Full abstraction for PCF

Abramsky, S; Jagadeesan, R; Malacaria, P

doi:10.1006/inco.2000.2930

Full Abstraction for PCF[*]

Samson.Abramsky[†] and Radha Jagadeesan[‡] and Pasquale Malacaria[§]

January 21, 2014

Abstract

An intensional model for the programming language PCF is described, in which the types of PCF are interpreted by games, and the terms by certain "history-free" strategies. This model is shown to capture definability in PCF. More precisely, every compact strategy in the model is definable in a certain simple extension of PCF. We then introduce an intrinsic preorder on strategies, and show that it satisfies some striking properties, such that the intrinsic preorder on function types coincides with the pointwise preorder. We then obtain an order-extensional fully abstract model of PCF by quotienting the intensional model by the intrinsic preorder. This is the first syntax-independent description of the fully abstract model for PCF. (Hyland and Ong have obtained very similar results by a somewhat different route, independently and at the same time).

We then consider the effective version of our model, and prove a Universality Theorem: every element of the effective extensional model is definable in PCF. Equivalently, every recursive strategy is definable up to observational equivalence.

Keywords Game semantics, full abstraction, sequentiality, PCF, functional computation, programming language semantics, Linear Logic.

1 Introduction

The Full Abstraction Problem for PCF [Plo77, Mil77, BCL85, Cur92b] is one of the longest-standing problems in the semantics of programming languages. There is quite widespread agreement that it is one of the most difficult; there is much less agreement as to what exactly the problem is, or more particularly as to the precise criteria for a solution. The usual formulation is that one wants a "semantic characterization" of the fully abstract model (by which we mean the inequationally fully abstract order-extensional model, which Milner proved to be uniquely specified up to isomorphism by these properties [Mil77]). The problem is to understand what should be meant by a "semantic characterization".

Our view is that the essential content of the problem, what makes it important, is that it calls for a semantic characterization of *sequential, functional computation at higher types*. The phrase "sequential functional computation" deserves careful consideration. On the one hand, sequentiality refers to a computational process extended over time, not a mere function; on the other hand, we want to capture just those sequential computations

[*]This research was supported by grants from UK SERC and ESPRIT Basic Research Action 6811 "CLICS II". Radha Jagadeesan was supported in part by grants from NSF and ONR. Pasquale Malacaria was supported in part by the HCM fellowship n. ERBCHBICT940947.

[†]University of Oxford, email: samson.abramsky@cs.oxac.uk

[‡]DePaul University Chicago, email: RJagadeesan@cs.depaul.edu

[§]Queen Mary University of London, email: p.malacaria@qmul.ac.uk in which the different parts or "modules" interact with each other in a purely functional fashion.

There have, to our knowledge, been just four models of PCF put forward as embodying some semantic analysis. Three are domain-theoretic: the "standard model" based on Scott-continuous functions [Plo77]; Berry's bidomains model based on stable functions [Ber79]; and the Bucciarelli-Ehrhard model based on strongly stable functions [BE91]. The fourth is the Berry-Curien model based on sequential algorithms [BC82].[1] Of these, we can say that the standard model gives a good account of functional computation at higher types, but fails to capture sequentiality, while the sequential algorithms model gives a good analysis of sequential computation, but fails to capture functional behaviour. In each case, the failure can calibrated in terms of *definability*: the standard model includes parallel functions; the sequential algorithms model includes algorithms which compute "functionals" which are sensitive to non-functional aspects of the behaviour of their arguments. The bidomains model also contains non-sequential functions; while the strongly stable model, in the light of a recent result by Ehrhard [Ehr], can be seen as the "extensional collapse" of the sequential algorithms model. In short, all these models are unsatisfactory because they contain "junk". On the other side of the coin, we have Milner's result that an order-extensional model is fully-abstract iff all its compact elements are definable.

Intensional Full Abstraction

This suggests that the key step towards solving the Full Abstraction problem for PCF is to capture PCF definability. This motivates the following definition. A model $\mathcal{M}$ (not necessarily extensional) is *intensionally fully abstract* if it is algebraic, and all its compact elements are definable in PCF. In support of this terminology, we have the fact that the fully abstract model can be obtained from an intensionally fully abstract model $\mathcal{M}$ in the following canonical fashion. Firstly, define a logical relation on $\mathcal{M}$ induced by the ordering on the ground types (which are assumed *standard*, i.e. isomorphic to the usual flat domains of natural numbers and booleans). Because of the definability properties of $\mathcal{M}$, this relation is a preorder at all types. In particular, it is *reflexive* at all types. This says that all elements of the model have extensional (functional) behaviour—there is no junk.

We can now apply Theorem 7.2.2 of [Sto88] to conclude that $\mathcal{M}$ can be collapsed by a continuous homomorphism to the fully abstract model. In short, the fully abstract model is the extensional collapse of any intensionally fully abstract model. Moreover, note that the collapsing map is a homomorphism, and in particular preserves application. This contrasts sharply with "collapses" of the standard model to obtain the fully abstract model, as in the work of Mulmuley [Mul87] and Stoughton and Jung [JS93], which are only homomorphic on the "inductively reachable" subalgebra.

Thus we propose that a reasonable factorization of the full abstraction problem is to look for a semantic presentation of an intensionally fully abstract model, which embodies a semantic analysis of sequential functional computation. The construction of such a model is our first main result; it is described in Sections 2 and 3.

We have explained how the (order-extensional, inequationally) fully abstract model can be obtained from any intensionally fully abstract model by means of a general construction, described in [Sto88]. However, this description of the fully abstract model leaves something to be desired. Firstly, just because the construction in [Sto88] is very general,

[1]Cartwright and Felleisen's model without error values turns out to be equivalent to the sequential algorithms model [CF92, Cur92a]. The main result in [CF92, Cur92a] is that the sequential algorithms model with errors is fully abstract for SPCF, an extension of PCF with a `catch` construct and errors. This is a fine result, but SPCF has a rather different flavour to PCF, and arguably is no longer purely functional in character.

it is unlikely to yield any useful information about the fully abstract model. Secondly, it is not entirely syntax-free: it refers to the *type structure* of PCF.

What would the ideal form of description of the fully abstract model be? We suggest that it should comprise the specification of a cartesian closed category whose objects are certain cpo's, given together with certain additional "intensional" structure, to be used to characterize sequentiality; and whose morphisms are continuous functions between these cpo's—not *all* continuous functions, of course, but only the sequential ones, as determined by the intensional structure. The interpretation of PCF generated from this category should then be the fully abstract model. Most of the attempts at solving the full abstraction problem of which we are aware, including Berry's bidomains, Curien's bicds, and Bucciarelli and Erhard's strongly stable functions, clearly fall within this general scheme. (Thus for example the intensional structure in bidomains is the stable ordering; for domains with coherence it is the coherence.)

