$ *and* $\tau : B \multimap C$ *are innocent, then so is* $\sigma ; \tau$.

Proof. Suppose sab, $t \in \tau$, and $ta \in P_{A \multimap C}$ with $\ulcorner sa \urcorner = \ulcorner ta \urcorner$. We must show that $tab \in \sigma ; \tau$. By definition of $\sigma ; \tau$, s and t have witnesses u and $v \in \mathsf{int}(A, B, C)$ such that $u \upharpoonright A, C = s$, $v \upharpoonright A, C = t$ and $u, v \upharpoonright A, B \in \sigma$, $u, v \upharpoonright B, C \in \tau$. Since $sab \in \sigma ; \tau$ there exist $m_1, ..., m_k$ such that $uam_1 \cdots m_k b$ witnesses this, with each m_i a B-move. Since $\ulcorner sa \urcorner = \ulcorner ta \urcorner$, Lemma 2.17 tells us that $\overline{ua} = \overline{va}$.

Now repeated application of Lemma 2.17 together with Lemma 2.9(3) tells us that $vam_1 \cdots m_k b$ witnesses that $tab \in \sigma ; \tau$. ∎

2.3. The Category

We can now define a category $\mathcal{G}$ of games and innocent strategies as follows.

Objects : Games

Morphisms $\sigma : A \to B$: Innocent strategies for $A \multimap B$

Lemma 2.13 says that identities exist, Propositions 2.11 and 2.18 say that composition is well defined in this category, and Proposition 2.12 says that it is associative, so we have:

THEOREM 2.19. $\mathcal{G}$ *is a category.*

2.3.1. Monoidal Structure

We have already given the object part of the tensor product. We now describe the corresponding action on morphisms which makes tensor into a bifunctor and $\mathcal{G}$ into a symmetric monoidal category.

DEFINITION. Given $\sigma : A \to B$ and $\tau : C \to D$, define $\sigma \otimes \tau : (A \otimes C) \multimap (B \otimes D)$ by

$$\sigma \otimes \tau = \{ s \in L_{A \otimes C \multimap B \otimes D} \mid s \upharpoonright A, B \in \sigma \wedge s \upharpoonright C, D \in \tau \}.$$

The idea is that $\sigma \otimes \tau$ responds like σ to moves in A or B and like τ to moves in C or D—the two strategies are played "in parallel," with no communication between them.

It is clear that this defines a strategy for $A \otimes C \multimap B \otimes D$. In fact it is also innocent and $\otimes$ defines a bifunctor on $\mathcal{G}$. We can now define natural isomorphisms unit, assoc, and comm with components $\mathsf{unit}_A : A \otimes I \to A$, $\mathsf{assoc}_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$, and $\mathsf{comm}_{A,B} : A \otimes B \to B \otimes A$ given by the obvious copycat strategies—in each case the set of moves of the domain game is isomorphic to the set of moves of the codomain game. It is then trivial to verify the following.

PROPOSITION 2.20. *The structure described above makes $\mathcal{G}$ into a symmetric monoidal category.*

2.3.2. Closed Structure

To make $\mathcal{G}$ into a symmetric monoidal *closed* category, we need to show that each functor $- \otimes B$ has a (specified) right adjoint. Observe first that the only difference between games $A \otimes B \multimap C$ and $A \multimap (B \multimap C)$ is in the tagging of moves in the disjoint unions. Therefore

$$\begin{aligned}\mathcal{G}(A \otimes B, C) &= \{\sigma \mid \sigma \text{ is an innocent strategy for } A \otimes B \multimap C\} \\ &\cong \{\sigma \mid \sigma \text{ is an innocent strategy for } A \multimap (B \multimap C)\} \\ &= \mathcal{G}(A, B \multimap C).\end{aligned}$$

Denote this isomorphism by $\Lambda_B(-)$, and let $\mathsf{ev}_{A,B} = \Lambda_A^{-1}(\mathsf{id}_{A \multimap B})$. Explicitly, $\mathsf{ev}_{A,B}$ is just the copycat strategy for the game $(A \multimap B) \otimes A \multimap B$, copying moves between the two occurrences of A and between the two occurrences of B. It is easy to check that

$$\Lambda_B(\sigma) \otimes \mathsf{id}_B ; \mathsf{ev}_{B,C} = \sigma$$

for all $\sigma : A \otimes B \to C$, so we can conclude:

PROPOSITION 2.21. *With the structure described above, $\mathcal{G}$ is a symmetric monoidal closed category.*

2.3.3. Products

Given games A and B, define a game $A \& B$ as follows.

$$\begin{aligned}M_{A\&B} &= M_A + M_B \\ \lambda_{A\&B} &= [\lambda_A, \lambda_B] \\ \star \vdash_{A\&B} n &\Leftrightarrow \star \vdash_A n \vee \star \vdash_B n \\ m \vdash_{A\&B} n &\Leftrightarrow m \vdash_A n \vee m \vdash_B n \\ P_{A\&B} &= \{s \in L_{A\&B} \mid s \upharpoonright A \in P_A \wedge s \upharpoonright B = \varepsilon\} \\ &\quad \cup \{s \in L_{A\&B} \mid s \upharpoonright B \in P_B \wedge s \upharpoonright A = \varepsilon\}.\end{aligned}$$

Note that the arena part of the definition is exactly the same as that of $A \otimes B$. The valid positions are different: a position is valid in $A \,\&\, B$ iff it is wholly in A or in B, and it is valid there. It is clear that $A \,\&\, B$ is a well-defined game.

We can now define projections $\pi_1 : A \,\&\, B \to A$ and $\pi_2 : A \,\&\, B \to B$ by the obvious copycat strategies. Given $\sigma : C \to A$ and $\tau : C \to B$, define $\langle \sigma, \tau \rangle : C \to A \,\&\, B$ by

$$\langle \sigma, \tau \rangle = \{ s \in L_{C \multimap A \,\&\, B} \mid s \upharpoonright C, A \in \sigma \wedge s \upharpoonright B = \varepsilon \}$$
$$\cup \{ s \in L_{C \multimap A \,\&\, B} \mid s \upharpoonright C, B \in \tau \wedge s \upharpoonright A = \varepsilon \}.$$

So $\langle \sigma, \tau \rangle$ behaves like σ if O starts in A and like τ if O starts in B. It is easy to check that this is a well-defined innocent strategy and that $\langle \sigma, \tau \rangle ; \pi_1 = \sigma$ and $\langle \sigma, \tau \rangle ; \pi_2 = \tau$. Furthermore, $\langle \sigma, \tau \rangle$ is unique in satisfying these conditions, so & defines a categorical product.

PROPOSITION 2.22. *$A \,\&\, B$ is the product of A and B, with projections given by π_1 and π_2.*

We have shown that $\mathcal{G}$ is a symmetric monoidal closed category with products. To obtain the full power of Cartesian closure, however, we need to see how to model the "of course" exponential (!) of linear logic. This is considered in the next section.

2.4. Exponential

We describe an attempt to model the exponential which does not quite work (because dereliction is not available at all types), but gives just enough to allow us to build a Cartesian closed category suitable for modelling programming languages.

DEFINITION. Given a game A, define the game $!A$ as follows.

$$M_{!A} = M_A$$
$$\lambda_{!A} = \lambda_A$$
$$\vdash_{!A} = \vdash_A$$
$$P_{!A} = \{ s \in L_{!A} \mid \text{for each initial move } m, s \upharpoonright m \in P_A \}.$$

The game $!A$ is clearly well-defined. The idea is that a play in $!A$ consists of a number of interwoven plays of A. Opponent switches between the different "threads" of play, and only one thread is visible to Player at a time. Note that any valid play of A is automatically a valid play of $!A$: this is the reason for insisting that the set of valid plays of a game is closed under taking the restriction to moves hereditarily justified by some set.

2.4.1. Promotion

Given a map $\sigma : !A \to B$, define its promotion $\sigma^\dagger : !A \to !B$ by

$$\sigma^\dagger = \{ s \in L_{!A \multimap !B} \mid \text{for all initial } m, s \upharpoonright m \in \sigma \}.$$

Note how similar the definition of $\sigma^\dagger$ from σ is to that of the game $!B$ from B.

LEMMA 2.23. *$\sigma^\dagger$ is an innocent strategy for $!A \multimap !B$.*

Proof. The response of $\sigma^\dagger$ at a position sa is defined to be the response of σ to $sa \upharpoonright m$, where m is the initial move which hereditarily justifies a. In turn this is determined by $\ulcorner sa \upharpoonright m \urcorner$, since σ is innocent; but $\ulcorner sa \upharpoonright m \urcorner = \ulcorner sa \urcorner$ so the response of $\sigma^\dagger$ depends only on the view, which is to say that $\sigma^\dagger$ is innocent. ∎

LEMMA 2.24. *If $\sigma : !A \to B$ and $\tau : !B \to C$, then $\sigma^\dagger ; \tau^\dagger = (\sigma^\dagger ; \tau)^\dagger$.*

Proof. Both $\sigma^\dagger ; \tau^\dagger$ and $(\sigma^\dagger ; \tau)^\dagger$ are innocent strategies and so are determined by their view-functions. Since P-views contain at most one initial move, these strategies are also determined by those plays containing at most one initial move. By definition of promotion, those of $(\sigma^\dagger ; \tau)^\dagger$ are exactly the plays of $\sigma^\dagger ; \tau$, and by definition of composition, a play s of $\sigma^\dagger ; \tau^\dagger$ which contains at most one initial move arises from a play s_1 of $\sigma^\dagger$ (possibly with many initial moves) and a play s_2 of $\tau^\dagger$ with at most one initial move. Again by definition of promotion, $s_2 \in \tau$ and hence $s \in \sigma^\dagger ; \tau$. The required result follows. ∎

2.4.2. *Dereliction*

If ! were to be a comonad, there would be a map $\mathsf{der}_A : !A \to A$ satisfying

$$\mathsf{der}_A^\dagger = \mathsf{id}_{!A}$$
$$\sigma^\dagger ; \mathsf{der}_A = \sigma$$

for all appropriate σ and for all objects A. This, together with Lemma 2.24 above, would make $(!, \mathsf{der}, (-)^\dagger)$ into a co-Kleisli triple. The first condition above also tells us what the view-function of der_A would have to be: it can only be the same as that of $\mathsf{id}_{!A}$, which is in turn the same as that of id_A. So the strategy would simply copy moves from the right-hand occurrence of A to a single thread of A in the left-hand $!A$. Unfortunately, this does not define a valid strategy for all objects. We give an example to demonstrate this fact. Consider the game $\mathbf{N} \multimap !\mathbf{N}$. If dereliction were possible, it would contain the following play.

$$!(\mathbf{N} \multimap !\mathbf{N}) \multimap (\mathbf{N} \multimap !\mathbf{N})$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad q$$
$$\qquad\qquad q$$
$$q$$
$$\qquad\qquad\qquad\qquad q$$
$$\qquad\qquad\qquad\qquad 3$$
$$3$$
$$\qquad\qquad 4$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 4$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad q \qquad (*)$$
$$\qquad\qquad q$$

At this point, it is valid for Opponent to play q in the extreme left-hand $\mathbf{N}$, but it would not be valid for Player to copy that move back to the right-hand side. The problem is in the idempotence of !. The instance of q marked $(*)$ is the start of a fresh thread in $!\mathbf{N}$, but its copy on the left-hand side can be interpreted as the start of a fresh thread of the whole of $!(\mathbf{N} \multimap !\mathbf{N})$.

There is a subclass of games for which this problem cannot arise, however: the *well opened* games.

DEFINITION. A game A is *well opened* iff for all $sm \in P_A$ with m initial, $s = \varepsilon$.

In a well opened game, initial moves can only happen at the first move. Note that if B is well opened then so in $A \multimap B$ for any game A, so while $!A$ is not well opened except in pathological cases, the game $!A \multimap B$ is well opened whenever B is. We are going to construct a Cartesian closed category in which all games are well opened and exponentials (in the ordinary sense, not the linear logic one) are given by $!A \multimap B$, so this observation is important.

We can now define the map der_A for each well opened game A as described above and check that it satisfies the conditions we expect from it, when it is defined.

LEMMA 2.25. *If B is well opened and $\sigma : !A \to B$ then $\sigma^\dagger ; \mathsf{der}_B = \sigma$ and $\mathsf{der}_B^\dagger = \mathsf{id}_{!B}$.*

Finally, a lemma that will be useful in the proof of full abstraction to follow.

LEMMA 2.26 (Bang lemma). *If B is well opened and $\sigma : !A \to !B$ then $\sigma = (\sigma ; \mathsf{der}_B)^\dagger$.*

Proof. Let $s \in P_{!A \multimap !B}$. Then $\ulcorner s \urcorner = \ulcorner s \upharpoonright m \urcorner$ for some initial B-move m, and its easy to check that $s \upharpoonright m \in P_{!A \multimap B}$. So the view function of the innocent strategy σ determines a strategy for $!A \multimap B$, which is of course $\sigma ; \mathsf{der}$. So the view function of $(\sigma ; \mathsf{der})^\dagger$ is the same as that of σ, as required. ∎

2.4.3. *Contraction*

The final piece of structure for the exponential is the comonoid structure, that is to say a map $\mathsf{con}_A : !A \to !A \otimes !A$ for each object A. As is to be expected, this can be defined by a copycat strategy, which copies moves from either occurrence of $!A$ on the right in to the $!A$ on the left, and back, respecting the justification pointers. In other words, the two plays on the right are interwoven on the left.

To be more explicit, for any $s \in P_{!A_0 \multimap !A_1 \otimes !A_2}$, let I be the set of occurrences of initial moves in A_1 and J be the set of occurrences of initial moves in A_2. Let $s_1 = s \upharpoonright I$ and $s_2 = s \upharpoonright J$. Then define

$$\mathsf{con}_A = \{ s \in P_{!A_0 \multimap !A_1 \otimes !A_2} \mid \forall t \sqsubseteq^{\text{even}} s . (t_1 \upharpoonright !A_0 = t_1 \upharpoonright !A_1) \wedge (t_2 \upharpoonright !A_0 = t_2 \upharpoonright !A_2) \}.$$

Contraction has the following naturality property, which is easy to verify.

LEMMA 2.27. *For any* $\sigma : !A \to B$

$$\begin{array}{ccc} !A & \xrightarrow{\mathsf{con}_A} & !A \otimes !A \\ {\scriptstyle \sigma^\dagger}\downarrow & & \downarrow{\scriptstyle \sigma^\dagger \otimes \sigma^\dagger} \\ !B & \xrightarrow[\mathsf{con}_B]{} & !B \otimes !B \end{array}$$

2.5. A Cartesian Closed Category

We can now define a Cartesian closed category of games, the *intensional category* $\mathscr{I}$, as follows.

$$\text{Objects} : \text{Well-opened games}$$

$$\text{Morphisms } \sigma : A \to B : \text{Innocent strategies for } !A \multimap B$$

We need to say what composition is and what the identities are. For any well opened game A, the strategy $\mathsf{der}_A : !A \multimap A$ is the identity map on A, and given morphisms $\sigma : A \to B$ and $\tau : B \to C$, that is to say strategies $\sigma : !A \multimap B$ and $\tau : !B \multimap C$, we define the composite morphism $\sigma ; \tau : A \to C$ to be $\sigma^\dagger ; \tau$. Lemma 2.25 ensures that dereliction really is the identity for composition, and Lemma 2.24 can be used to show that composition is associative, so we do indeed have a category.

For products, note that if A and B are well opened then so is $A \,\&\, B$. Define projections $\mathsf{fst} : A \,\&\, B \to A$ and $\mathsf{snd} : A \,\&\, B \to B$ to be the strategies $\mathsf{der}_{A \,\&\, B} ; \pi_1$ and $\mathsf{der}_{A \,\&\, B} ; \pi_2$, respectively. Pairing can be defined exactly as in $\mathscr{G}$ and works for the same reasons.

For the closed structure, observe that $!(A \,\&\, B) = !A \otimes !B$. Therefore

$$\begin{aligned} \mathscr{I}(A \,\&\, B, C) &= \mathscr{G}(!(A \,\&\, B), C) \\ &= \mathscr{G}(!A \otimes !B, C) \\ &\cong \mathscr{G}(!A, !B \multimap C) \\ &= \mathscr{I}(A, !B \multimap C). \end{aligned}$$

So we can define $A \Rightarrow B$ to be the game $!A \multimap B$, giving well-defined exponentials. $\mathscr{I}$ is therefore a Cartesian closed category.

2.5.1. Functional Representation of Innocent Strategies

We have previously made use of the fact that innocent strategies can be represented as partial functions of type

$$\mathsf{P}\text{-views of odd-length positions} \rightharpoonup \mathsf{P}\text{-moves}.$$

We now make this connection more precise. Fix a game A, and let f be such a partial function. Define

$$T_0(f) = \{\varepsilon\}$$
$$T_{n+1}(f) = \{sab \mid s \in T_n(f), sa \in P_A, f(\ulcorner sa \urcorner) = b\}$$
$$\mathsf{traces}(f) = \bigcup_n T_n(f).$$

Say that f is *safe* if $\mathsf{traces}(f) \subseteq P_A$; and say that f is *saturated* if

$$f(\ulcorner sa \urcorner) = b \Leftrightarrow \exists tab \in \mathsf{traces}(f). \ulcorner ta \urcorner = \ulcorner sa \urcorner.$$

So if f is safe, then $\mathsf{traces}(f)$ is an innocent strategy for A, and if f is saturated, then every entry in its graph can be "used" by the strategy $\mathsf{traces}(f)$, so that the view function of $\mathsf{traces}(f)$ is f.

Given a set S of even-length positions, define a partial function $\mathsf{fun}(S)$ by

$$\mathsf{fun}(S)(\ulcorner sa \urcorner) = b \Leftrightarrow \exists tab \in S. \ulcorner ta \urcorner = \ulcorner sa \urcorner.$$

Then $\mathsf{fun}(\sigma)$ is the view function of a strategy σ and is clearly safe and saturated. Furthermore, for any innocent strategy σ and any safe and saturated function f, we have

$$\mathsf{traces}(\mathsf{fun}(\sigma)) = \sigma$$
$$\mathsf{fun}(\mathsf{traces}(f)) = f.$$

We have shown the following:

LEMMA 2.28. *There is a one-to-one correspondence between innocent strategies and safe, saturated view functions.*

2.5.2. Order Enrichment

There is an obvious order on strategies, namely subset inclusion. We shall show that this ordering makes $\mathscr{G}$ and $\mathscr{I}$ into cpo-enriched categories. First, observe that if σ and τ are innocent strategies for some game A, then

$$\sigma \subseteq \tau \Leftrightarrow \mathsf{fun}(\sigma) \subseteq \mathsf{fun}(\tau)$$

so we can move freely between the "set of traces" and the functional representation of innocent strategies. For the purposes of studying the ordering, the functional representation is more useful, because as we shall see, the compact strategies are precisely those whose view functions are finite.

Let Δ be a directed set of innocent strategies for A. It is easy to check that $\bigcup \Delta$ is again an innocent strategy, so the hom-sets in $\mathscr{G}$ and $\mathscr{I}$ are cpos (with least element the empty strategy, $\{\varepsilon\}$). Composition in $\mathscr{G}$ is easily seen to be continuous, and since $\mathsf{fun}(\sigma) = \mathsf{fun}(\sigma^\dagger)$, promotion is continuous, and therefore composition in $\mathscr{I}$ is continuous too. This shows that $\mathscr{G}$ and $\mathscr{I}$ are cpo-enriched, and it is clear that any strategy with finite view function must be compact. We now show that no strategy whose view-function is infinite is compact.

Let σ be an innocent strategy. It is clear that the set

$$\Delta = \{\tau \mid \tau \subseteq \sigma, \mathsf{fun}(\tau) \text{ finite}\}$$

is directed and has σ as an upper bound. We claim that σ is in fact the supremum of Δ. For any even-length sequence s, define $\mathsf{evenpref}(s)$ to be the set of even-length prefixes of s, i.e.,

$$\mathsf{evenpref}(\varepsilon) = \{\varepsilon\}$$
$$\mathsf{evenpref}(sab) = \mathsf{evenpref}(s) \cup \{sab\}.$$

Then for any $s \in \sigma$, the partial function $\mathsf{fun}(\mathsf{evenpref}(s))$ is safe and saturated and hence gives rise to an innocent strategy σ_s with finite view function. So we have $s \in \sigma_s \in \Delta$, so that $\sigma \subseteq \bigcup \Delta$, and therefore $\sigma = \bigcup \Delta$.

We now know that any strategy is the supremum of a set of strategies with finite view functions, so no strategy with infinite view function is compact. We have therefore shown that the compact strategies are precisely those with finite view function and further that the set of strategies for a game A forms an algebraic cpo. In fact we can go further and show that it is a dI-domain, but since we will make no use of this in what follows, we omit the details.

PROPOSITION 2.29. *The categories $\mathcal{G}$ and $\mathcal{I}$ are both dI-domain enriched; the compact maps are those strategies σ such that* $\mathsf{fun}(\sigma)$ *is finite.*

DEFINITION. Given a strategy σ, write $|\sigma|$ for the cardinality of the view function of σ, considered as a set of pairs. So σ is compact if and only if $|\sigma|$ is finite.

2.6. The Extensional Category

The category $\mathcal{I}$ has many properties desirable for modelling sequential languages. It is Cartesian closed, and each map is a strategy which is a kind of "sequential computation." However, the intensional nature of the maps means that they are not really *functions*. Here we remedy this by constructing a new category $\mathcal{E}$ as the quotient of $\mathcal{I}$ with respect to the intrinsic preorder. The development in this section follows closely that of Abramsky *et al.* [AJM97].

The category $\mathcal{E}$ is Cartesian closed and sequential, but is also well pointed—a categorical formalization of being a category of functions. It should be noted that while one can define the intrinsic preorder on any Cartesian closed category and take the quotient as we do here, the fact that $\mathcal{E}$ is well pointed is a result of the particular structure of $\mathcal{I}$.

First, we define a game Σ as follows.

$$M_\Sigma = \{q, a\}$$
$$\lambda_\Sigma(q) = \mathsf{OQ}$$
$$\lambda_\Sigma(a) = \mathsf{PA}$$
$$\vdash_\Sigma = \{(\star, q), (q, a)\}$$
$$P_\Sigma = \{\varepsilon, q, qa\}.$$

There are only two strategies for Σ, namely the empty strategy and

$$\top = \{\varepsilon, qa\}.$$

Of course both these strategies are innocent.

We can now define the intrinsic preorder $\preccurlyeq$ on each homset of $\mathscr{I}$.

DEFINITION. Given $f,\ g : A \to B$, write $f \preccurlyeq g$ if and only if for all maps $\alpha : [A \Rightarrow B] \to \Sigma$, if '$f$' ; $\alpha = \top$ then 'g' ; $\alpha = \top$.

The map α in the above definition should be thought of as a test on the maps f and g, where f passes the test α if 'f' ; $\alpha = \top$. Then $f \preccurlyeq g$ just means that any test passed by f is passed by g too.

The category $\mathscr{E}$ is defined as the quotient of $\mathscr{I}$ by $\preccurlyeq$. Let $\approx$ be the equivalence relation on strategies defined by

$$\sigma \approx \tau \Leftrightarrow \sigma \preccurlyeq \tau \wedge \tau \preccurlyeq \sigma.$$

We write the equivalence class of a strategy σ as $[\sigma]$. Objects of $\mathscr{E}$ are those of $\mathscr{I}$, i.e., well opened games, and morphisms from A to B are equivalence classes of such morphisms from $\mathscr{I}$, that is equivalence classes of innocent strategies for $!A \multimap B$. The identity on A is given by $[\mathsf{id}_A]$, and the composite of $[\sigma]$ and $[\tau]$ is defined to be $[\sigma;\tau]$. It is straightforward to check that these are well defined and that $\mathscr{E}$ is a pointed-poset-enriched Cartesian closed category, the partial order $\leqslant$ on each homset being that induced by $\preccurlyeq$. In the remainder of the paper, we will often identify a map in $\mathscr{E}$ with a representant strategy, without comment.

The following two lemmas are useful in the analysis of the properties of $\preccurlyeq$ and hence of $\leqslant$. They describe properties of the test α, seen as a map in $\mathscr{G}$, rather than in $\mathscr{I}$. In what follows, we will frequently identify the game $I \multimap A$ with A and use strategies of A as strategies for $I \multimap A$ and so on. Note that because $I = !I$, a strategy for A can also be considered a map from I to A in $\mathscr{I}$.

LEMMA 2.30 (Separation of head occurrence). *Let A be any well opened game and B be a well opened game with only one possible first move q. Let $\alpha : !A \multimap B$ respond to the initial question q with a move m in $!A$. Then there exists a strategy $\alpha' : !A \otimes A \multimap B$ which responds to q with m in the separate tensor factor A, such that the following diagram (in $\mathscr{G}$) commutes.*

$$\begin{array}{ccc} !A & \xrightarrow{\mathsf{con}_A} & !A \otimes !A \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle \mathsf{id} \otimes \mathsf{der}} \\ B & \xleftarrow[\alpha']{} & !A \otimes A \end{array}$$

Proof. Simply relabel the moves of α so that all moves justified by the first m now occur in the separate tensor factor of A. It is easy to check that this is an innocent strategy and that it behaves as claimed. ∎

LEMMA 2.31 (Function space test decomposition). *Let* $\alpha: (A \multimap B) \multimap \Sigma$ *be an innocent strategy, where* B *is well opened. Then there exist strategies* $\alpha_1: I \multimap A$ *and* $\alpha_2: B \multimap \Sigma$ *such that* $\alpha_1 \multimap \alpha_2 = \alpha$ (*identifying the game* $I \multimap \Sigma$ *with* Σ).

Proof. Consider the possible switches of component that Player can make during a play of $(A \multimap B) \multimap \Sigma$. The switching condition immediately tells us that Player can only respond to the initial move in Σ in either Σ or B, to a move in B only in B or Σ, and to a move in A only in A or in Σ. But when Opponent has just played a move in A, there must be an odd number of moves in B, and therefore an unanswered question in B, so Player cannot respond by playing the answer a in Σ, because that would violate the bracketing condition. So Player can only respond to a move in A by another move in A.

Let s be a position of the game whose last move is an O-move in B. Then it is easy to see that $\ulcorner s \urcorner = \ulcorner s \upharpoonright B, \Sigma \urcorner$. If the last move of s is an O-move in A, however, then $\ulcorner s \urcorner = q \cdot b \cdot \ulcorner s \upharpoonright A \urcorner$ where b is the first move played in B. Therefore the play of α in B and Σ determines an innocent strategy α_2 for $B \multimap \Sigma$, namely

$$\alpha_2 = \{ s \in P_{B \multimap \Sigma} \mid s \in \alpha \},$$

and the play in A determines a strategy α_1 for A, given by

$$\alpha_1 = \{ s \in P_A \mid qbs \in \alpha \},$$

and these strategies clearly behave as stated. ∎

This lemma shows that testing a strategy of function type corresponds to supplying it with an argument and testing the output. But the lemma only applies to *linear* tests, i.e., maps from $A \multimap B$ to Σ in the category $\mathcal{G}$, while the intrinsic preorder is concerned with tests in $\mathcal{I}$, which are maps from $!(A \multimap B)$ to Σ in $\mathcal{G}$. However, the following lemma rectifies this situation, by showing that linear tests suffice.

LEMMA 2.32 (Linear tests suffice). *Let* σ *and* τ *be strategies for a well-opened game* A. *Then* $\sigma \preccurlyeq \tau$ *if and only if for every* $\alpha: A \multimap \Sigma$,

$$\sigma ; \alpha = \top \Rightarrow \tau ; \alpha = \top.$$

Proof. If $\sigma \preccurlyeq \tau$ then any strategy $\alpha: A \multimap \Sigma$ gives rise to a test β in $\mathcal{I}$ by the strategy $\mathsf{der}_A ; \alpha: !A \multimap \Sigma$, and then $\sigma ; \beta = \sigma^\dagger ; \mathsf{der} ; \alpha = \sigma ; \alpha$, and similarly for τ, so it follows that

$$\sigma ; \alpha = \top \Rightarrow \tau ; \alpha = \top.$$

For the converse, suppose that this condition holds. We must show that for any $\alpha: !A \multimap \Sigma$, if $\sigma^\dagger ; \alpha = \top$ then $\tau^\dagger ; \alpha = \top$. This we shall do by induction on the number of initial A-moves occurring in $\sigma^\dagger \parallel \alpha$, supposing that $\sigma^\dagger ; \alpha = \top$. For the base case, if no initial A-moves occur, then α must be the constantly $\top$ strategy, and $\tau^\dagger ; \alpha = \top$ trivially. For the inductive step, suppose that $n+1$ initial A-moves occur.