In Section 4, we will explain how the category of games described in Section 2 does indeed give rise to a category of sequential domains in exactly this sense. This yields the first syntax-independent description of the fully abstract model for PCF.

A still more stringent requirement on a description of the fully abstract model is that it should yield effective methods for deciding observation equivalence on terms. For example, consider "Finitary PCF", i.e. PCF based on the booleans rather than the natural numbers. The interpretation of each type of Finitary PCF in the fully abstract model is a finite poset. A natural question is whether these finite posets can be effectively presented. Suppose that we have a category of sequential domains as described in the previous paragraph, yielding a fully abstract model of PCF. If the "intensional structure" part of the interpretation of each type could itself be specified in a finite, effective fashion, then such a model would immediately yield a positive solution to this problem. Because of its intensional character, our model does not meet this requirement: there are infinitely many strategies at each functional type of Finitary PCF. The same point occurs in one form or another with all the currently known descriptions of the fully abstract model for PCF. A remarkable result by Ralph Loader [Loa96] shows that this is in fact inevitable. Loader proved that observation equivalence for Finitary PCF is undecidable. This shows that an intensional description of the fully abstract model is the best that we can hope to do.

Related Work

The results in the present paper were obtained in June 1993 (the results on Intensional Full Abstraction in Section 3) and September 1993 (the results on the intrinsic preorder and (extensional) Full Abstraction in Section 4). They were announced on various electronic mailing lists in June and September 1993. An extended abstract of the present paper appeared in the Proceedings of the Second Symposium on Theoretical Aspects of Computer Science, which was held in Sendai in April 1994 [AJM94].

Independently, and essentially simultaneously, Martin Hyland and Luke Ong gave a different model construction, also based on games and strategies, which led to the same model of PCF, and essentially the same results on Intensional Full Abstraction. Following our work on the intrinsic preorder, they showed that similar results held for their model. What is interesting is that such similar results have been obtained by somewhat different routes. Hyland and Ong's approach is based on dialogue games and innocent strategies, in the tradition of Lorentzen's dialogue interpretations of logical proofs [Lor60, Lor61], and the work by Kleene and Gandy on the semantics of higher-type recursion theory [Gan93], while our approach is closer to process semantics and the Geometry of Interaction [AJ94a, Mal93]. Further work is needed to understand more fully the relationship between the two approaches.

Independently, Hanno Nickau obtained essentially the same model and results as Hyland and Ong [Nic94]. A very different description of the fully abstract model for PCF was obtained by Peter O'Hearn and Jon Riecke, using Kripke logical relations [OR95]. This construction is very interesting, and probably of quite general applicability, but does not appear to us to embody a specific semantic analysis of sequentiality.

Since the results described in this paper were obtained, there has been significant further progress in the use of game semantics to give fully abstract models for programming languages. These results all build on the concepts, methods and results developed in the present paper, and that of Hyland and Ong. For an expository account of some of these results, and some references, see [AM99]; there is an overview in [Abr97]. The main results of the present paper are recast in an abstract, axiomatic form in [Abr00]. There have also been some significant applications of game semantics, notably [MH99, GM00].

2 The Model

We shall refer to [AJ94a] for general background and motivation on game semantics.

We begin by fixing some notation. If X is a set, we write $X^\star$ for the set of finite sequences (words, strings) on X. We shall use s, t, u, v and primed and subscripted variants of these to denote sequences, and a, b, c, d, m, n and variants to denote elements of these sequences. Concatenation of sequences will be indicated by juxtaposition, and we will not distinguish notationally between an element and the corresponding unit sequence. Thus *e.g.* as denotes a sequence with first element a and tail s. If $f : X \to Y$, then $f^\star : X^\star \to Y^\star$ is the unique monoid homomorphism extending f. We write $|s|$ for the length of a finite sequence, and s_i for the i'th element of s, $1 \leq i \leq |s|$. Given a set S of sequences, we write S^{even} for the subset of even length sequences and S^{odd} for the subset of odd length sequences. If $Y \subseteq X$ and $s \in X^\star$, we write $s{\restriction}Y$ for the result of deleting all occurrences of symbols not in Y from s. We write $s \sqsubseteq t$ if s is a prefix of t, *i.e.* for some u, $su = t$. We always consider sequences under this prefix ordering and use order-theoretic notions [DP90] without further comment.

Given a family of sets $\{X_i\}_{i \in I}$ we write $\sum_{i \in I} X_i$ for their disjoint union (coproduct); we fix

$$\sum_{i \in I} X_i = \{(i, x) \mid i \in I, x \in X_i\}$$

as a canonical concrete representation. In particular, we write $X_1 + X_2$ for $\sum_{i \in \{1,2\}} X_i$. If $s \in (\sum_{i \in I} X_i)^\star$ and $i \in I$, we define $s{\restriction}i \in X_i$ inductively by:

$$\begin{array}{lcl} \epsilon{\restriction}i & = & \epsilon \\ ((j,a)s){\restriction}i & = & \left\{ \begin{array}{ll} a(s{\restriction}i), & i = j \\ s{\restriction}i, & i \neq j. \end{array} \right. \end{array}$$

We use $\mathtt{fst}$ and $\mathtt{snd}$ as notation for first and second projection functions. Note that with s as above, $\mathtt{fst}^\star(s)$ is a sequence of indices $i_1 \cdots i_k \in I^\star$ tracking which components of the disjoint union the successive elements of s are in.

We will also need some notation for manipulating partial functions. We write $f : X \rightharpoonup Y$ if f is a partial function from the set X to the set Y; and $fx \succeq y$ for "fx is defined and equal to y". If $f : X \rightharpoonup Y$ is an injective partial function, we write $f^* : Y \rightharpoonup X$ for the converse, which is also an injective partial function. (NB: the reader should beware of confusing $f^\star$ with f^*. In practice, this should not be a problem.) If $f, g : X \rightharpoonup Y$ are partial functions with disjoint domains of definition, then we write $f \vee g : X \rightharpoonup Y$ for the partial function obtained by taking the union of (the graphs of) f and g. We write 0_X for the everywhere-undefined partial function on X and sometimes $\mathtt{id}_X$, sometimes 1_X for the identity function on X. We shall omit subscripts whenever we think we can get away with it.

2.1 Games

The games we consider are between Player and Opponent. A *play* or *run* of the game consists of an alternating sequence of moves, which may be finite or infinite. Our plays are always with Opponent to move first.