We know that α responds to the initial question in Σ by playing an initial A-move m, so by separation of head occurrence, Lemma 2.30, we obtain a strategy $\alpha' : !A \otimes A \multimap \Sigma$, which plays this move m in the separate tensor factor A, and furthermore satisfies

$$\sigma^\dagger \otimes \sigma ; \alpha' = \top.$$

Currying, we obtain $\Lambda^{-1}(\sigma ; \Lambda(\alpha')) : !A \multimap \Sigma$ such that

$$\sigma^\dagger ; \Lambda^{-1}(\sigma ; \Lambda(\alpha')) = \top$$

using only n initial A-moves. So by the inductive hypothesis,

$$\tau^\dagger ; \Lambda^{-1}(\sigma ; \Lambda(\alpha')) = \top.$$

Currying this the other way gives

$$\sigma ; \Lambda^{-1}(\tau^\dagger ; \Lambda(\alpha')) = \top,$$

so by hypothesis,

$$\tau ; \Lambda^{-1}(\tau^\dagger ; \Lambda(\alpha')) = \top.$$

Finally, uncurrying this gives

$$\tau^\dagger \otimes \tau ; \alpha' = \top$$

which by the property of α' tells us that $\tau^\dagger ; \alpha = \top$ as required. ∎

LEMMA 2.33 (Function space extensionality). *Let σ and $\tau : A \to B$ in $\mathscr{I}$, i.e., σ and τ are maps from $!A$ to B in $\mathscr{G}$. Then $\sigma \preccurlyeq \tau$ if and only if for all $\alpha : I \multimap A$ and all $\beta : B \multimap \Sigma$.*

$$\alpha^\dagger ; \sigma ; \beta = \top \Rightarrow \alpha^\dagger ; \tau ; \beta = \top.$$

Proof. Suppose $\sigma \preccurlyeq \tau$ and let α and β be as above, such that $\alpha^\dagger ; \sigma ; \beta = \top$. Then $\alpha^\dagger \multimap \beta$ gives us a strategy of type $(!A \multimap B) \multimap \Sigma$, and hence, by dereliction, a test on $A \Rightarrow B$ in $\mathscr{I}$. But

$$\text{`}\sigma\text{'};(\mathsf{der} ; \alpha^\dagger \multimap \beta) = \alpha^\dagger ; \sigma ; \beta = \top,$$

so since $\sigma \preccurlyeq \tau$,

$$\text{`}\tau\text{'};(\mathsf{der} ; \alpha^\dagger \multimap \beta) = \top$$

and therefore

$$\alpha^\dagger ; \tau ; \beta = \top.$$

For the converse, suppose that for all $\alpha: I \multimap A$ and all $\beta: B \multimap \Sigma$

$$\alpha^\dagger ; \sigma ; \beta = \top \Rightarrow \alpha^\dagger ; \tau ; \beta = \top,$$

and let $\gamma: (!A \multimap B) \multimap \Sigma$ be a (linear) test strategy such that

$$\sigma ; \gamma = \top.$$

By Lemma 2.31, γ can be written as $\gamma_1 \multimap \gamma_2$, and then we have

$$\gamma_1 ; \sigma ; \gamma_2 = \top.$$

By the Bang lemma (Lemma 2.26), $\gamma_1 = (\gamma_1 ; \mathsf{der})^\dagger$, so we can apply the hypothesis to obtain

$$\gamma_1 ; \tau ; \gamma_2 = \top,$$

and hence

$$\tau ; \gamma = \top.$$

We have shown that if σ passes a linear test, then so does τ ; but by Lemma 2.32, linear tests suffice to show that $\sigma \preccurlyeq \tau$, so the proof is complete. ▮

This is a very important lemma. It shows that $\mathscr{E}$, the quotient of $\mathscr{I}$ by $\preccurlyeq$, is well pointed, so that it can be considered to be a category of *functions*: we have succeeded in constructing a Cartesian closed category of sequential functions.

2.6.1. Order-Theoretic Properties of $\mathscr{E}$: Rationality

Recall that the intensional category $\mathscr{I}$ is enriched over dI-domains, providing rich order-theoretic properties which allow, among other things, the construction of fixed points of endomorphisms as least upper bounds of chains. We have already noted that $\mathscr{E}$ is a pointed-poset-enriched CCC, meaning that each homset has a pointed partial order structure given by $\leqslant$ (we write the least element of the homset $\mathscr{E}(A, B)$ as $\perp_{A,B}$) with respect to which composition, currying, and pairing are monotone. In addition, for any $f: A \to B$ we have that $f ; \perp_{B,C} = \perp_{B,C}$. The question of whether these posets are in fact cpos is still open; however, $\mathscr{E}$ does satisfy a certain weaker completeness property called *rationality* [AJM97] which is sufficient for our purposes.

A *rational category* is a pointed-posed-enriched CCC such that for any $f: A \to A$ the chain

$$f^{(0)} = \perp : 1 \to A$$
$$f^{(n+1)} = f^{(n)} ; f$$

has a least upper bound $f^\nabla : 1 \to A$, and for any $g: A \to B$,

$$f^\nabla ; g = \bigsqcup (f^{(n)} ; g).$$

It is easy to check that $\mathscr{E}$ is indeed rational.

In a rational category, every endomorphism $f : A \to A$ has at least fixed point, calculated as above, and all maps preserve the least upper bounds of chains calculated by iteration of endomorphisms. In the remainder of the paper, we will make use of these properties of $\mathscr{E}$ without further comment. Occasionally we will need to consider lubs of chains constructed repeated applications of a functor F. This can be justified provided the functor is *closed*, meaning that its action can be internalized as a map

$$[A \Rightarrow B] \to [FA \Rightarrow FB]$$

in $\mathscr{E}$. Rationality of $\mathscr{E}$ then implies that the functor F is well behaved with respect to the order. It is automatic that the product and exponential functors in a Cartesian closed category are closed in this way; in fact all the functors we need to consider are closed. For a careful treatment of these issues, see [McC98].

2.7. Sums

The motivation for introducing a new notion of game is to extend the scope of game semantics to handle sum types. In this section we describe a notion of sum of games which is particularly well behaved in the extensional category $\mathscr{E}$ and which we will later use to model the sum type of **FPC**.

DEFINITION. Given games A and B, define $A + B$ by

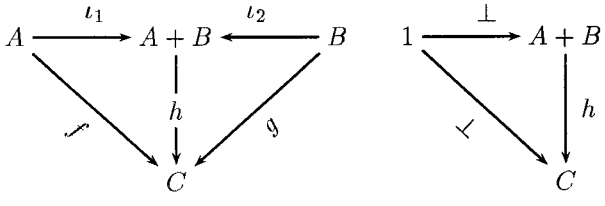
$$\begin{aligned}
M_{A+B} &= M_A + M_B + \{q, l, r\} \\
\lambda_{A+B}(q) &= \mathsf{OQ} \\
\lambda_{A+B}(l) &= \mathsf{PA} \\
\lambda_{A+B}(r) &= \mathsf{PA} \\
\lambda_{A+B}(a) &= \lambda_A(a) \\
\lambda_{A+B}(b) &= \lambda_B(b) \\
\star &\vdash_{A+B} q \\
q &\vdash_{A+B} l \\
q &\vdash_{A+B} r \\
l \vdash_{A+B} a &\Leftrightarrow \star \vdash_A a \\
r \vdash_{A+B} b &\Leftrightarrow \star \vdash_B b \\
a \vdash_{A+B} a' &\Leftrightarrow a \vdash_A a' \\
b \vdash_{A+B} b' &\Leftrightarrow b \vdash_B b' \\
P_{A+B} &= \{\varepsilon, q\} \cup \{qls \mid s \in P_A\} \cup \{qrs \mid s \in P_B\}.
\end{aligned}$$

In the above, a and a' range over M_A while b and b' range over M_B.

A play of $A + B$ consists of an initial question q by Opponent, followed by either the answer l and a play of A or the answer r and a play of B. The initial moves of A and B are justified by the answers l and r, respectively. (This construction is not possible in the original category of Hyland and Ong [HO97] because questions can only be justified by other questions in their approach. It was, however, suggested by Abramsky and Jagadeesan in the closing sections of their paper on multiplicative linear logic as a weak interpretation of the additive disjunction [AJ94].)

In the category $\mathscr{E}$, this sum construction enjoys a universal property as follows.

DEFINITION. Let $\mathbf{C}$ be a ppo-enriched category. A *strict sum* of objects A and B is an object $A + B$ together with two maps $\iota_1 : A \to A + B$ and $\iota_2 : B \to A + B$ such that for any $f : A \to C$ and $g : B \to C$ there exists a unique $h : A + B \to C$ such that the following two diagrams commute.



As usual for universal properties, the object $A + B$ is unique up to isomorphism, provided it exists. Furthermore, if a category has all strict sums, we can turn $+$ into a functor by defining

$$f + g = [f ; \iota_1,\ g ; \iota_2],$$

for some specified choice of strict sums. The fact that this is functorial follows immediately from the universal property.

For games A and B, the injection maps $\iota_1 : A \to A + B$ and $\iota_2 : B \to A + B$ in $\mathscr{E}$ are defined by the obvious copycat strategies:

$$\iota_1 = \{\varepsilon, ql\} \cup \{qls \mid \forall t \sqsubseteq^{\text{even}} s.(t \upharpoonright A = t \upharpoonright A + B)\},$$

$$\iota_2 = \{\varepsilon, qr\} \cup \{qrs \mid \forall t \sqsubseteq^{\text{even}} s.(t \upharpoonright B = t \upharpoonright A + B)\}.$$

So, for example, ι_1 answers q with l and then plays copycat between A on the left and the copy of A in $A + B$ on the right. It is easy to see that this is an innocent strategy; in fact it is history-free.

Now suppose that we have $\sigma : A \to C$ and $\tau : B \to C$, for some well-opened game C. We can define a strategy $[\sigma, \tau] : A + B \to C$ as follows. In response to the first move in C, $[\sigma, \tau]$ plays q in $A + B$. If O now plays l, it continues playing as σ would, except that each time σ would play an initial move in A, $[\sigma, \tau]$ first plays q again

and waits for the response l before continuing. (If O plays r at this stage, the strategy $[\sigma, \tau]$ has no response.) The case when O plays r in answer to the first q is symmetrical. It is easy to verify that $[\sigma, \tau]$ is an innocent strategy and that it makes the diagrams in the definition of strict sum above commute.

Note that the maps ι_1 and ι_2 have right inverses given by $[\mathsf{id}_A, \bot]$ and $[\bot, \mathsf{id}_B]$, respectively. Furthermore, these injections and projections make linear use of their source types, corresponding to strategies (i.e., maps in $\mathscr{G}$)

$$\mathsf{inl}: A \multimap A + B \qquad \mathsf{outl}: A + B \multimap A$$

$$\mathsf{inr}: B \multimap A + B \qquad \mathsf{outr}: A + B \multimap B.$$

We will make use of these strategies in Section 5.2.

LEMMA 2.34. *For any two games A and B, the diagram*

$$A \xrightarrow{\iota_1} A + B \xleftarrow{\iota_2} B$$

in $\mathscr{E}$ is a strict sum.

Proof. By the construction above we know that for any maps $f: A \to C$ and $g: B \to C$ in $\mathscr{E}$ there exists an $h: A + B \to C$ making the required diagrams commute, namely $[f, g]$, so it just remains to show that h is unique.

Any map $\alpha: I \to A + B$ in $\mathscr{E}$ is either $\bot$ or factors as

$$I \xrightarrow{\alpha'} A \xrightarrow{\iota_1} A + B$$

or as

$$I \xrightarrow{\alpha'} B \xrightarrow{\iota_2} A + B,$$

so that by Lemma 2.33, if $\bot ; h = \bot ; h'$, $\iota_1 ; h = \iota_1 ; h'$, and $\iota_2 ; h = \iota_2 ; h'$ then $h = h'$. Therefore there can be at most one h satisfying the diagrams in the definition of strict sum. ∎

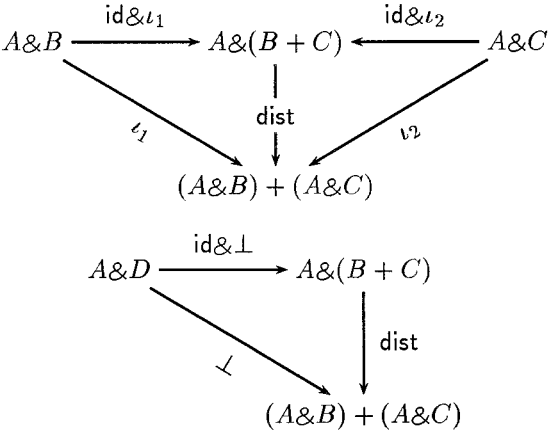
It is worth remarking that the situation in $\mathscr{I}$ is much less satisfactory. The strategy $[\sigma, \tau]$ described above is not unique in making the relevant diagrams commute, and indeed it does not seem possible to turn $+$ into a functor on $\mathscr{I}$. For this reason we work in $\mathscr{E}$ from now on.

2.7.1. Distribution of Product over Sum

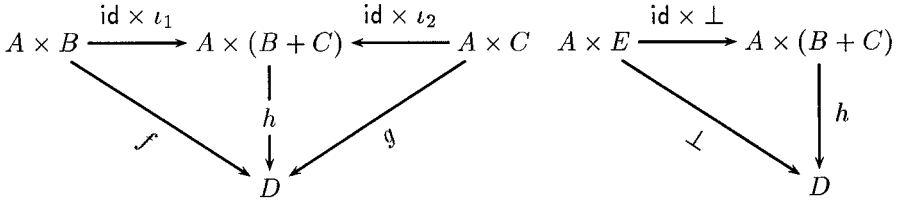
In order to model conditionals, it will be necessary for the product functor in $\mathscr{E}$ to distribute over $+$ in some reasonable sense. What is needed is a map (better, a natural transformation)

$$\mathsf{dist}: A \,\&\, (B + C) \to (A \,\&\, B) + (A \,\&\, C)$$

such that the following diagrams commute.



In fact, in a Cartesian closed category with strict sums (such as $\mathscr{E}$), the existence of such a distribution map is an automatic consequence of the following *parametrized* strict sum property: for any $f : A \times B \to D$ and $g : A \times C \to D$ there exists a unique $h : A \times (B + C) \to D$ such that the following diagrams commute.



Under these circumstances, the distribution map is uniquely defined, and the universal property above can be used to show that distribution is a natural transformation. The same technique can also be used to show that the strict sum functor is closed.

2.8. Games for recursive types

As demonstrated in [AM95b], games admit a treatment of recursive types very similar to that of information systems [Win93]. In fact there is an elegant general theory, essentially a modified version of the I-categories of Edalat and Smyth [ES93, Smy92, Eda93], which applies to all the categories of games we have considered. The full details are beyond the scope of this paper, but appear in [McC98]. Here we shall content ourselves with an outline of the main ideas.

2.8.1. Existence of Solutions

Suppose given a functor $F : \mathscr{E} \to \mathscr{E}$. We would like to solve the "domain equation" $D = F(D)$. For games A and B, define $A \trianglelefteq B$ if and only if

$$
\begin{aligned}
M_A &\subseteq M_B \\
\lambda_A &= \lambda_B \upharpoonright M_A \\
\vdash_A &= \vdash_B \cap ((M_A + \{\star\}) \times M_A) \\
P_A &= P_B \cap M_A^*.
\end{aligned}
$$

LEMMA 2.35. *The ordering $\trianglelefteq$ is a (large) dcpo, with least element I and directed suprema given by taking componentwise union.*

If the action on objects of the functor F is monotone and continuous with respect to $\trianglelefteq$, we can find a game D solving the required equation by setting $D = \bigsqcup_{n \geqslant 0} F^n(I)$.

2.8.2. Example: the Lazy Natural Numbers

The datatype of "lazy natural numbers" is given by the solution of the equation

$$D = I + D.$$

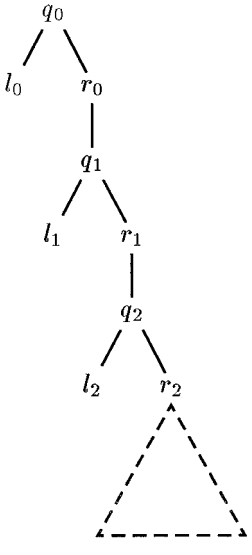
In our framework, a solution to this equation is obtained by forming a chain of games

$$I, I + I, I + (I + I), I + (I + (I + I)), \ldots$$

and taking the least upper bound with respect to $\trianglelefteq$. Each game in this chain has the following form:

- Opponent begins by asking an initial question q_0;
- Player may respond with either l_0 or r_0. If P plays l_0, the game is over. If player plays r_0, O may ask another question q_1;
- Play can continue in this way, with O asking questions q_n and P responding with l_n or r_n until either an l answer is given or the "bottom of the game tree" is reached.

Each successive iterate of the chain allows O to ask one further q_n before reaching the bottom of the game tree; the least upper bound is a tree with no bottom, so that the game can continue forever if P always plays the r answers.



2.8.3. Canonicity of Solutions

Having demonstrated that solutions of certain equations exist, it is reasonable to ask whether they are in some sense *canonical*. In a series of papers [Fre91, Fre92, Fre90], Freyd has recently proposed and investigated a suitable notion of canonicity called the *minimal invariant* condition which captures abstractly the key properties guaranteed by the well-known limit/colimit-coincidence theorem for recursively defined objects the category of Scott domains [SP82]. Loosely speaking, an object such that $D = F(D)$ is a minimal invariant for F if the least solution to the equation

$$f = Ff : D \to D$$

is the identity. For a reasonable class of functors, including all those that will be encountered in this paper, our solutions do satisfy this condition, as we now show.

For any games A and B with $A \trianglelefteq B$, there are canonical inclusion and projection maps $\mathsf{inc}_{A,B}$ and $\mathsf{proj}_{B,A}$ given by copycat strategies: as a set, each of these is equal to the identity strategy on A. Suppose the functor F is closed, so that it behaves well with respect to the order on maps, and preserves the inclusions and projections. The least solution of the equation above can be calculated as $\bigsqcup_{n \geqslant 0} F^n(\bot)$. We show by induction that $F^n(\bot) = \mathsf{proj}_{D,F^n(I)}; \mathsf{inc}_{F^n(I),D}$. This is clearly true for $n = 0$, and for the inductive step we calculate

$$\begin{aligned} F^{n+1}(\bot) &= F(\mathsf{proj}_{D,F^n(I)}); F(\mathsf{inc}_{F^n(I),D}) \\ &= \mathsf{proj}_{D,F^{n+1}(I)}; \mathsf{inc}_{F^{n+1}(I),D} \end{aligned}$$

since F preserves the inclusions and projections. Therefore

$$\bigsqcup_{n \geqslant 0} F^n(\bot) = \bigsqcup_{n \geqslant 0} \mathsf{proj}_{D,F^n(I)}; \mathsf{inc}_{F^n(I),D} = \mathsf{id}_D.$$

In summary:

LEMMA 2.36. *Let $F : \mathscr{E} \to \mathscr{E}$ be a closed functor which preserves inclusions and projections and is continuous with respect to $\trianglelefteq$. Then F has a minimal invariant given by $D = \bigsqcup_{n \geqslant 0} F^n(I)$.*

This result can be extended to handle mixed variance functors $F : \mathscr{E}^{\mathrm{op}} \times \mathscr{E} \to \mathscr{E}$ such as $[- \Rightarrow -]$ and to provide canonical solutions of parametrized equations such as

$$G(A) = F(A, G(A)).$$

In the latter case, the solution is a functor G which also satisfies the hypotheses of the lemma. We can easily show that each of the functors, $\times$, $\Rightarrow$, and $+$ satisfies these conditions, so we have canonical solutions for all domain equations built out of these type constructors.

The canonicity properties of such recursively defined games are extremely useful. In particular, Pitts has developed an elegant theory of *invariant relations*, relying on the minimal invariant condition. A special instance of this theory is the method of formal approximation relations [Plo85] which can be used to demonstrate computational adequacy of models of recursively types programming languages. We will make use of this later.

3. THE LANGUAGE FPC

Here we give the definition of the metalanguage **FPC**. This language, and similar ones, has appeared in [Plo85, Gun92, Win93]. A detailed treatment can be found in [Fio96]. It is a type theory with products, exponentials, sums, and recursive types; we consider it as a typed functional programming language in its own right, as has been done by Plotkin, Fiore, Winskel, and Gordon [Gor95a, Gor95b].

3.1. Syntax

There are two syntactic classes of variables: TypeVar for type variables and Var for expression variables. The syntax of **FPC** is defined as follows.

$$\begin{aligned}
\mathtt{T} &\in \text{TypeVar}.\\
\tau \in \text{Types} ::= {} & \mathtt{T} \mid \tau_1 + \tau_2 \mid \tau_1 \times \tau_2 \mid \tau_1 \to \tau_2 \mid \mu\mathtt{T}.\tau.\\
\mathtt{x} &\in \text{Var}.\\
M \in \text{Exp} ::= {} & \mathtt{x}\\
& \mid \mathtt{inl}_{\tau_1,\tau_2}(M) \mid \mathtt{inr}_{\tau_1,\tau_2}(M)\\
& \mid \mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2\\
& \mid (M_1, M_2)\\
& \mid \mathtt{fst}(M) \mid \mathtt{snd}(M)\\
& \mid \lambda \mathtt{x} : \tau.M\\
& \mid M_1(M_2)\\
& \mid \mathtt{intro}_{\mu\mathtt{T}.\tau}(M)\\
& \mid \mathtt{elim}(M).
\end{aligned}$$

The type tags on $\mathtt{inl}_{\tau_1,\tau_2}(M)$, $\mathtt{inr}_{\tau_1,\tau_2}(M)$, and $\mathtt{intro}_{\mu\mathtt{T}.\tau}(M)$ are necessary to ensure that a given term-in-context can have only one type. However, we will omit them whenever we think we can get away with it.

Since we are going to give a call-by-name semantics to **FPC**, in particular making the sum constructor lazy, sum will not be associative. Therefore it might be considered useful to have more type constructors corresponding to sums of all arities;

Type variables

$$\frac{}{\Theta, \mathtt{T}, \Theta' \vdash \mathtt{T}}$$

Recursive types

$$\frac{\Theta, \mathtt{T} \vdash \tau}{\Theta \vdash \mu \mathtt{T}.\tau}$$

Sums, products and exponentials

$$\frac{\Theta \vdash \tau_1 \quad \Theta \vdash \tau_2}{\Theta \vdash \tau_1 + \tau_2} \qquad \frac{\Theta \vdash \tau_1 \quad \Theta \vdash \tau_2}{\Theta \vdash \tau_1 \times \tau_2} \qquad \frac{\Theta \vdash \tau_1 \quad \Theta \vdash \tau_2}{\Theta \vdash \tau_1 \to \tau_2}$$

FIG. 2. Well-formed types of **FPC**.

in particular the unary sum will correspond to lifting. This does not add anything of technical difficulty, so for the sake of simplicity we stick to binary sums, but bear in mind the fact that we really intend to have all finite sums.

A well formed type consists of a list of distinct type variables Θ and a type τ, all of whose free variables appear in Θ. We will write $\Theta \vdash \tau$ to indicate that τ is a well formed type in context Θ. The well formed types are defined inductively in Fig. 2. The variable $\mathtt{T}$ is bound in $\mu \mathtt{T}.\tau$ and we denote substitution of a type τ' for the free occurrences of $\mathtt{T}$ in τ by $\tau[\mathtt{T} \mapsto \tau']$. As usual, we identify types up to α-equivalence. For the most part we are going to be concerned with closed types, i.e., those types τ such that $\vdash \tau$ is derivable.

An expression context Θ, Γ consists of a list of distinct type variables, Θ, and a list of (variable, type) pairs, Γ. The variables occurring in Γ must all be distinct, and if τ is a type occurring in Γ then $\Theta \vdash \tau$ must be a well formed type. Each entry in Γ is written as $\mathtt{x}{:}\tau$. Well-formed expressions are given by judgements $\Theta, \Gamma \vdash M : \tau$ where Θ, Γ is an expression context; the inductive definition is given in Fig. 3. The expression $\mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2$ binds $\mathtt{x}_1$ in M_1 and $\mathtt{x}_2$ in M_2, while $\lambda \mathtt{x}{:}\tau.M$ binds $\mathtt{x}$ in M. Expressions are identified up to α-equivalence, and we denote the substitution of N for free occurrences of $\mathtt{x}$ in M by $M[N/\mathtt{x}]$. Note that the type context Θ plays vary little part in this definition. In fact, we will mainly work with terms of closed type, so that Θ is empty. However, as it stands the language supports *parametricity*; this is not important for us, but Fiore treats it in his thesis [Fio96].

DEFINITION. An **FPC** *program* is a closed term of closed type, i.e., an expression M such that $\vdash M{:}\tau$ is derivable. We write Prog for the set of programs, tagged with their types. The notion of *context* $C[-]$ with hole of a given type can also be defined; informally, a context is just a term with (possibly several occurrences of) a "hole" in it, and $C[M]$ denotes the result of filling in each hole with the expression M. Unlike expressions, contexts are *not* identified modulo α-equivalence. We

Variables

$$\frac{}{\Theta, \Gamma_1, \mathtt{x} : \tau, \Gamma_2 \vdash \mathtt{x} : \tau}$$

Sums

$$\frac{\Theta, \Gamma \vdash M : \tau \quad \Theta \vdash \tau'}{\Theta, \Gamma \vdash \mathtt{inl}_{\tau,\tau'}(M) : \tau + \tau'} \qquad \frac{\Theta, \Gamma \vdash M : \tau \quad \Theta \vdash \tau'}{\Theta, \Gamma \vdash \mathtt{inr}_{\tau',\tau}(M) : \tau' + \tau}$$

$$\frac{\Theta, \Gamma \vdash M : \tau_1 + \tau_2 \quad \Theta, \Gamma, \mathtt{x}_1 : \tau_1 \vdash M_1 : \tau \quad \Theta, \Gamma, \mathtt{x}_2 : \tau_2 \vdash M_2 : \tau}{\Theta, \Gamma \vdash \mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2 : \tau}$$

Products

$$\frac{\Theta, \Gamma \vdash M_1 : \tau_1 \quad \Theta, \Gamma \vdash M_2 : \tau_2}{\Theta, \Gamma \vdash (M_1, M_2) : \tau_1 \times \tau_2}$$

$$\frac{\Theta, \Gamma \vdash M : \tau_1 \times \tau_2}{\Theta, \Gamma \vdash \mathtt{fst}(M) : \tau_1} \qquad \frac{\Theta, \Gamma \vdash M : \tau_1 \times \tau_2}{\Theta, \Gamma \vdash \mathtt{snd}(M) : \tau_2}$$

Exponentials

$$\frac{\Theta, \Gamma, \mathtt{x} : \tau_1 \vdash M : \tau_2}{\Theta, \Gamma \vdash \lambda \mathtt{x} : \tau_1.M : \tau_1 \to \tau_2}$$

$$\frac{\Theta, \Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Theta, \Gamma \vdash M_2 : \tau_1}{\Theta, \Gamma \vdash M_1(M_2) : \tau_2}$$

Recursive types

$$\frac{\Theta, \Gamma \vdash M : \tau[\mathtt{T} \mapsto \mu \mathtt{T}.\tau]}{\Theta, \Gamma \vdash \mathtt{intro}_{\mu \mathtt{T}.\tau}(M) : \mu \mathtt{T}.\tau}$$

$$\frac{\Theta, \Gamma \vdash M : \mu \mathtt{T}.\tau}{\Theta, \Gamma \vdash \mathtt{elim}(M) : \tau[\mathtt{T} \mapsto \mu \mathtt{T}.\tau]}$$

FIG. 3. Well-formed expressions of **FPC**.

will be interested in closed contexts of type τ, that is those contexts $C[-]$ such that if M is a closed expression of the same type as the hole, then $C[M]$ is a closed expression of type τ. If $C[-]$ is such a context, we write $C[-] : \tau$.

3.2. Operational Semantics

We equip the language **FPC** with an operational semantics. In contrast to the work of Fiore and Plotkin, our semantics is call-by-name. As usual it is given in terms of a "big-step" evaluation relation $\Downarrow$. For readability, we assume all the terms

Sums

$$\frac{}{\mathtt{inl}_{\tau,\tau'}(M) \Downarrow \mathtt{inl}_{\tau,\tau'}(M)} \qquad \frac{}{\mathtt{inr}_{\tau,\tau'}(M) \Downarrow \mathtt{inr}_{\tau,\tau'}(M)}$$

$$\frac{M \Downarrow \mathtt{inl}(M') \quad M_1[M'/\mathtt{x}_1] \Downarrow M''}{\mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2 \Downarrow M''}$$

$$\frac{M \Downarrow \mathtt{inr}(M') \quad M_2[M'/\mathtt{x}_2] \Downarrow M''}{\mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2 \Downarrow M''}$$

Products

$$\frac{}{(M_1, M_2) \Downarrow (M_1, M_2)}$$

$$\frac{M \Downarrow (M_1, M_2) \quad M_1 \Downarrow M'}{\mathtt{fst}(M) \Downarrow M'} \qquad \frac{M \Downarrow (M_1, M_2) \quad M_2 \Downarrow M'}{\mathtt{snd}(M) \Downarrow M'}$$

Exponentials

$$\frac{}{\lambda\mathtt{x} : \tau.M \Downarrow \lambda\mathtt{x} : \tau.M}$$

$$\frac{M_1 \Downarrow \lambda\mathtt{x} : \tau.M \quad M[M_2/\mathtt{x}] \Downarrow M'}{M_1(M_2) \Downarrow M'}$$

Recursive types

$$\frac{}{\mathtt{intro}_{\mu\mathtt{T}.\tau}(M) \Downarrow \mathtt{intro}_{\mu\mathtt{T}.\tau}(M)}$$

$$\frac{M \Downarrow \mathtt{intro}(M') \quad M' \Downarrow M''}{\mathtt{elim}(M) \Downarrow M''}$$

FIG. 4. Operational semantics of **FPC**.

which appear in the definition above are well formed in some expression context and omit this context. The definition of the evaluation relation is given in Fig. 4.

3.3. Observational Preorder

We are now in a position to define the *observational preorder* $\sqsubseteq_{\approx}$ on **FPC** programs. Given two programs M and N of the same type, define

$$M \sqsubseteq_{\approx} N \Leftrightarrow \forall C[\,-\,]:\tau_1 + \tau_2\,[\,C[M]\Downarrow \Rightarrow C[N]\Downarrow\,],$$

where τ_1 and τ_2 are any closed types and $M\Downarrow$ means that there exists some N such that $M \Downarrow N$.