A game is a structure $A = (M_A, \lambda_A, P_A, \approx_A)$, where

- M_A is the set of moves.
- $\lambda_A : M_A \to \{P, O\} \times \{Q, A\}$ is the labelling function.

  The labelling function indicates if a move is by Player (P) or Opponent (O), and if a move is a question (Q) or an answer (A). The idea is that questions correspond to requests for data, while answers correspond to data (*e.g.* integer or boolean values). In a higher-order context, where arguments may be functions which may themselves be applied to arguments, all four combinations of Player/Opponent with Question/Answer are possible. λ_A can be decomposed into two functions $\lambda_A^{PO} : M_A \to \{P, O\}$ and $\lambda_A^{QA} : M_A \to \{Q, A\}$.

  We write

$$\begin{array}{rcl} \{P, O\} \times \{Q, A\} & = & \{PQ, PA, OQ, OA\} \\ \langle \lambda_A^{PO}, \lambda_A^{QA} \rangle & = & \lambda_A, \\ M_A^P & = & \lambda_A^{-1}(\{P\} \times \{Q, A\}), \\ M_A^O & = & \lambda_A^{-1}(\{O\} \times \{Q, A\}), \\ M_A^Q & = & \lambda_A^{-1}(\{P, O\} \times \{Q\}), \\ M_A^A & = & \lambda_A^{-1}(\{P, O\} \times \{A\}) \end{array}$$

  etc., and define

$$\overline{P} = O, \;\; \overline{O} = P,$$

$$\overline{\lambda_A^{PO}}(a) = \overline{\lambda_A^{PO}(a)}, \;\; \overline{\lambda_A} = \langle \overline{\lambda_A^{PO}}, \lambda_A^{QA} \rangle.$$

- Let $M_A^{\circledast}$ be the set of all finite sequences s of moves satisfying:

$$\begin{array}{ll} \textbf{(p1)} & s = at \implies a \in M_A^O \\ \textbf{(p2)} & (\forall i : 1 \leq i < |s|) \; [\lambda_A^{PO}(s_{i+1}) = \overline{\lambda_A^{PO}(s_i)}] \\ \textbf{(p3)} & (\forall t \sqsubseteq s) \; (|t \restriction M_A^A| \leq |t \restriction M_A^Q|). \end{array}$$

  Then P_A, the set of valid *positions* of the game, is a non-empty prefix closed subset of $M_A^{\circledast}$.

  The conditions **(p1)**–**(p3)** can be thought of as global rules applying to all games. **(p1)** says that Opponent moves first, and **(p2)** that Opponent and Player alternate. **(p3)** is known as the *bracketing condition*, and can be nicely visualised as follows. Write each question in a play as a left parenthesis "(", and each answer as a right parenthesis ")". Then the string must be well-formed in the usual sense, so that each answer is associated with a unique previous question—the most recently asked, as yet unanswered question. In particular, note that a question by Player must be answered by Opponent, and vice versa.

- $\approx_A$ is an equivalence relation on P_A satisfying

$$\begin{array}{ll} \textbf{(e1)} & s \approx_A t \implies \lambda_A^{\star}(s) = \lambda_A^{\star}(t) \\ \textbf{(e2)} & s \approx_A t, s' \sqsubseteq s, t' \sqsubseteq t, |s'| = |t'| \implies s' \approx_A t' \\ \textbf{(e3)} & s \approx_A t, sa \in P_A \implies \exists b. \, sa \approx_A tb. \end{array}$$

  Note in particular that **(e1)** implies that if $s \approx_A t$, then $|s| = |t|$.

For example, the game for **Nat** has one possible opening move $*$ (request for data), with $\lambda_{\mathbf{Nat}}(*) = OQ$; and for each $n \in \omega$, a possible response $\underline{n}$ with $\lambda_{\mathbf{Nat}}(\underline{n}) = PA$. $\approx_{\mathbf{Nat}}$ is the identity relation on $P_{\mathbf{Nat}}$. The game for **Bool** is defined similarly.

2.2 Strategies

A *strategy* for Player in A is a non-empty subset $\sigma \subseteq P_A^{\mathtt{even}}$ such that $\overline{\sigma} = \sigma \cup \mathtt{dom}(\sigma)$ is prefix-closed, where

$$\mathtt{dom}(\sigma) = \{sa \in P_A^{\mathtt{odd}} \mid \exists b.\ sab \in \sigma\}.$$

We will be interested in a restricted class of strategies, the history-free (or history independent, or history insensitive) ones. A strategy σ is *history-free* if it satisfies

- $sab, tac \in \sigma \implies b = c$
- $sab, t \in \sigma, ta \in P_A \implies tab \in \sigma$ (equivalently, $ta \in \mathtt{dom}(\sigma)$).

Henceforth, "strategy" will always by default mean "history-free strategy".

Given any strategy σ, we can define $\mathtt{fun}(\sigma) : M_A^O \rightharpoonup M_A^P$ by

$$\mathtt{fun}(\sigma)(a) \succeq b \text{ iff } (\exists s)\ [sab \in \sigma].$$

Conversely, given $f : M_A^O \rightharpoonup M_A^P$ we can define $\mathtt{traces}(f) \subseteq (M_A^{\circledast})^{\mathtt{even}}$ inductively by:

$$\mathtt{traces}(f) = \{\epsilon\} \cup \{sab \mid s \in \mathtt{traces}(f), sa \in P_A, f(a) \succeq b\}.$$

We say that f induces the strategy $\sigma_f = \mathtt{traces}(f)$, if $\mathtt{traces}(f) \subseteq P_A$. Note that if τ is a strategy, we have

$$\mathtt{fun}(\sigma_f) \subseteq f, \qquad \sigma_{\mathtt{fun}(\tau)} = \tau$$

so there is always a least partial function on moves canonically inducing a (history-free) strategy.

Proposition 2.1 *If $f : M_A^O \rightharpoonup M_A^P$ is any partial function, then $\mathtt{traces}(f) \subseteq M_A^{\circledast}$.*

Proof Certainly any $s \in \mathtt{traces}(f)$ satisfies "O moves first" and the alternation condition. We show that it satisfies the bracketing condition by induction on $|s|$. If $s = tab$, then since $ta \in P_A$ and $|ta|$ is odd, the number of questions in ta must exceed the number of answers; hence s satisfies the bracketing condition. □

The equivalence relation on positions extends to a relation on strategies, which we shall write as $\mathrel{\underset{\approx}{\sqsubset}}$.
$\sigma \mathrel{\underset{\approx}{\sqsubset}} \tau$ iff:

$$sab \in \sigma, s' \in \tau, sa \approx s'a' \implies \exists b'.\ [s'a'b' \in \tau \wedge sab \approx s'a'b']. \tag{1}$$

By abuse of notation we write the symmetric closure of this relation as $\approx$:

$$\sigma \approx \tau \quad \text{iff} \quad \sigma \mathrel{\underset{\approx}{\sqsubset}} \tau \wedge \tau \mathrel{\underset{\approx}{\sqsubset}} \sigma.$$

Interpreting the equivalence on positions as factoring out coding conventions, $\sigma \approx \tau$ expresses the fact that σ and τ are the same modulo coding conventions. $\sigma \approx \sigma$ expresses a "representation independence" property of strategies.