4. A MODEL OF FPC

We have seen that $\mathscr{E}$ is Cartesian closed and has strict sums and that type constructors built out of these functors have minimal invariants. Therefore $\mathscr{E}$ has all the structure required to interpret **FPC**. It is well known how to interpret an **FPC** type with n free type variables as a functor $(\mathscr{E}^{\mathrm{op}} \times \mathscr{E})^n \to \mathscr{E}$. For example, a type variable is interpreted as the appropriate projection; if τ_1 and τ_2 are types with a single free type variable $\mathtt{T}$, then

$$[\![\tau_1 + \tau_2]\!](A, B) = [\![\tau_1]\!](A, B) + [\![\tau_2]\!](A, B)$$
$$[\![\tau_1 \to \tau_2]\!](A, B) = [\![\tau_1]\!](B, A) \Rightarrow [\![\tau_2]\!](A, B)$$

(note how the order of variables is switched in the contravariant position); and the type $\mu\mathtt{T}.\tau_1$ is interpreted as the minimal invariant of the functor $[\![\tau_1]\!]$. For further details, see any of [Plo85, Fio96, McC98].

Closed types, then, are interpreted as objects of $\mathscr{E}$, and the following equations hold.

$$[\![\tau_1 + \tau_2]\!] = [\![\tau_1]\!] + [\![\tau_2]\!]$$
$$[\![\tau_1 \times \tau_2]\!] = [\![\tau_1]\!] \times [\![\tau_2]\!]$$
$$[\![\tau_1 \to \tau_2]\!] = [\![\tau_1]\!] \Rightarrow [\![\tau_2]\!]$$
$$[\![\mu\mathtt{T}.\tau_1]\!] = [\![\tau_1[\mathtt{T} \mapsto \mu\mathtt{T}.\tau_1]]\!].$$

The interpretation of terms is now standard—see Figs. 5 and 6. A term $\Gamma \vdash M : \tau$ is interpreted as a morphism $[\![\Gamma \vdash M]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$, where the denotation of a context Γ is the product of the denotations of its elements. Note that since we solve domain equations up to identity in $\mathscr{E}$, the denotation of $\mathtt{intro}(M)$ is the same as that of M. Note also how the distribution of product over sum is used in the interpretation of the `case` construct.

The following two results can be proved by straightforward inductive arguments.

LEMMA 4.1 (Substitution). *Suppose* $\Gamma, \mathtt{x} : \tau_1 \vdash M : \tau_2$ *and* $\Gamma \vdash N : \tau_1$. *Then* $\Gamma \vdash M[N/\mathtt{x}] : \tau_2$ *and*

$$[\![\Gamma \vdash M[N/\mathtt{x}] : \tau_2]\!] = \langle \mathsf{id}_{[\![\Gamma]\!]}, [\![\Gamma \vdash N : \tau_1]\!] \rangle ; [\![\Gamma, \mathtt{x} : \tau_1 \vdash M : \tau_2]\!].$$

PROPOSITION 4.2 (Soundness). *Let* $\Gamma \vdash M$ *and* $\Gamma \vdash N$, *and suppose* $M \Downarrow N$. *Then* $[\![M]\!] = [\![N]\!]$.

In order to extend this to an inequational soundness theorem, we require the following.

PROPOSITION 4.3 (Computation adequacy). *Let* M *be an* **FPC** *program of type* τ. *If* $[\![M]\!] \neq \bot$, *then* M *converges.*

Variables

$$[\![\Gamma_1, \mathtt{x} : \tau, \Gamma_2 \vdash \mathtt{x} : \tau]\!] = \pi_{n+1} : [\![\Gamma_1]\!] \times [\![\tau]\!] \times [\![\Gamma_2]\!] \to [\![\tau]\!], \quad \text{where } n = |\Gamma_1|.$$

Sums

$$\frac{[\![\Gamma \vdash M : \tau]\!] = m : [\![\Gamma]\!] \to [\![\tau]\!]}{[\![\Gamma \vdash \mathtt{inl}_{\tau,\tau'}(M)]\!] = m\,; \iota_1 : [\![\Gamma]\!] \to [\![\tau]\!] + [\![\tau']\!]}$$

$$\frac{[\![\Gamma \vdash M : \tau]\!] = m : [\![\Gamma]\!] \to [\![\tau]\!]}{[\![\Gamma \vdash \mathtt{inr}_{\tau',\tau}(M)]\!] = m\,; \iota_2 : [\![\Gamma]\!] \to [\![\tau']\!] + [\![\tau]\!]}$$

$$\frac{\begin{array}{rcl}[\![\Gamma \vdash M : \tau_1 + \tau_2]\!] & = & m : [\![\Gamma]\!] \to [\![\tau_1]\!] + [\![\tau_2]\!] \\ [\![\Gamma, \mathtt{x}_1 : \tau_1 \vdash M_1 : \tau]\!] & = & m_1 : [\![\Gamma]\!] \times [\![\tau_1]\!] \to [\![\tau]\!] \\ [\![\Gamma, \mathtt{x}_2 : \tau_2 \vdash M_2 : \tau]\!] & = & m_2 : [\![\Gamma]\!] \times [\![\tau_2]\!] \to [\![\tau]\!]\end{array}}{\begin{array}{l}[\![\Gamma \vdash \mathtt{case}\ M\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).M_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).M_2 : \tau]\!] \\ = \quad \langle \mathsf{id}_{[\![\Gamma]\!]}, m\rangle\,; \mathsf{dist}\,; [m_1, m_2] : [\![\Gamma]\!] \to [\![\tau]\!]\end{array}}$$

Products

$$\frac{[\![\Gamma \vdash M_1 : \tau_1]\!] = m_1 : [\![\Gamma]\!] \to [\![\tau_1]\!] \quad [\![\Gamma \vdash M_2 : \tau_2]\!] = m_2 : [\![\Gamma]\!] \to [\![\tau_2]\!]}{[\![\Gamma \vdash (M_1, M_2) : \tau_1 \times \tau_2]\!] = \langle m_1, m_2\rangle : [\![\Gamma]\!] \to [\![\tau_1]\!] \times [\![\tau_2]\!]}$$

$$\frac{[\![\Gamma \vdash M : \tau_1 \times \tau_2]\!] = m : [\![\Gamma]\!] \to [\![\tau_1]\!] \times [\![\tau_2]\!]}{[\![\Gamma \vdash \mathtt{fst}(M) : \tau_1]\!] = m\,; \pi_1 : [\![\Gamma]\!] \to [\![\tau_1]\!]}$$

$$\frac{[\![\Gamma \vdash M : \tau_1 \times \tau_2]\!] = m : [\![\Gamma]\!] \to [\![\tau_1]\!] \times [\![\tau_2]\!]}{[\![\Gamma \vdash \mathtt{snd}(M) : \tau_2]\!] = m\,; \pi_2 : [\![\Gamma]\!] \to [\![\tau_2]\!]}$$

FIG. 5. Denotational semantics of **FPC**.

Exponentials

$$\frac{[\![\Gamma, \mathtt{x} : \tau_1 \vdash m : \tau_2]\!] = m : [\![\Gamma]\!] \times [\![\tau_1]\!] \to [\![\tau_2]\!]}{[\![\Gamma \vdash \lambda \mathtt{x} : \tau_1.M : \tau_1 \to \tau_2]\!] = \Lambda(M) : [\![\Gamma]\!] \to [[\![\tau_1]\!] \Rightarrow [\![\tau_2]\!]]}$$

$$\frac{\begin{array}{rcl}[\![\Gamma \vdash M_1 : \tau_1 \to \tau_2]\!] & = & m_1 : [\![\Gamma]\!] \to [[\![\tau_1]\!] \Rightarrow [\![\tau_2]\!]] \\ [\![\Gamma \vdash M_2 : \tau_1]\!] & = & m_2 : [\![\Gamma]\!] \to [\![\tau_1]\!]\end{array}}{[\![\Gamma \vdash M_1(M_2) : \tau_2]\!] = \langle m_1, m_2\rangle\,; \mathsf{ev} : [\![\Gamma]\!] \to [\![\tau_2]\!]}$$

Recursive types

$$\frac{[\![\Gamma \vdash M : \tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!] = m : [\![\Gamma]\!] \to [\![\tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!]}{[\![\Gamma \vdash \mathtt{intro}(M) : \mu\mathtt{T}.\tau]\!] = m : [\![\Gamma]\!] \to [\![\mu\mathtt{T}.\tau]\!]}$$

$$\frac{[\![\Gamma \vdash M : \mu\mathtt{T}.\tau]\!] = m : [\![\Gamma]\!] \to [\![\mu\mathtt{T}.\tau]\!]}{[\![\Gamma \vdash \mathtt{elim}(M) : \tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!] = m : [\![\Gamma]\!] \to [\![\tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!]}$$

FIG. 6. Denotational semantics of **FPC**, continued.

This is proved using an adaptation of Plotkin's method of *formal approximation relations* [Plo85], assisted by Pitts' more general theory of invariant relations [Pit96]. The proof is by no means trivial, but is standard, so we omit the details. It should be remarked that this is an instance of a quite general computational adequacy result for models of **FPC** in rational categories [McC98]. It is now straightforward to establish the first half of our full abstraction theorem.

THEOREM 4.4 (Inequational soundness). *Let M and N be two* **FPC** *programs of type τ. Then*

$$[\![M]\!] \leqslant [\![N]\!] \Rightarrow M \sqsubseteq_{\sim\tau} N.$$

5. FULL ABSTRACTION

The remainder of the paper is devoted to showing that the model of **FPC** in $\mathscr{E}$ is complete and hence fully abstract. The proof hinges on a *definability* result: we show that all of the compact elements of our model are the denotation of some **FPC** term. This involves a detailed analysis of the possible behaviours of a strategy which bears some resemblance to the definability results for games models of **PCF** [AJM97, HO97] but differs in some important respects. In particular, our definability result holds only in the extensional category $\mathscr{E}$ and its proof involves manipulation of strategies which preserves extensional equivalence (equality of maps in $\mathscr{E}$) but not equality of strategies.

The structure of the proof is as follows. First we reduce the question of completeness to that of definability of compact strategies on finite types, that is, types built with no nontrivial use of recursion. Next a detailed analysis of the behaviour of such strategies is used to obtain a decomposition theorem, characterizing the behaviour of a strategy (up to extensional equivalence) in terms of smaller substrategies. Finally, it is shown that this decomposition of strategies corresponds precisely to the decomposition of **FPC** terms into subterms, facilitating an inductive proof of the required definability result.

5.1. Finite Types Suffice

The first step on the way to the full abstraction theorem is to show that it suffices to prove definability for compact strategies at finite types. The finite types are given by the grammar

$$\tau ::= \mathtt{null} \mid \tau \to \tau \mid \tau + \tau \mid \tau \times \tau.$$

We use the type `null` as an abbreviation for $\mu \mathtt{T}.\mathtt{T}$. Its denotation in the model is the terminal object. Note that all finite types are automatically closed. We will refer to a game A such that $A = [\![\tau]\!]$ for some finite type as a finite type; similarly, all games which are the denotation of an **FPC**-type are themselves referred to as **FPC**-types.

We first characterise all **FPC**-types as limits of certain kinds of chains.

DEFINITION. A chain of games $A_1 \trianglelefteq A_2 \trianglelefteq \cdots$ is a *definable chain* if each A_i is a finite type, and if $A = \bigsqcup A_i$ is an **FPC**-type then each $\mathsf{inc}_{A_i, A}$ and proj_{A, A_i} is definable.

The proof of the following result is largely routine and is omitted.

PROPOSITION 5.1. *Let $\Theta \vdash \tau$ be an* **FPC***-type, and let $\tau_1, ..., \tau_n$ be finite types, where $n = |\Theta|$. Then $A = [\![\tau[\Theta \mapsto \vec{\tau}_i]]\!]$ is the least upper bound of a definable chain of finite types.*

We can now show that a definability result for finite types will suffice to prove full abstraction. We will frequently confuse strategies σ with equivalence classes $[\sigma]$; that is to say we blur the distinction between maps in $\mathscr{I}$ and maps in $\mathscr{E}$. In particular, saying that a strategy σ is definable means that there exists an **FPC** term M such that $[\![M]\!] = [\sigma]$. Since every operation we use is compatible with the extensional equivalence relation on strategies induced by $\preccurlyeq$, this will cause no difficulty.

PROPOSITION 5.2 (Finite types suffice). *If all compact strategies at finite types are definable, the games model of* **FPC** *is fully abstract.*

Proof. Let $\vdash e_1 : \tau$ and $\vdash e_2 : \tau$ be expressions such that $[\![e_1]\!] \not\preccurlyeq [\![e_2]\!]$. We shall show that $e_1 \not\sqsubseteq e_2$.

By definition of $\preccurlyeq$ there exists a map $\alpha : \tau \to 1 + 1$ such that $[\![e_1]\!]; \alpha \neq \bot$ and $[\![e_2]\!]; \alpha = \bot$. Note that this holds independent of the choice of representative strategies for the equivalence classes $[\![e_1]\!]$ and $[\![e_2]\!]$, so we can assume that α is compact.

By Lemma 5.1, $[\![\tau]\!]$ is the least upper bound of a definable chain of finite types. Let the inclusions and projections between the chain and $[\![\tau]\!]$ be i_n and p_n. We have $\bigsqcup p_n; i_n = \mathsf{id}$ so that

$$[\![e_1]\!]; \bigsqcup (p_n; i_n); \alpha \neq \bot$$

and

$$[\![e_2]\!]; \bigsqcup (p_n; i_n); \alpha = \bot,$$

so for some n,

$$[\![e_1]\!]; p_n; i_n; \alpha \neq \bot$$

and

$$[\![e_2]\!]; p_n; i_n; \alpha = \bot.$$

Then $i_n; \alpha$ is a compact strategy at a finite type and therefore is definable, say by $\mathtt{y} : \tau' \vdash m$, and since the chain approximating $[\![\tau]\!]$ is definable, p_n is definable,

say by $\mathtt{x}:\tau \vdash n$. Then for any expression e_3 of type τ, $[\![e_3]\!]; p_n; i_n; \alpha = [\![(\lambda \mathtt{y}:\tau'.m)((\lambda \mathtt{x}:\tau.n)(e_3))]\!]$, so we have

$$[\![(\lambda \mathtt{y}:\tau'.m)((\lambda \mathtt{x}:\tau.n)(e_1))]\!] \neq \bot$$

$$[\![(\lambda \mathtt{y}:\tau'.m)((\lambda \mathtt{x}:\tau.n)(e_2))]\!] = \bot$$

and by computational adequacy, $(\lambda \mathtt{y}:\tau'.m)((\lambda \mathtt{x}:\tau.n)(e_1))$ converges, but $(\lambda \mathtt{y}:\tau'.m)((\lambda \mathtt{x}:\tau.n)(e_2))$ does not. So $e_1 \not\sqsubseteq e_2$.

We can conclude that $e_1 \sqsubseteq e_2 \Rightarrow [\![e_1]\!] \leqslant [\![e_2]\!]$, so the model is complete; since it is also sound, it is fully abstract. ▌

We have reduced the problem of proving completeness to that of showing that all compact strategies at finite types are definable. In fact we can go slightly further. Let the *basic types* be those generated by the grammar

$$\sigma ::= \mathtt{null} \mid \sigma \times \sigma \mid \tau \to \tau + \tau$$

where τ ranges over finite types.