Proposition 2.2 (Properties of $\mathrel{\underset{\approx}{\sqsubset}}$)
$\mathrel{\underset{\approx}{\sqsubset}}$ is a partial preorder relation (i.e. transitive) on strategies. Hence $\approx$ is a partial equivalence relation (i.e. symmetric and transitive).

Proof Suppose $\sigma \mathrel{\underset{\approx}{\sqsubset}} \tau$ and $\tau \mathrel{\underset{\approx}{\sqsubset}} \upsilon$, and $s \in \sigma$, $u \in \upsilon$, $sab \in \sigma$ and $sa \approx ua''$. By induction on $|sa|$ using the definition of $\sigma \mathrel{\underset{\approx}{\sqsubset}} \tau$ and (e3), there is $ta'b' \in \tau$ with $sab \approx ta'b'$. But then $ta' \approx ua''$, and since $\tau \mathrel{\underset{\approx}{\sqsubset}} \upsilon$, $ua''b'' \in \upsilon$ with $ta'b' \approx ua''b''$ and hence $sab \approx ta'b' \approx ua''b''$ as required. □

¿From now on, we are only interested in those history-free strategies σ such that $\sigma \approx \sigma$.We write $\mathtt{Str}(A)$ for the set of such strategies over A. If σ is such a strategy for a game A, we shall write $\sigma : A$. We write $\hat{A}$ for the set of partial equivalence classes of strategies on A, which we think of as the set of "points" of A. We write $[\sigma] = \{\tau \mid \sigma \approx \tau\}$ when $\sigma \approx \sigma$.

2.3 Multiplicatives

Tensor The game $A \otimes B$ is defined as follows. We call the games A and B the *component* games.

- $M_{A \otimes B} = M_A + M_B$, the disjoint union of the two move sets.
- $\lambda_{A \otimes B} = [\lambda_A, \lambda_B]$, the source tupling.
- $P_{A \otimes B}$ is the set of all $s \in M^{\circledast}_{A \otimes B}$ such that:
  1. *Projection condition:* The restriction to the moves in M_A (resp. M_B) is in P_A (resp. P_B).
  2. *Stack discipline:* Every answer in s must be in the same component game as the corresponding question.
- $s \approx_{A \otimes B} t$ iff $s{\upharpoonright}A \approx_A t{\upharpoonright}_A \ \wedge\ s{\upharpoonright}B \approx_B t{\upharpoonright}B \ \wedge\ \mathtt{fst}^\star(s) = \mathtt{fst}^\star(t)$.

We omit the easy proof that $\approx_{A \otimes B}$ satisfies **(e1)**–**(e3)**. Note that, if the equivalence relations $\approx_A$ and $\approx_B$ are the identities on P_A and P_B respectively, then $\approx_{A \otimes B}$ is the identity on $P_{A \otimes B}$.

The tensor unit is given by

$$I = (\varnothing, \varnothing, \{\epsilon\}, \{(\epsilon, \epsilon)\}).$$

Linear Implication The game $A \multimap B$ is defined as follows. We call the games A and B the *component* games.

- $M_{A \multimap B} = M_A + M_B$, the disjoint union of the two move sets.
- $\lambda_{A \multimap B} = [\overline{\lambda_A}, \lambda_B]$.
- $P_{A \multimap B}$ is the set of all $s \in M^{\circledast}_{A \multimap B}$ such that:
  1. *Projection condition:* The restriction to the moves in M_A (resp. M_B) is in P_A (resp. P_B).
  2. *Stack discipline:* Every answer in s must be in the same component game as the corresponding question.
- $s \approx_{A \multimap B} t$ iff $s{\upharpoonright}A \approx_A t{\upharpoonright}_A \ \wedge\ s{\upharpoonright}B \approx_B t{\upharpoonright}B \ \wedge\ \mathtt{fst}^\star(s) = \mathtt{fst}^\star(t)$.

Note that, by **(p1)**, the first move in any position in $P_{A \multimap B}$ must be in B.

We refer to the condition requiring answers to be given in the same components as the corresponding questions as the *stack discipline*. It ensures that computations must evolve in a properly nested fashion. This abstracts out a key structural feature of functional computation, and plays an important rôle in our results.

Proposition 2.3 (Switching Condition) *If a pair of successive moves in a position in $A \otimes B$ are in different components, (i.e. one was in A and the other in B), then the second move was by Opponent (i.e. it was Opponent who switched components). If two successive moves in $A \multimap B$ are in different components, the second move was by Player (i.e. it was Player who switched components).*

Proof Each position in $A \otimes B$ can be classified as in one of four "states": (O, O), i.e. an even number of moves played in both components, so Opponent to move in both; (P, O), meaning an odd number of moves played in the first component, so Player to move there, and an even number of moves played in the second component, so Opponent to play there; (O, P); and (P, P). Initially, we are in state (O, O). After Opponent moves, we are in (P, O) or (O, P), and Player can only move in the same component that Opponent has just moved in. After Player's move, we are back in the state (O, O). A simple induction shows that this analysis holds throughout any valid play, so that we can never in fact reach a state (P, P), and Player must always play in the same component as the preceding move by Opponent. A similar analysis applies to $A \multimap B$; in this case the initial state is (P, O), after Opponent's move we are in (P, P), and after Player's response we are in (O, P) or (P, O). $\square$

Note that, by comparison with [AJ94a], the Switching Condition is a consequence of our definition of the multiplicatives rather than having to be built into it. This is because of our global condition **(p1)**, which corresponds to restricting our attention to "Intuitionistic" rather than "Classical" games. Note also that the unreachable state (P, P) in $A \otimes B$ is precisely the problematic one in the analysis of Blass' game semantics in [AJ94a].

2.4 The Category of Games

We build a category $\mathcal{G}$:

$$\begin{array}{rcl} \text{Objects} & : & \text{Games} \\ \text{Morphisms} & : & [\sigma] : A \to B \text{ is a partial equivalence class } [\sigma] \in \widehat{A \multimap B} \end{array}$$

We shall write $\sigma : A \to B$ to mean that σ is a strategy in $A \multimap B$ satisfying $\sigma \approx \sigma$.

There are in general two ways of defining a (history-free) strategy or operation on strategies: in terms of the representation of strategies as sets of positions, or via the partial function on moves inducing the strategy. Some notation will be useful in describing these partial functions. Note that the type of the function f inducing a strategy in $A \multimap B$ is

$$f : M_A^P + M_B^O \rightharpoonup M_A^O + M_B^P.$$

Such a function can be written as a matrix

$$f = \left(\begin{array}{cc} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \end{array} \right)$$

where

$$\begin{array}{ll} f_{1,1} : M_A^P \rightharpoonup M_A^O & f_{1,2} : M_B^O \rightharpoonup M_A^O \\ f_{2,1} : M_A^P \rightharpoonup M_B^P & f_{2,2} : M_B^O \rightharpoonup M_B^P. \end{array}$$

For example, the twist map

$$M_A^P + M_A^O \cong M_A^O + M_A^P$$

corresponds to the matrix

$$\left(\begin{array}{cc} 0 & \mathtt{id}_{M_A^O} \\ \mathtt{id}_{M_A^P} & 0 \end{array} \right)$$

where 0 is the everywhere-undefined partial function. (Compare the interpretation of axiom links in [Gir89a].) The strategy induced by this function is the copy-cat strategy as defined in [AJ94a]. As a set of positions, this strategy is defined by:

$$\mathtt{id}_A = \{ s \in P_{A \multimap A}^{\mathtt{even}} \mid s \upharpoonright 1 = s \upharpoonright 2 \}.$$

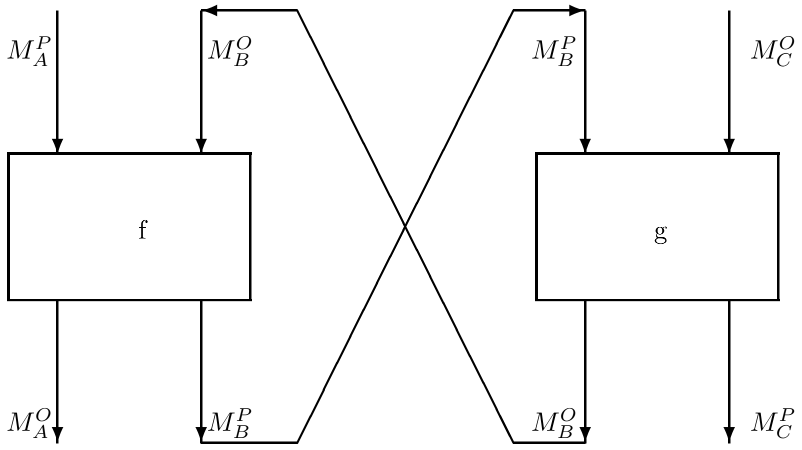
In process terms, this is a bi-directional one place buffer [Abr94]. These copy-cat strategies are the identity morphisms in $\mathcal{G}$.

Composition The composition of (history-free) strategies can similarly be defined either in terms of the set representation, or via the underlying functions on moves inducing the strategies. We begin with the set representation. Given $\sigma : A \to B$, $\tau : B \to C$, we define

$$\begin{array}{rcl} \sigma \| \tau & = & \{s \in (M_A + M_B + M_C)^\star \mid s{\upharpoonright}A, B \in \overline{\sigma},\ s{\upharpoonright}B, C \in \overline{\tau}\} \\ \sigma ; \tau & = & \{s{\upharpoonright}A, C \mid\ s \in \sigma \| \tau\}^{\texttt{even}}. \end{array}$$

This definition bears a close resemblance to that of "parallel composition plus hiding" in the trace semantics of CSP [Hoa85]; see [AJ94a] for an extended discussion of the analogies between game semantics and concurrency semantics, and [Abr94] for other aspects.

We now describe composition in terms of the functions inducing strategies. Say we have $\sigma_f : A \to B$, $\sigma_g : B \to C$. We want to find h such that $\sigma_f ; \sigma_g = \sigma_h$. We shall compute h by the "execution formula" [Gir89b, Gir89a, Gir88]. Before giving the formal definition, let us explain the idea, which is rather simple. We want to hook the strategies up so that Player's moves in B under σ get turned into Opponent's moves in B for τ, and vice versa. Consider the following picture:



Assume that the Opponent starts in C. There are two possible cases:

- The move is mapped by g to a response in C: In this case, this is the response of the function h.
- The move is mapped by g to a response in B. In this case, this response is interpreted as a move of the Opponent in B and fed as input to f. In turn, if f responds in A, this is the response of the function h. Otherwise, if f responds in B, this is fed back to g. In this way, we get an internal dialogue between the strategies f and g.

It remains to give a formula for computing h according to these ideas. This is the execution formula:

$$h = \texttt{EX}(f, g) = \bigvee_{k \in \omega} m_k.$$

The join in the definition of h can be interpreted concretely as union of graphs. It is well-defined because it is being applied to a family of partial functions with pairwise disjoint domains of definition. The functions $m_k : M_A^P + M_C^O \rightharpoonup M_A^O + M_C^P$ are defined by

$$m_k = \pi^\star \circ ((f + g) \circ \mu)^k \circ (f + g) \circ \pi.$$

The idea is that m_k is the function which, when defined, feeds an input from M_A^P or M_C^O exactly k times around the channels of the internal feedback loop and then exits from M_A^O or M_C^P. The retraction

$$\pi : M_A + M_C \lhd M_A + M_B + M_B + M_C : \pi^\star$$

is defined by

$$\pi^\star = [\mathtt{inl}, 0, 0, \mathtt{inr}] \qquad \pi = [\mathtt{in}_1, \mathtt{in}_4]$$

and the "message exchange" function $\mu : M_A^O + M_B^P + M_B^O + M_C^P \rightharpoonup M_A^P + M_B^O + M_B^P + M_C^O$ is defined by

$$\mu = 0 + [\mathtt{inr}, \mathtt{inl}] + 0.$$

Here, 0 is the everywhere undefined partial function.

The fact that this definition of composition coincides with that given previously in terms of sets of positions is proved in [AJ94a, Proposition 3].