LEMMA 5.3. *For every finite type τ there exists a basic type σ such that $[\![\tau]\!] \cong [\![\sigma]\!]$, and the isomorphism is definable.*

Because of this lemma, we need only consider the question of definability for compact strategies of basic type.

5.2. Decomposition

In this section, a decomposition of strategies is described which will facilitate an inductive proof of the definability of all finite strategies at finite types and hence show that the model of **FPC** in $\mathscr{E}$ is fully abstract. Intuitively, the decomposition of strategies corresponds closely to the decomposition of **FPC** terms into their subterms.

We now begin a detailed analysis of what a strategy can do. Throughout this section we will be working with strategies and composition at the linear level; i.e., we are working in $\mathscr{G}$. The crucial case is that of a strategy $\sigma : !T \otimes (!A \multimap B + C) \to D + E$ which responds to the initial q in $D + E$ by q' in $B + C$. We shall extract three substrategies from σ and characterize the tests which σ passes in terms of the behaviour of these substrategies only.

The substrategies required are as follows.

$$\mathsf{arg}(\sigma) : !T \multimap !A$$

$$\mathsf{cont}_{\mathsf{L}}(\sigma) : !T \otimes (!A \multimap B) \multimap D + E$$

$$\mathsf{cont}_{\mathsf{R}}(\sigma) : !T \otimes (!A \multimap C) \multimap D + E$$

The strategies $\mathsf{cont}_{\mathsf{L}}(\sigma)$ and $\mathsf{cont}_{\mathsf{R}}(\sigma)$ are easy to extract: they are given by

$$\mathsf{cont}_{\mathsf{L}}(\sigma) = \mathsf{id}_{!T} \otimes (\mathsf{id}_{!A} \multimap \mathsf{inl}); \sigma$$

$$\mathsf{cont}_{\mathsf{R}}(\sigma) = \mathsf{id}_{!T} \otimes (\mathsf{id}_{!A} \multimap \mathsf{inr}); \sigma.$$

(Recall from Section 2.7 that inl and inr are the strategies corresponding to the injections ι_1 and ι_2.) Define

$$\mathsf{arg}(\sigma) = \{s \mid qq's \in \sigma, q' \text{ not answered in } s\}.$$

LEMMA 5.4. *After a suitable relabelling of moves and adjustment of justification pointers,* $\mathsf{arg}(\sigma)$ *is a well defined innocent strategy for* $!T \multimap !A$.

Proof. First note that if $qq's \in \sigma$ and q' is not answered in s then all moves of s are in the $!T$ and $!A$ components, because of the bracketing condition; note further that the O/P labelling of these moves is the same as it is in $!T \multimap !A$. As for justification, we need to alter the justifier of any initial move m of $!T$ in s to point to the unique initial $!A$-move in the P-view of s at the point when m is played. It is then easy to check that s is a valid position of $!T \multimap !A$ and furthermore that if $|s|$ is odd then

$$\ulcorner qq's \urcorner = qq' \ulcorner s \urcorner$$

so that $\mathsf{arg}(\sigma)$ is an innocent strategy. ∎

The idea behind these substrategies is that they capture the behaviour of σ in a manner close to the syntax of **FPC**. The strategy σ begins by interrogating its input in the type $(!A \multimap B + C)$. The play between the initial question in $B + C$ and its answers corresponds to σ applying this input to an argument, which is given by the strategy $\mathsf{arg}(\sigma)$. The behaviour of σ then branches according to the result of this application: if l is played the behaviour becomes that of $\mathsf{cont}_{\mathsf{L}}(\sigma)$, and if r is played, it becomes $\mathsf{cont}_{\mathsf{R}}(\sigma)$. The fact that these three substrategies completely determine σ (up to extensional equivalence) in this way is made precise in the success lemma, to follow; these observations form the core of an inductive proof of definability of finite strategies, because the substrategies extracted have strictly smaller view functions than does σ.

We now embark upon the proof of the success lemma. First a definition and some lemmas which smooth the way a little.

DEFINITION. Suppose σ is a strategy for a game A and s is a justified sequence of moves from A. Then σ is *prepared to play* s if for all P-moves m in s, $\mathsf{fun}(\sigma)(\ulcorner s_{<m} \urcorner) = m$.

Suppose σ is prepared to play s; then if $s \in P_A$, $s \in \sigma$. The reason for introducing this definition is to state the following two lemmas.

LEMMA 5.5. *Let s and $t \in \sigma$ and suppose that u is some justified sequence obtained from s and t by interleaving, permuting, and possibly omitting moves, but that every move of u is justified by the same move as it was in s or t, and that if m is a* P*-move in u then the move immediately preceding it is the same as in s or t. Then σ is prepared to play u.*

Proof. Let m be a P-move of u which came from s. We show that $\ulcorner u_{<m} \urcorner = \ulcorner s_{<m} \urcorner$ so that $\mathsf{fun}(\sigma)(\ulcorner u_{<m} \urcorner) = m$. Proceed by induction on $|u_{<m}|$. The base case is trivial. For the inductive step, if the predecessor of m is an initial move, then both the views in question consist just of this move, so we are done. Otherwise, suppose $u_{<m} = u' a u'' b$ with b justified by a. Then

$$\begin{aligned}
\ulcorner u_{<m} \urcorner &= \ulcorner u' \urcorner ab \\
&= \ulcorner u_{<a} \urcorner ab \\
&= \{\text{inductive hypothesis}\} \\
&\quad \ulcorner s_{<a} \urcorner ab \\
&= \{\text{justifier of } b \text{ in } s \text{ is } a\} \\
&\quad \ulcorner s_{\leqslant b} \urcorner \\
&= \{\text{predecessor of } m \text{ in } s \text{ is } b\} \\
&\quad \ulcorner s_{<m} \urcorner. \quad \blacksquare
\end{aligned}$$

LEMMA 5.6. *Suppose $\sigma : A \multimap B$ and $\tau : B \multimap C$ are innocent strategies and that s is a sequence of moves of A, B, and C along with justification pointers such that $s \upharpoonright A, C \in P_{A \multimap C}$ and has even length, σ is prepared to play $s \upharpoonright A \multimap B$, and τ is prepared to play $s \upharpoonright B \multimap C$. Then $s \in \sigma \parallel \tau$.*

Proof. An easy induction shows that each prefix of s is a valid interaction sequence and then either σ or τ supplies the next move, which must be valid by Lemma 2.15, or it is supplied by Opponent in one of the two "visible" components, A or C, and is therefore valid by hypothesis. $\blacksquare$

This lemma considerably reduces the burden of proof when claiming that some string is contained in $\sigma \parallel \tau$. We do not have to show that all the relevant restrictions are valid positions, which involves laborious checking of the visibility condition, bracketing condition, and so on; rather, we check that the visible part of the string is valid and that σ and τ are prepared to play their parts. Lemma 5.5 is useful in establishing this. We make use of these facts in the proof of the next lemma, which is the key on the way to our definability result.

LEMMA 5.7 (Success lemma). *Let $\sigma : !T \otimes (!A \multimap B + C) \to D + E$ be a strategy which responds to the initial q in $D + E$ by q' in $B + C$, and let $\alpha_1 : I \to !T$,*

$\alpha_2 : I \to !A \multimap B + C$, *and* $\beta : D + E \to \Sigma$. *Then* $\alpha_1 \otimes \alpha_2 ; \sigma ; \beta = \top$ *if and only if one of the following conditions is true.*

- β *is the constantly* $\top$ *strategy.*
- $\alpha_1 ; \mathsf{arg}(\sigma) ; \mathcal{A}^{-1}(\alpha_2) = m ;$ inl *for some strategy* m *and*

$$(\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) ; \mathsf{cont}_{\mathsf{L}}(\sigma) ; \beta = \top.$$

- $\alpha_1 ; \mathsf{arg}(\sigma) ; \mathcal{A}^{-1}(\alpha_2) = m ;$ inr *for some strategy* m *and*

$$(\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outr})) ; \mathsf{cont}_{\mathsf{R}}(\sigma) ; \beta = \top.$$

Recall that outl *and* outr *are the right inverses to* inl *and* inr *introduced in Section* 2.7.

Proof. If the first condition is true then it is clear that any test using β succeeds, so we assume that β is not the constantly $\top$ strategy. This proof will require a detailed analysis of play in composite strategies, and we will need to consider the restriction of strings to the subsequence relevant to a given strategy repeatedly; this makes the notation rather messy, so we prefer to omit the restriction symbols, since the component in question is always clear from the strategy concerned.

Suppose first that the test succeeds; i.e.,

$$\alpha_1 \otimes \alpha_2 ; \sigma ; \beta = \top.$$

Then the play in $\alpha_1 \otimes \alpha_2 \parallel \sigma \parallel \beta = \top$ must be as shown in Fig. 7.

It begins with $q_1 q_2 q_3 s$, where q_3 is not answered in s. But because the test will ultimately succeed, giving a well-balanced position, that is to say one in which all questions have been answered, q_3 must eventually receive an answer, which is either l or r. Suppose it is l, so the play has the form $q_1 q_2 q_3 s l \cdots$. After l, play continues

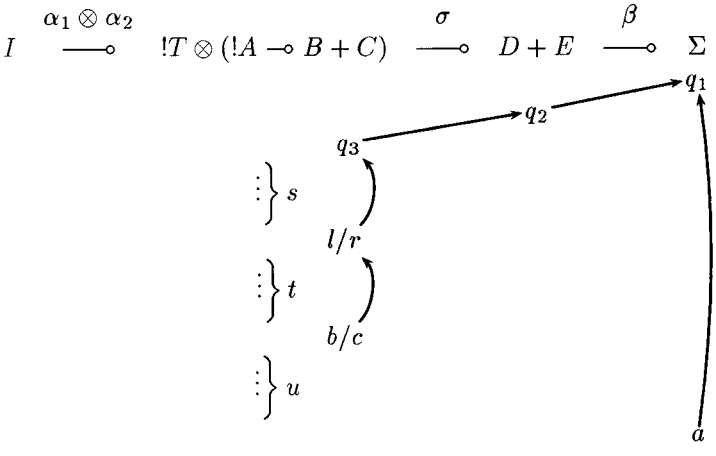


FIG. 7. Successful test of σ.

with a string t, and then the component $B + C$ may or may not be reentered with a move b, followed by some more play u and eventually the answer a. Therefore the play has the form $q_1 q_2 q_3 sltbua$. The key "checkpoints" in this play are the moves q_3, l, and b. Note that, because of the bracketing condition, $s \upharpoonright D + E$, $\Sigma = \varepsilon$.

We have $q_2 q_3 s \in \sigma$, and q_3 is not answered in s, so $s \in \mathsf{arg}(\sigma)$. We also have $q_3 sl \in \alpha_1 \otimes \alpha_2$, and it is then simple to check that $q_3 sl \in \alpha_1 \parallel \mathsf{arg}(\sigma) \parallel \Lambda^{-1}(\alpha_2)$, so that $\alpha_1 ; \mathsf{arg}(\sigma) ; \Lambda^{-1}(\alpha_2) = m ; \mathsf{inl}$ for some m.

We now claim that $q_1 q_2 tbsua \in (\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) \parallel \mathsf{cont}_{\mathsf{L}}(\sigma) \parallel \beta$, witnessing the fact that $(\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) ; \mathsf{cont}_{\mathsf{L}}(\sigma) ; \beta = \top$. Our argument will use Lemmas 5.5 and 5.6. First of all, observe that the part of this play relevant to β is the same play as in the original sequence $q_1 q_2 q_3 sltbua$, so we need only concern ourselves with whether or not $\mathsf{cont}_{\mathsf{L}}(\sigma)$ and $\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})$ are prepared to play their parts. By Lemma 5.6, this will establish the result.

We know that $q_2 q_3 sltbu \in \sigma$. Furthermore, it is easy to show that during t, neither player's view includes any move of s, so that no move of t is justified in s, by the visibility condition. So the permutation $q_2 q_3 ltbsu$ is a justified sequence with each move justified by the same move as in $q_2 q_3 sltbu$. The only moves whose predecessors have changed in his permutation are the first move of s, the first move of u, and l; but these are all O-moves, from σ's point of view, so we can conclude that σ is prepared to play $q_2 q_3 ltbsu$, and therefore $\mathsf{cont}_{\mathsf{L}}(\sigma)$ is prepared to play $q_2 tbsu$.

It remains to show that $\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})$ is prepared to play its part, namely $tbsu$. For this it suffices to show that $\alpha_1 \otimes \alpha_2$ is prepared to play $tq_3 slbu$. We know that $q_3 sltbu \in \alpha_1 \otimes \alpha_2$, and no move of t is justified by one of s, so $tq_3 slbu$ is a justified sequence, and the only moves whose predecessors have changed in this permutation are q_3, b, and the first move of t, which are all O-moves from the point of view of $\alpha_1 \otimes \alpha_2$, which is all that is needed, by Lemma 5.5.

For the converse, suppose that $\alpha_1 ; \mathsf{arg}(\sigma) ; \Lambda^{-1}(\alpha_2) = m ; \mathsf{inl}$ for some strategy m and

$$(\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) ; \mathsf{cont}_{\mathsf{L}}(\sigma) ; \beta = \top.$$

We shall show that $\alpha_1 \otimes \alpha_2 ; \sigma ; \beta = \top$. (The other case is similar.) Suppose the plays witnessing these two facts are

$$q_3 sl \in \alpha_1 \parallel \mathsf{arg}(\sigma) \parallel \Lambda^{-1}(\alpha_2)$$

and

$$q_1 q_2 tbua \in (\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) \parallel \mathsf{cont}_{\mathsf{L}}(\sigma) \parallel \beta = \top,$$

where no move of t is in $!A \multimap B$, so b is the initial move of B. See Fig. 8.

We claim that $u = sv$ for some sequence v and that the play $q_2 q_3 sltbv \in \sigma$ witnesses the success of the test.