Proposition 2.4 *Composition is monotone with respect to* $\underset{\approx}{\sqsubseteq}$*:*

$$\sigma, \sigma' : A \to B,\ \tau, \tau' : B \to C,\ \sigma \underset{\approx}{\sqsubseteq} \sigma',\ \tau \underset{\approx}{\sqsubseteq} \tau' \implies \sigma;\tau \underset{\approx}{\sqsubseteq} \sigma';\tau'.$$

Proof We follow the analysis of composition given in the proof of Proposition 1 of [AJ94a]. Suppose $\sigma \underset{\approx}{\sqsubseteq} \sigma'$, $\tau \underset{\approx}{\sqsubseteq} \tau'$, $ca \in \sigma;\tau$ and $c \approx c'$. Then $ca = u{\upharpoonright}A, C$ for uniquely determined $u = cb_1 \cdots b_k a$ such that $u{\upharpoonright}A, B \in \sigma$, $u{\upharpoonright}B, C \in \tau$. We must have $c \in M_C$. Since $\tau \underset{\approx}{\sqsubseteq} \tau'$, $c'b_1' \in \tau'$ for unique b_1', and $cb_1 \approx c'b_1'$. Now $b_1 \in \mathtt{dom}(\sigma)$ and $\sigma \underset{\approx}{\sqsubseteq} \sigma'$ implies that $b_1'b_2' \in \sigma'$ for unique b_2', and $b_1b_2 \approx b_1'b_2'$. Continuing in this way, we obtain a uniquely determined sequence $u' = c'b_1' \cdots b_k'a'$ such that $u'{\upharpoonright}A, B \in \sigma'$, $u'{\upharpoonright}B, C \in \tau'$, and $ca \approx c'a'$, as required. This argument is extended to general strings $s \in \sigma;\tau$ by an induction on $|s|$. □

We say that a string $s \in (M_{A_1} + \ldots + M_{A_n})^\star$ is *well-formed* if it satisfies the bracketing condition and the stack discipline; and *balanced* if it is well-formed, and the number of questions in s equals the number of answers. Note that these properties depend only on the string $\bar{s}$ obtained from s by replacing each question in $A_1, \ldots, A_n$ by $(_1, \ldots, (_n$ respectively, and each answer in $A_1, \ldots, A_n$ by $)_1, \cdots,)_n$ respectively.

Lemma 2.5 *The balanced and well-formed strings in* $(M_{A_1} + \cdots + M_{A_n})^\star$ *are generated by the following context-free grammar:*

$$\textsc{bal} \ ::= \ \epsilon \mid \textsc{bal}\ \textsc{bal} \mid (_i\, \textsc{bal}\,)_i \ \ (i = 1, \ldots, n)$$

$$\textsc{wf} \ ::= \ \epsilon \mid \textsc{bal}\ \textsc{wf} \mid (_i\, \textsc{wf} \ \ (i = 1, \ldots, n).$$

(More precisely, s is well-formed (balanced) iff $\bar{s}$ *is derivable from* WF *(*BAL*) in the above grammar.)*

Proof It is easy to see that the terminal strings derivable from BAL are exactly the balanced ones, and that strings derivable from WF are well-formed. Now suppose that s is well-formed. We show by induction on $|s|$ that s is derivable from WF. If s is non-empty, it must begin with a question, $s = (_i t$. If this question is not answered in s, then t is well-formed, and by induction hypothesis t is derivable from WF, hence s is derivable via the production $\textsc{wf} \to (_i\textsc{wf}$. If this question is answered, so $s = (_i u)_i v$, then $(_i u)_i$ is balanced, and hence derivable from BAL, and v is well-formed, and so by induction hypothesis derivable from WF. Then s is derivable from WF via the production $\textsc{wf} \to \textsc{bal}\ \textsc{wf}$. □

Lemma 2.6 (Projection Lemma) *If* $s \in (M_{A_1} + \cdots + M_{A_n})^\star$ *is well-formed (balanced), then so is* $s{\upharpoonright}A_{i_1}, \ldots, A_{i_k}$ *for any subsequence* $A_{i_1}, \ldots, A_{i_k}$ *of* $A_1, \ldots, A_n$*.*

Proof We use the characterization of well-formed and balanced strings from the previous lemma, and argue by induction on the size of the derivation of s from WF or BAL. Suppose s is well-formed. If s is empty, the result is immediate. If s is derivable via $\textsc{wf} \to \textsc{bal}\ \textsc{wf}$, so $s = tu$ where t is balanced and u is well-formed, then we can apply the induction hypothesis to t and u. Similarly when $s = (_i t$ where t is well-formed, we can apply the induction hypothesis to t. The argument when s is balanced is similar. □

Lemma 2.7 (Parity Lemma) *If $s \in \sigma \| \tau$ is such that $s = tmun$, where m, n are moves in the "visible" components A and C, then:*

- *if m, n are in the* same *component, then $|u \upharpoonright B|$ is even.*
- *if m, n are in* different *components, then $|u \upharpoonright B|$ is odd.*

Proof Firstly, we consider the case where all moves in u are in B. Suppose for example that m and n are both in A. Then the first move in u is by σ, while the last move is by τ, since it must have been σ which returned to A. Thus $|u|$ is even. Similarly if m and n are both in C. Now suppose that m is in A while n is in C. Then the first and last moves in u were both by σ, so $|u|$ is odd; and similarly if m is in C and n is in A.

Now we consider the general case, and argue by induction on $|u|$. Suppose m and n are both in A. Let $u = u_1 m_1 u_2$, where all moves in u_1 are in B. Suppose firstly that m_1 is in A; then $|u_1|$ is even, and by induction hypothesis $|u_2 \upharpoonright B|$ is even, so $|u \upharpoonright B|$ is even. If m_1 is in C, then $|u_1|$ is odd, and by induction hypothesis $|u_2 \upharpoonright B|$ is odd, so $|u \upharpoonright B|$ is even. The other cases are handled similarly. □

Proposition 2.8 *If $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$, then $\sigma;\tau$ satisfies the bracketing condition and the stack discipline.*

Proof By the Projection Lemma, it suffices to verify that every $s \in \sigma \| \tau$ is well-formed. We argue by induction on $|s|$. The basis is trivial. Suppose $s = tm$. If m is a question, it cannot destroy well-formedness. If m is an answer with no matching question, then by induction hypothesis t is balanced. Suppose m is in A or B; then by the Projection Lemma, $t \upharpoonright A, B$ is balanced, so m has no matching question in $s \upharpoonright A, B = (t \upharpoonright A, B)m$, contradicting $s \upharpoonright A, B \in \sigma$. A similar argument applies when m is in B or C.

So we need only consider $s = umvn$ where m, n are a matching question-answer pair. It remains to show that m and n must be in the same component. Suppose firstly that m and n both occur in A or B. Note that v is balanced, and then by the Projection Lemma, so is $v \upharpoonright A, B$. So m and n will be paired in $s \upharpoonright A, B \in \sigma$, and hence they must be in the same component. Similarly when m and n are both in B or C.

The final case to be considered is when m and n both occur in A or C. Since v is balanced, by the Projection Lemma so is $v \upharpoonright B$. It follows that $|v \upharpoonright B|$ is even, so by the Parity Lemma, m and n must be in the same component. □

Combining Propositions 2.4.2 and 2.4.6 with Proposition 2 from [AJ94a], we obtain:

Proposition 2.9 *$\mathcal{G}$ is a category.*

2.5 $\mathcal{G}$ as an autonomous category

We have already defined the object part of the tensor product $A \otimes B$, linear implication $A \multimap B$, and the tensor unit I. The action of tensor on morphisms is defined as follows. If $\sigma_f : A \rightarrow B$, $\sigma_g : A' \rightarrow B'$, then $\sigma_f \otimes \sigma_g : A \otimes A' \rightarrow B \otimes B'$ is induced by the partial function

$$\begin{aligned}
&(M_A^P + M_{A'}^P) + (M_B^O + M_{B'}^O) \\
&\cong (M_A^P + M_B^O) + (M_{A'}^P + M_{B'}^O) \overset{f+g}{\rightharpoonup} (M_A^O + M_B^P) + (M_{A'}^O + M_{B'}^P) \\
&\cong (M_A^O + M_{A'}^O) + (M_B^P + M_{B'}^P).
\end{aligned}$$

The natural isomorphisms for associativity, commutativity and unit of the tensor product:

$$\begin{array}{rcrcl}
\mathtt{assoc}_{A,B,C} & : & (A \otimes B) \otimes C & \cong & A \otimes (B \otimes C) \\
\mathtt{symm}_{A,B} & : & A \otimes B & \cong & B \otimes A \\
\mathtt{unit}_A & : & A \otimes I & \cong & A
\end{array}$$

are induced by the evident bijections on the sets of moves:

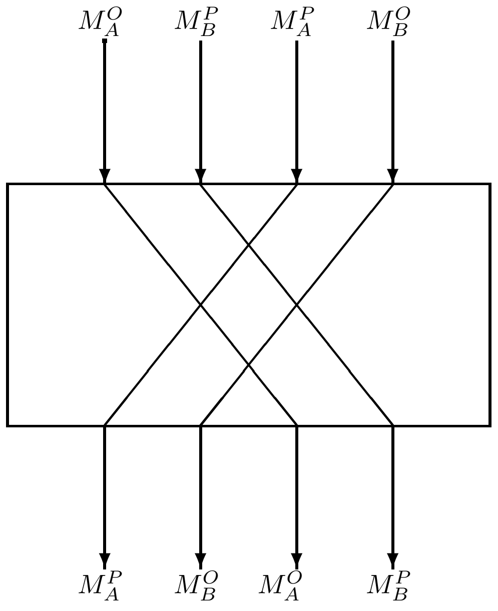
$$((M_A^P + M_B^P) + M_C^P) + (M_A^O + (M_B^O + M_C^O)) \cong ((M_A^O + M_B^O) + M_C^O) + (M_A^P + (M_B^P + M_C^P))$$

$$(M_A^P + M_B^P) + (M_B^O + M_A^O) \cong (M_A^O + M_B^O) + (M_B^P + M_A^P)$$

$$(M_A^P + \varnothing) + M_A^O \cong (M_A^O + \varnothing) + M_A^P.$$

The application morphism $\mathtt{App}_{A,B} : (A \multimap B) \otimes A \to B$ is induced by

$$((M_A^O + M_B^P) + M_A^P) + M_B^O \cong ((M_A^P + M_B^O) + M_A^O) + M_B^P.$$



This "message switching" function can be understood in algorithmic terms as follows. A demand for output from the application at M_B^O is switched to the function part of the input, $A \multimap B$; a demand by the function input for information about its input at M_A^O is forwarded to the input port A; a reply with this information about the input at M_A^P is sent back to the function; an answer from the function to the original demand for output at M_B^P is sent back to the output port B. Thus, this strategy does indeed correspond to a protocol for linear function application—linear in that the "state" of the input changes as we interact with it, and there are no other copies available allowing us to backtrack.

As for currying, given $\sigma_f : A \otimes B \to C$, $\Lambda(\sigma_f) : A \to (B \multimap C)$ is induced by

$$M_A^P + (M_B^P + M_C^O) \cong (M_A^P + M_B^P) + M_C^O \xrightarrow{f} (M_A^O + M_B^O) + M_C^P \cong M_A^O + (M_B^O + M_C^P).$$

For discussion of these definitions, and most of the verification that they work as claimed, we refer to Section 3.5 of [AJ94a].

Proposition 2.10 *1. If $\sigma \approx \sigma'$ and $\tau \approx \tau'$ then $\sigma \otimes \tau \approx \sigma' \otimes \tau'$.*

2. $\sigma \otimes \tau$ satisfies the stack discipline.

Proposition 2.11 *$\mathcal{G}$ is an autonomous category.*

2.6 Products

The game $A\&B$ is defined as follows.

$$\begin{array}{rcl} M_{A\&B} & = & M_A + M_B \\ \lambda_{A\&B} & = & [\lambda_A, \lambda_B] \\ P_{A\&B} & = & P_A + P_B \\ \approx_{A\&B} & = & \approx_A + \approx_B . \end{array}$$

The projections

$$A \xleftarrow{\texttt{fst}} A\&B \xrightarrow{\texttt{snd}} B$$

are induced by the partial injective maps

$$(M_A^P + M_B^P) + M_A^O \rightharpoonup (M_A^O + M_B^O) + M_A^P$$
$$(M_A^P + M_B^P) + M_B^O \rightharpoonup (M_A^O + M_B^O) + M_B^P$$

which are undefined on M_B^P and M_A^P respectively. Pairing *cannot* be defined in general on history-free strategies in $\mathcal{G}$; however, it can be defined on the co-Kleisli category for the comonad !, as we will see.

2.7 Exponentials

Our treatment of the exponentials is based on [AJ93]. The game $!A$ is defined as the "infinite symmetric tensor power" of A. The symmetry is built in via the equivalence relation on positions.