We know that $q_2 q_3 s \in \sigma$ and $q_2 tb \in \mathsf{cont}_{\mathsf{L}}(\sigma)$, so $q_2 q_3 ltb \in \sigma$. Interleaving these to obtain $q_2 q_3 ltbs$ changes only the predecessor of the first move of s, which is an O-move from σ's point of view, so σ is prepared to play this sequence, and hence

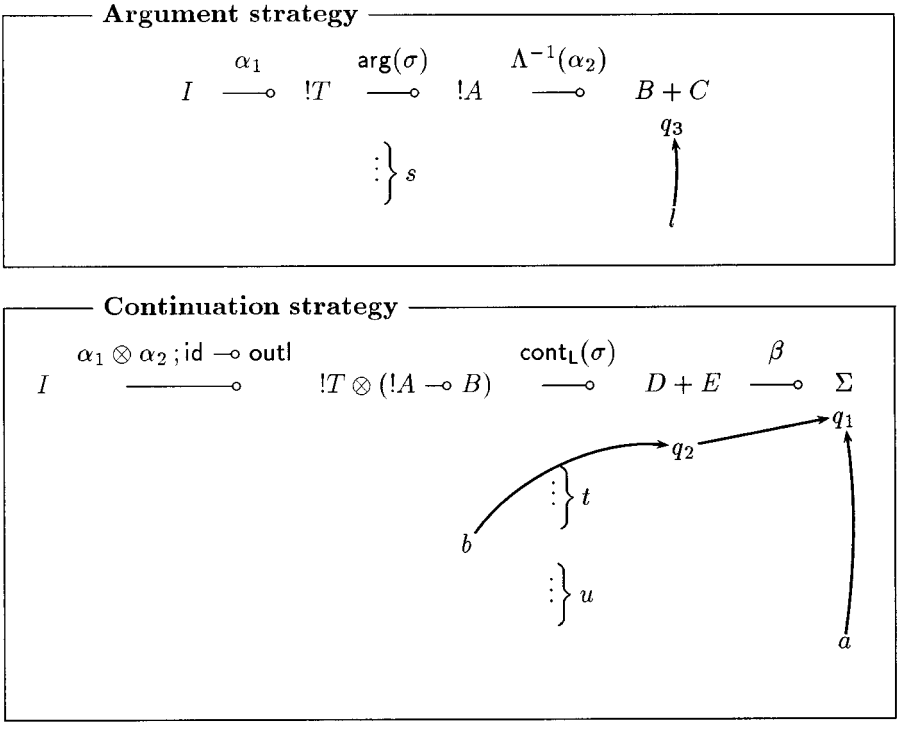


FIG. 8. Success of $\mathsf{arg}(\sigma)$ and $\mathsf{cont}_{\mathsf{L}}(\sigma)$.

$\mathsf{cont}_{\mathsf{L}}(\sigma)$ is prepared to play $q_2 tbs$. We also know that $q_3 sl \in \alpha_1 \otimes \alpha_2$. Therefore $bs \in \alpha_1 \otimes \alpha_2 ; (\mathsf{id} \multimap \mathsf{outl})$. But $t \in \alpha_1 \otimes \alpha_2 ; (\mathsf{id} \multimap \mathsf{outl})$, so $tbs \in \alpha_1 \otimes \alpha_2 ; (\mathsf{id} \multimap \mathsf{outl})$. We can therefore conclude that $q_2 tbs \in \alpha_1 \otimes \alpha_2 ; (\mathsf{id} \multimap \mathsf{outl}) \parallel \mathsf{cont}_{\mathsf{L}}(\sigma)$. This means that u must have the form sv for some v.

It is clear that the part of $q_2 q_3 sltbv$ relevant to β is the same as in the successful test

$$(\alpha_1 \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) ; \mathsf{cont}_{\mathsf{L}}(\sigma) ; \beta = \top,$$

so we just need to show that σ and $\alpha_1 \otimes \alpha_2$ are prepared to play their parts.

We already know that $q_2 tbsv \in \mathsf{cont}_{\mathsf{L}}(\sigma)$, so $q_2 q_3 ltbsv \in \sigma$. Rearranging to produce $q_2 q_3 sltbv$ affects the predecessors of l and the first moves of s and v, but these are all O-moves from σ's point of view, so σ is prepared to play this sequence.

Finally we must show that $\alpha_1 \otimes \alpha_2$ is prepared to play $q_3 sltbv$. We have $q_3 sl \in \alpha_1 \otimes \alpha_2$ and $tbsv \in \alpha_1 \otimes \alpha_2 ; (\mathsf{id} \multimap \mathsf{outl})$. This implies that $\alpha_1 \otimes \alpha_2$ is prepared to play $tq_3 slbv$. Permuting this to obtain $q_3 sltbv$ changes only the predecessors of the first move of t, q_3, and b, all of which are O-moves from the point of view of $\alpha_1 \otimes \alpha_2$, so the proof is complete. ∎

We can now describe the decomposition of strategies in general.

DEFINITION. Let $T = (A_1 \Rightarrow B_1 + C_1) \,\&\, \cdots \,\&\, (A_n \Rightarrow B_n + C_n)$ and let $\sigma : T \to D + E$ be a map in $\mathscr{I}$; in other words σ is a strategy for $!T \multimap D + E$. We

decompose σ according to its response to the initial move q of $D + E$. There are four cases.

- σ has no response. Then we write $\sigma = \bot$.
- $\mathsf{fun}(\sigma)(q) = l$. Let $\sigma_1 = \{s \mid qls \in \sigma\}$, so that $\sigma_1 = \sigma\,;\mathsf{outl} : !T \multimap D$, that is to say $\sigma_1 \colon T \to D$ in $\mathscr{I}$. Note that if $|\sigma|$ is finite then $|\sigma_1| < |\sigma|$. We write $\sigma = \mathsf{L}(\sigma_1)$ in this case.
- $\mathsf{fun}(\sigma)(q) = r$. As in the previous case, we obtain $\sigma_1 : T \to E$ and write $\sigma = \mathsf{R}(\sigma_1)$.
- $\mathsf{fun}(\sigma)(q) = q'$, the initial question of some $A_i \Rightarrow B_i + C_i$. For simplicity of notation, let $A = A_i$, $B = B_i$, and $C = C_i$, and permute the components of T so that $T = T' \,\&\, (A \Rightarrow B + C)$. For any two games G_1 and G_2, we have $!(G_1 \,\&\, G_2) = !G_1 \otimes !G_2$, so $\sigma : !T' \otimes !(A \Rightarrow B + C) \multimap D + E$. By separation of head occurrence (Lemma 2.30), we obtain

$$\sigma' : !T' \otimes !(A \Rightarrow B + C) \otimes (!A \multimap B + C) \multimap D + E.$$

Applying the decomposition previously described gives us the following strategies.

$$\mathsf{arg}(\sigma') : !(T \,\&\, A \Rightarrow B + C) \multimap !A$$

$$\mathsf{cont}_{\mathsf{L}}(\sigma') : !(T \,\&\, A \Rightarrow B + C) \otimes (!A \multimap B) \multimap D + E$$

$$\mathsf{cont}_{\mathsf{R}}(\sigma') : !(T \,\&\, A \Rightarrow B + C) \otimes (!A \multimap C) \multimap D + E.$$

Define $\sigma_1 = \mathsf{arg}(\sigma')\,;\mathsf{der}$, so that $\sigma_1 : T \,\&\, (A \Rightarrow B + C) \to A$; define $\sigma_2 = \mathsf{id} \otimes \mathsf{der}\,;\mathsf{cont}_{\mathsf{L}}(\sigma')$, so that $\sigma_2 \colon T \,\&\, (A \Rightarrow B + C) \,\&\, (A \Rightarrow B) \to D + E$; similarly define $\sigma_3 \colon T \,\&\, (A \Rightarrow B + C) \,\&\, (A \Rightarrow C) \to D + E$. Note that if $|\sigma|$ is finite then $|\sigma_1|$, $|\sigma_2|$ and $|\sigma_3| < |\sigma|$. We then write $\sigma = \mathsf{C}(\sigma_1, \sigma_2, \sigma_3)$.

This decomposition is exhaustive: every strategy for such a type has one of these forms.

5.3. Definability

We now come to the climax of our work on **FPC** : the definability result, from which full abstraction follows.

PROPOSITION 5.8 (Definability). *Let A and B be finite types and $\sigma : A \to B$ be a finite strategy. Then σ is definable.*

Proof. By induction on $|\sigma|$, which is finite by hypothesis. If $|\sigma| = 0$ then $\sigma = \{\varepsilon\} = \bot$ which is trivially definable. Suppose that the result holds for all strategies σ' at finite types with $|\sigma'| < |\sigma|$. By Lemma 5.3, we can assume that A and B are basic types ; proceed by induction on the structure of B. If $B = I$ then again $\sigma = \bot$ which is definable. If $B = B_1 \,\&\, B_2$ then $\sigma = \langle \sigma_1, \sigma_2 \rangle$ for some $\sigma_1 : A \to B_1$ and $\sigma_2 : A \to B_2$ with $|\sigma_1|, |\sigma_2| \leqslant |\sigma|$. By the inner inductive hypothesis, σ_1 and σ_2 are definable by expressions e_1 and e_2 ; so σ is defined by (e_1, e_2). Finally, if

$B = B_1 \Rightarrow B_2 + B_3$, uncurrying gives $\sigma' : A \,\&\, B_1 \to B_2 + B_3$. If σ' is definable by some expression

$$\mathtt{x} : A, \mathtt{y} : B_1 \vdash e : B_1 + B_2$$

then σ is defined by $\mathtt{x} : A \vdash \lambda \mathtt{y} : B_1.e$, so it suffices to show that σ' is definable. The type of σ' is suitable to apply the decomposition, so one of the following is true.

- $\sigma' = \bot$,
- $\sigma' = \mathsf{L}(\sigma_1)$,
- $\sigma' = \mathsf{R}(\sigma_1)$, or
- $\sigma' = \mathsf{C}(\sigma_1, \sigma_2, \sigma_3)$,

where $|\sigma_1|, |\sigma_2|, |\sigma_3| < |\sigma'| = |\sigma|$. So by the outer inductive hypothesis, each substrategy σ_i is definable. We now use the success lemma to show that σ' is itself definable.

In the first case, σ' is trivially definable. In the second case, $\sigma = \sigma_1 ; \iota_1$, so since σ_1 is definable by an expression e, σ is defined by $\mathtt{inl}(e)$. The third case is similar.

Turning to the fourth case, let us rename the types to correspond to the names in the success lemma, considering σ' as a strategy:

$$\sigma' : !(T' \,\&(A \Rightarrow B + C)) \multimap D + E.$$

By function space extensionality (Lemma 2.33), it suffices to consider the behaviour of $\alpha^\dagger ; \sigma' ; \beta$ where $\alpha : I \multimap T' \,\&\, (A \Rightarrow B + C)$ and $\beta : D + E \multimap \Sigma$; and it suffices to consider only the case when β is not the constantly $\top$ strategy. Making use of the identity of $!(T \,\&(A \Rightarrow B + C))$ and $!T \otimes !(A \Rightarrow B + C)$, we can split $\alpha^\dagger$ into $\alpha_1^\dagger \otimes \alpha_2^\dagger$, so we are considering

$$\alpha_1^\dagger \otimes \alpha_2^\dagger ; \sigma' ; \beta.$$

Defining σ'' by separation of head occurrence, as in the description of the decomposition, this is the same as

$$\alpha_1^\dagger \otimes \alpha_2^\dagger \otimes \alpha_2 ; \sigma'' ; \beta.$$

By the success lemma (Lemma 5.7), this gives $\bot$ if and only if one of the following conditions holds.

- $\alpha_1^\dagger \otimes \alpha_2^\dagger ; \mathsf{arg}(\sigma'') ; \Lambda^{-1}(\alpha_2) = m ; \mathsf{inl}$ for some m and

$$\alpha_1^\dagger \otimes \alpha_2^\dagger \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl}) ; \mathsf{cont}_{\mathsf{L}}(\sigma'') ; \beta = \top.$$

- $\alpha_1^\dagger \otimes \alpha_2^\dagger ; \mathsf{arg}(\sigma'') ; \Lambda^{-1}(\alpha_2) = m ; \mathsf{inr}$ for some m and

$$\alpha_1^\dagger \otimes \alpha_2^\dagger \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outr}) ; \mathsf{cont}_{\mathsf{R}}(\sigma'') ; \beta = \top.$$

By the bang lemma (Lemma 2.26), $\sigma_1^\dagger = \mathsf{arg}(\sigma'')$, so

$$\alpha_1^\dagger \otimes \alpha_2^\dagger ; \mathsf{arg}(\sigma'') ; \mathcal{A}^{-1}(\alpha_2) = \alpha ; \langle \pi_2, \sigma_1 \rangle ; \mathsf{ev}.$$

We also have

$$\alpha ; \langle \mathsf{id}, \pi_2 ; (\mathsf{id} \Rightarrow \mathsf{der} ; \mathsf{outl}) \rangle ; \sigma_2 = (\alpha_1^\dagger \otimes \alpha_2^\dagger \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl})) ; \mathsf{cont}_\mathsf{L}(\sigma''),$$

so defining ϕ to be the map

$$\begin{array}{c} (T' \,\&\, (A \Rightarrow B + C)) \,\&\, B \\ \downarrow{\scriptstyle \pi_1} \\ T' \,\&\, (A \Rightarrow B + C) \\ \downarrow{\scriptstyle \langle \mathsf{id}, \pi_2 ; (\mathsf{id} \Rightarrow \mathsf{der}; \mathsf{outl}) \rangle} \\ T' \,\&\, (A \Rightarrow B + C) \,\&\, (A \Rightarrow B) \\ \downarrow{\scriptstyle \sigma_2} \\ D + E \end{array}$$

we have

$$\langle \alpha, \gamma \rangle ; \phi = \alpha_1^\dagger \otimes \alpha_2^\dagger \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outl}) ; \mathsf{cont}_\mathsf{L}(\sigma'')$$

for any strategy γ of suitable type. Note that $\mathsf{der} ; \mathsf{outl}$ is definable by the term

$$\mathtt{outl(x)} = \mathtt{case\ x\ of\ inl(x_1).x_1\ or\ inr(x_2)}\,.\,\Omega,$$

so that since σ_2 is definable, so is ϕ. Similarly we define ψ such that

$$\langle \alpha, \gamma \rangle ; \psi = \alpha_1^\dagger \otimes \alpha_2^\dagger \otimes (\alpha_2 ; \mathsf{id} \multimap \mathsf{outr}) ; \mathsf{cont}_\mathsf{R}(\sigma'').$$

The map $\mathsf{der} ; \mathsf{outr}$ is definable by a term $\mathtt{outr(x)}$ as above, so since σ_3 is definable, so is ψ. We now have $\alpha^\dagger ; \sigma' ; \beta = \top$ if and only if one of the following is true.

- $\alpha ; \langle \pi_2, \sigma_1 \rangle ; \mathsf{ev} = m ; \iota_1$ for some m and $\langle \alpha, \gamma \rangle ; \phi ; \beta = \top$ for all γ,
- $\alpha ; \langle \pi_2, \sigma_1 \rangle ; \mathsf{ev} = m ; \iota_2$ for some m and $\langle \alpha, \gamma \rangle ; \psi ; \beta = \top$ for all γ.

So $\alpha ; \sigma' ; \beta = \top$ if and only if

$$\alpha ; \langle \mathsf{id}, \langle \pi_2, \sigma_1 \rangle ; \mathsf{ef} \rangle ; \mathsf{dist} ; [\phi, \psi] ; \beta = \top,$$

which is to say that

$$\sigma' \approx \langle \mathsf{id}, \langle \pi_2, \sigma_1 \rangle ; \mathsf{ev} \rangle ; \mathsf{dist} ; [\phi, \psi].$$

But σ_1 is definable by an expression e_1, so $\langle \pi_2, \sigma_1 \rangle ; \mathsf{ev}$ is defined by $[\![\mathtt{x}(e_1)]\!]$, where $\mathtt{x}$ is the variable associated with the type $A \Rightarrow B + C$. Suppose the ϕ and ψ are defined by e_2 and e_3. Then by the definition of the semantics of **FPC**, we have

$$\sigma' \approx [\![\mathtt{case}\ \mathtt{x}(e_1)\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).e_2\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).e_3]\!]$$

so that σ' is definable, completing the proof. ▮

Putting together Propositions 5.2 and 5.8, we obtain

THEOREM 5.9 (Full abstraction). *The model $\mathscr{E}$ of* **FPC** *is fully abstract.*

Proof. By Proposition 5.2, it suffices to show that finite strategies at finite types are definable ; but this is exactly Proposition 5.8. ▮

6. RELATED AND FUTURE WORK

Having developed a fully abstract model of **FPC**, one would hope to be able to use it to analyse the language, for example to find reasoning principles for proving observational equality of terms. Some results of this kind have already been obtained, which we will now sketch.

It is reasonably straightforward to use the definability result of the previous section to show that the following grammar gives a class of **FPC** terms sufficient to denote all the compact strategies ; we call these the *compact terms.*

$$\begin{aligned} K ::= {} & \Omega \mid \mathtt{inl}(K) \mid \mathtt{inr}(K) \mid \mathtt{case}\ \alpha\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).K_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).K_2 \\ & \mid (K_1, K_2) \mid \lambda \mathtt{x} : \tau.K \mid \mathtt{intro}(K). \\ \alpha ::= {} & \mathtt{x} \mid \alpha K \mid \mathtt{outl}(\alpha) \mid \mathtt{outr}(\alpha) \\ & \mid \mathtt{fst}(\alpha) \mid \mathtt{snd}(\alpha) \mid \mathtt{elim}(\alpha). \end{aligned}$$

This allows us to prove a strong context lemma for **FPC** : two terms M and N of type τ are observationally equivalent if and only if for any compact term F of type $A \to \mathtt{null} + \mathtt{null}$, FM converges if and only if FN does. This is strictly stronger than the usual context lemma for such languages because of the restriction on the use of application in the term F: the only applications are to terms from the grammar above. This in turn allows a purely syntactic proof of full abstraction for models of the (untyped) lazy and call-by-value λ-calculi. The models are described indirectly, by translation into **FPC**, and the new context lemma is used to establish full abstraction. Details of this work appear in [McC98, McC96]. The model of the lazy λ-calculus is isomorphic to that previously given in the setting of history-free strategies [AM95a].

The results of the present paper all work at the extensional level, that is to say, in the category $\mathscr{E}$: we do not even have a model of **FPC** in $\mathscr{I}$. This is in sharp contrast with the earlier results on **PCF**, which are stronger because of the tight

intensional correspondence between programs and strategies. Several groups are currently engaged in work addressing this problem and with it the question of modelling call-by-value languages intentionally. Differing approaches are being considered by Honda and Yoshida and by Abramsky in conjunction with the present author.

There has also been a good deal of other work aimed at extending, improving, and gaining more understanding of the games models of **PCF** first presented in [AJM97, HO97, Nic94]. Danos, Herbelin, and Regnier have shown that the dynamics of those games models correspond to the operation of two already known abstract machines for λ-reduction [DHR96]; and Baillot [Bai95] has given details of the correspondence between the model of [AJM97] and Girard's geometry of interaction [Gir89].

More recently, considerable advances have been made by considering the effect of relaxing the conditions imposed on the strategies described in the present paper. It has been discovered that relaxing the innocence condition gives a fully abstract model of Idealised Algol, a prototypical imperative language with procedures and local variables [AM96]; while relaxing the bracketing condition, but retaining innocence, gives a fully abstract model of **PCF** extended with the control operator `call/cc` [Lai97]. This may also lead to a better understanding of the connections between game semantics and the intensional semantics provided by sequential algorithms [Cur93].

Note added in proof. Between the submission of this paper for publication and its acceptance, the state of the art in game semantics has progressed considerably. The foremost advance from the point of view of the work presented here is that an *intensional* model of sums has been discovered which gives rise to a model of **FPC** in the category $\mathscr{I}$. After quotienting to $\mathscr{E}$ this model coincides with the one presented here. In essence, the change is to model the sum of games A and B as $!A + !B$, that is, to make use of Girard's translation of intuitionistic disjunction into linear logic [Gir87]. This same advance yields an intensional account of call-by-value programming languages and allows fully abstract models to be built for them. This discovery was first made in a slightly different setting by Honda and Yoshida [HY97]; Abramsky and the present author then showed that no changes were required to the games presented here in order for the construction to be carried out [AM98a, McC97]. In a different vein, much richer programming languages incorporating sophisticated store and control features have been brought into the scope of game semantics and given fully abstract models. See, for example, [AM97, AHM98, AM98b, Lai98]. A model of classical linear logic has been obtained by embracing nondeterminism [BDER97], and the relationship between games models and other models of linear logic has been examined [BDER98].

ACKNOWLEDGMENTS

The author thanks Samson Abramsky, Andrew Pitts, Peter O'Hearn, Julian Rathke, and Steve Brooks for their help in both the research and the writing of this paper. The paper has a long history, and the author has had several affiliations and sources of financial support during this time, so thanks are extended to the EPSRC, Imperial College, St John's College, Oxford, and Oxford University Computing Laboratory.

Received December 1996.

REFERENCES

[AHM98] Abramsky, S., Honda, K., and McCusker, G. (1998), A fully abstract game semantics for general references, *in* "Proceedings, Thirteenth Annual IEEE Symposium on Logic in Computer Science," pp. 334–344, IEEE Computer Society Press (1994), Los Alamitos, CA.

[AJ94] Abramsky, S., and Jagadeesan, R. (1994), Games and full completeness for multiplicative linear logic, *J. Symbolic Logic* **59**, 543–574. Also appeared as Technical Report 92/24 of the Department of Computing, Imperial College of Science, Technology and Medicine.

[AJM94] Abramsky, S., Jagadeesan, R., and Malacaria, P. (1994), Full abstraction for PCF (extended abstract, *in* "Theoretical Aspects of Computer Software. International Symposium TACS'94, Sendai, Japan, April 1994" (M. Hagiya and J. C. Mitchell, Eds.), Lecture Notes in Computer Science, Vol. 789, pp. 1–15, Springer-Verlag, Berlin.

[AJM97] Abramsky, S., Jagadeesan, R., and Malacaria, P. (1997), Full abstraction for PCF, *Inform. and Comput*.

[AM95a] Abramsky, S., and McCusker, G. (1995), Games and full abstraction for the lazy λ-calculus, *in* "Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science," pp. 234–243, IEEE Computer Society Press, Los Alamitos, CA.

[AM95b] Abramsky, S., and McCusker, G. (1995), Games for recursive types, *in* "Theory and Formal Methods of Computing 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods" (C. L. Hankin, I. C. Mackie, and R. Nagarajan, Eds.), Imperial College Press, Cambridge, UK.

[AM96] Abramsky, S., and McCusker, G. (1996), Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions (extended abstract), *in* "Proceedings of 1996 Workshop on Linear Logic," Electronic Notes in Theoretical Computer Science, Vol. 3, Elsevier, Amsterdam.

[AM97] Abramsky, S., and McCusker, G. (1997), Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions, *in* "Algol-like Languages" (P. W. O'Hearn and R. D. Tennent, Eds.), Vol. 2, pp. 297–329, Birkhäuser, Basel.

[AM98a] Abramsky, S., and McCusker, G. (1998), Call-by-value games, *in* "Computer Science Logic: 11th International Workshop Proceedings" (M. Nielsen and W. Thomas, Eds.), Lecture Notes in Computer Science, pp. 1–17, Springer-Verlag, Berlin.

[AM98b] Abramsky, S., and McCusker, G. (1998), Full abstraction for Idealized Algol with passive expression, *Theoret. Comput*.

[AM98c] Abramsky, S., and McCusker, G. (1998), Game semantics, Notes to accompany Samson Abramsky's lectures at the 1997 Marktoberdorf Summer School, to appear.

[Bai95] Baillot, P. (1995), Stratégie d'Abramsky–Jagadeesan–Malacaria et géométrie de l'interaction (résumé), Unpublished manuscript.

[BDER97] Baillot, P., Danos, V., Ehrhard, T., and Regnier, L. (1997), Believe it or not, AJM's games model is a model of classical linear logic, *in* "Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science," IEEE Computer Society Press, Los Alamitos, CA.

[BDER98] Baillot, P., Danos, V., Ehrhard, T., and Regnier, L. (1998), Timeless games, *in* "Computer Science Logic: 11th International Workshop Proceedings" (M. Nielsen and W. Thomas, Eds.), Lecture Notes in Computer Science, pp. 56–77, Springer-Verlag, Berlin.

[CCF94] Cartwright, R., Curien, P.-L., and Felleisen, M. (1994), Fully abstract semantics for observably sequential languages, *Inform. and Comput*. **111**, 297–401.

[Cur93] Curien, P.-L. (1993), "Categorical Combinators, Sequential Algorithms and Functional Programming," Progress in Theoretical Computer Science, Birkhäuser, Basel.

[DHR96] Danos, V., Herbelin, H., and Regnier, L. (1996), Game semantics and abstract machines, *in* "Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science," pp. 394–405, IEEE Computer Society Press, Los Alamitos, CA.

[Eda93] Edalat, A. (1993), Self-duality, minimal invariant objects and Karoubi invariance in information categories, *in* "Theory and Formal Methods 1993: Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods" (G. L. Burn, S. J. Gay, and M. D. Ryan, Eds.), Springer-Verlag Workshops in Computer Science, Springer-Verlag, Berlin.

[ES93] Edalat, A., and Smyth, M. (1993), I-categories as a framework for solving domain equations, *Theoret. Comput. Sci.* **115**, 77–106.

[Fio96] Fiore, M. P. (1996), "Axiomatic Domain Theory in Categories of Partial Maps," Distinguished Dissertations in Computer Science, Cambridge University Press, Cambridge, UK.

[FP94] Fiore, M. P., and Plotkin, G. D. (1994), An axiomatization of computationally adequate domain theoretic models of FPC, *in* "Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science," pp. 92–102, IEEE Computer Society Press, Los Alamitos, CA.

[Fre90] Freyd, P. J. (1990), Recursive types reduced to inductive types, *in* "Proceedings Fifth Annual IEEE Symposium on Logic in Computer Science," IEEE Computer Society Press, Los Alamitos, CA.

[Fre91] Freyd, P. J. (1991), Algebraically complete categories, *in* "Proc. 1990 Como Category Theory Conference" (A. Carbondi *et al.*, Eds.), Lecture Notes in Mathematics, Vol. 1488, pp. 95–104, Springer-Verlag, Berlin.

[Fre92] Freyd, P. J. (1992), Remarks on algebraically compact categories, *in* "Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham, 1991," LMS Lecture Notes Series, Vol. 177, Cambridge University Press, Cambridge, UK.

[Gir87] Girard, J.-Y. (1987), Linear Logic, *Theoret. Comput. Sci.* **50**, 1–102.

[Gir89] Girard, J.-Y. (1989), Geometry of interaction 1: Interpretation of system F, *in* "Logic Colloquium" (R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, Eds.), pp. 221–260, North-Holland, Amsterdam.

[Gor95a] Gordon, A. D. (1995), Bisimilarity as a theory of functional programming, *in* "Participants Proceedings of the Eleventh Conference on the Mathematical Foundations of Computer Science, New Orleans, 1995," Electronic Notes in Theoretical Computer Science, Vol. 1, Elsevier, Amsterdam.

[Gor95b] Gordon, A. D. (1995), Bisimilarity as a Theory of Functional Programming: Mini-Course, Notes Series BRICS-NS-95-3, BRICS, Department of Computer Science, University of Aarhus.

[Gun92] Gunter, C. A. (1992), "Semantics of Programming Languages: Structures and Techniques," Foundations of Computing, MIT Press, Cambridge, MA.

[HO97] Hyland, J. M. E., and Ong, C.-H. L. (1997), On full abstraction for PCF: I, II and III, *Inform. and Comput.*, in press.

[HY97] Honda, K., and Yoshida, N. (1997), Game theoretic analysis of call-by-value computation, *in* "Proceedings, 25th International Colloquium on Automata, Languages and Programming: ICALP '97" (P. Degano, R. Gorrieri, and A. Marchietti-Spaccameli, Eds.), Lecture Notes in Computer Science, Vol. 1256, pp. 225–236, Springer-Verlag, Berlin.

[IEE97] IEEE Computer Science Press, (1997), "Twelfth Annual IEEE Symposium on Logic in Computer Science."

[Lai97] Laird, J. (1997), Full abstraction for functional languages with control, *in* "Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science," pp. 58–67, IEEE Computer Society Press, Los Alamitos, CA.

[Lai98] Laird, J. (1998), "A Semantic Analysis of Control," Ph.D. thesis, University of Edinburgh.

[McC96] McCusker, G. (1996), Full abstraction by translation, *in* "Proceedings of the Third Workshop of the Theory and Formal Methods Section," Department of Computing, Imperial College, London.

[McC97] McCusker, G. (1997), Games and definability for FPC, *Bull. Symbolic Logic* **3**, 347–362.

[McC98] McCusker, G. (1998), "Games and Full Abstraction for a Functional Metalanguage with Recursive Types," Distinguished Dissertations in Computer Science, Springer-Verlag, Berlin.

[Mil77] Milner, R. (1977), Fully abstract models of types lambda-calculi, *Theoret. Comput. Sci.* **4**, 1–22.

[Nic94] Nickau, H. (1994), Hereditarily sequential functionals, *in* "Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg," Lecture Notes in Computer Science, Springer-Verlag, Berlin.

[Ong95] Ong, C.-H. L. (1995), Correspondence between operational and denotational semantics, *in* "Handbook of Logic in Computer Science" (S. Abramsky, D. Gabbay, and T. S. E. Maibaum, Eds.), Vol. 4, pp. 269–356, Oxford University Press, Oxford.

[OR95] O'Hearn, P. W., and Riecke, J. G. (1995), Kripke logical relations and PCF, *Inform. and Computation* **120**, 107–116.

[Pit96] Pitts, A. M. (1996), Relational properties of domains, *Inform. and Comput.* **127**, 66–90.

[Plo77] Plotkin, G. (1977), LCF considered as a programming language, *Theoret. Comput. Sci.* **5**, 223–255.

[Plo85] Plotkin, G. (1985), Lectures on predomains and partial functions, Notes for a course given at the Center for the Study of Language and Information, Stanford.

[Smy92] Smyth, M. B. (1992), I-categories and duality, *in* "Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham, 1991" (M. P. Fourman, P. T. Johnstone, and A. M. Pitts, Eds.), LMS Lecture Notes Series, Vol. 177, Cambridge University Press, Cambridge, UK.

[SP82] Smyth, M. B., and Plotkin, G. D. (1982), The category-theoretic solution of recursive domain equations, *SIAM J. Comput.* **11**, 761–783.

[Win93] Winskel, G. (1993), "The Formal Semantics of Programming Languages," Foundations of Computing, The MIT Press, Cambridge, MA.