- $M_{!A} = \omega \times M_A = \sum_{i\in\omega} M_A$, the disjoint union of countably many copies of the moves of A. So, moves of $!A$ have the form (i, m), where i is a natural number, called the index, and m is a move of A.
- Labelling is by source tupling:

$$\lambda_{!A}(i, a) = \lambda_A(a).$$

- We write $s{\upharpoonright}i$ to indicate the restriction to moves with index i. $P_{!A}$ is the set of all $s \in M_{!A}^{\circledast}$ such that:
  1. *Projection condition:* $(\forall i)\ [s{\upharpoonright}i \in P_A]$.
  2. *Stack discipline:* Every answer in s is in the same index as the corresponding question.
- Let $S(\omega)$ be the set of permutations on ω.

$$s \approx_{!A} t \iff (\exists \pi \in S(\omega))[(\forall i \in \omega.\, s{\upharpoonright}i \approx_A t{\upharpoonright}\pi(i)) \ \wedge\ (\pi \circ \texttt{fst})^*(s) = \texttt{fst}^*(t)].$$

Dereliction For each game A and $i \in \omega$, we define a strategy

$$\texttt{der}_A^i : !A \to A$$

induced by the partial function h_i:

$$\begin{array}{rcl} h_i(j, a) & = & \begin{cases} a, & i = j \\ \text{undefined}, & i \neq j \end{cases} \\ h_i(a) & = & (i, a). \end{array}$$

In matrix form

$$h_i = \begin{pmatrix} 0 & \texttt{in}_i \\ \texttt{in}_i^* & 0 \end{pmatrix}.$$

Proposition 2.12 *1. For all i, j:*

$$\mathtt{der}^i_A \approx \mathtt{der}^j_A.$$

2. $\mathtt{der}^i_A$ satisfies the stack discipline.

By virtue of this Proposition, we henceforth write $\mathtt{der}_A$, meaning $\mathtt{der}^i_A$ for arbitrary choice of i.

Promotion A *pairing function* is an injective map

$$p : \omega \times \omega \rightarrowtail \omega.$$

Given $\sigma_f : !A \to B$ and a pairing function p, we define $\sigma^\dagger_p : !A \to !B$ as the strategy induced by the partial function $f^\dagger_p$ defined by:

$$\begin{array}{lcl} f^\dagger_p(p(i,j),a) & = & \begin{cases} (p(i,j'),a'), & f(j,a) = (j',a') \\ (i,b), & f(j,a) = b \end{cases} \\ f^\dagger_p(i,b) & = & \begin{cases} (p(i,j),a), & f(b) = (j,a) \\ (i,b'), & f(b) = b'. \end{cases} \end{array}$$

In matrix form

$$f^\dagger_p = \begin{pmatrix} t \circ (1 \times f_{1,1}) \circ t^* & t \circ (1 \times f_{1,2}) \\ (1 \times f_{2,1}) \circ t^* & 1 \times f_{2,2} \end{pmatrix}$$

where

$$t(i,(j,a)) = (p(i,j),a).$$

Proposition 2.13 *1. If $\sigma, \tau : !A \to B$, $\sigma \approx \tau$, and p, q are pairing functions, then $\sigma^\dagger_p \approx \tau^\dagger_q$.*

2. $\sigma^\dagger_p$ satisfies the stack discipline.

By virtue of this Proposition, we shall henceforth write $\sigma^\dagger$, dropping explicit reference to the pairing function.

Proposition 2.14 *For all $\sigma : !A \to B$, $\tau : !B \to C$:*

$$\begin{array}{llcl} (\mathbf{m1}) & \sigma^\dagger ; \tau^\dagger & \approx & (\sigma^\dagger ; \tau)^\dagger \\ (\mathbf{m2}) & \mathtt{der}^\dagger_A ; \sigma & \approx & \sigma \\ (\mathbf{m3}) & \sigma^\dagger ; \mathtt{der}_B & \approx & \sigma. \end{array}$$

As an immediate consequence of this Proposition and standard results [Man76]:

Proposition 2.15 $(!, \mathtt{der}, (\cdot)^\dagger)$ *is a comonad in "Kleisli form". If we define, for $\sigma : A \to B$, $!\sigma = (\mathtt{der}_A ; \sigma)^\dagger : !A \to !B$, and $\delta_A : !A \to !!A$ by $\delta_A = \mathtt{id}^\dagger_{!A}$, then $(!, \mathtt{der}, \delta)$ is a comonad in the standard sense.*

Contraction and Weakening For each game A, we define $\mathtt{weak}_A : !A \to I$ by $\mathtt{weak}_A = \{\epsilon\}$.

A *tagging function* is an injective map

$$c : \omega + \omega \rightarrowtail \omega.$$

Given such a map, the contraction strategy $\mathtt{con}^c_A : !A \to !A \otimes !A$ is induced by the function

$$\begin{pmatrix} 0 & (r \times 1) \circ \mathtt{inl}^* \vee (s \times 1) \circ \mathtt{inr}^* \\ \mathtt{inl} \circ (r^* \times 1) \vee \mathtt{inr} \circ (s^* \times 1) & 0 \end{pmatrix}$$

where $r = \omega \xrightarrow{\mathtt{inl}} \omega + \omega \xrightarrow{c} \omega$, $s = \omega \xrightarrow{\mathtt{inr}} \omega + \omega \xrightarrow{c} \omega$.

Again, it is easily verified that $\mathtt{con}^c_A \approx \mathtt{con}^{c'}_A$ for any tagging functions c, c'.

Proposition 2.16 $\mathtt{con}_A$, $\mathtt{weak}_A$ *are well-defined strategies which give a cocommutative comonoid structure on* $!A$, i.e. *the following diagrams commute:*

$$\begin{tikzcd}
!A \arrow[rr, "{[\mathtt{con}_A]}"] \arrow[d, "{[\mathtt{con}_A]}"'] & & !A\otimes!A \arrow[d, "{[\mathtt{id}_A\otimes\mathtt{con_A}]}"] \\
!A\otimes!A \arrow[r, "{[\mathtt{con}_A\otimes\mathtt{id}_A]}"'] & (!A\otimes!A)\otimes!A \arrow[r, "{[\mathtt{assoc}_A]}"'] & !A\otimes(!A\otimes!A)
\end{tikzcd}$$

$$\begin{tikzcd}
!A \arrow[r, "{[\mathtt{con}_A]}"] \arrow[d, "{[\mathtt{id}_A]}"'] & !A\otimes!A \arrow[d, "{[\mathtt{id}_A\otimes\mathtt{weak}_A]}"] \\
!A & !A\otimes I \arrow[l, "{[\mathtt{unit}_A]}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
!A \arrow[r, "{[\mathtt{con}_A]}"] \arrow[dr, "{[\mathtt{con}_A]}"'] & !A\otimes!A \arrow[d, "{[\mathtt{symm}_{A,A}]}"] \\
 & !A\otimes!A
\end{tikzcd}$$

2.8 The co-Kleisli category

By Proposition 2.7.4, we can form the co-Kleisli category $K_!(\mathcal{G})$, with:

Objects The objects of $\mathcal{G}$.

Morphisms $K_!(\mathcal{G})(A,B) = \mathcal{G}(!A,B)$.

Composition If $\sigma : !A \to B$ and $\tau : !B \to C$ then composition in $K_!(\mathcal{G})$ is given by:

$$\sigma \fatsemi \tau = \sigma^\dagger;\tau.$$

Identities The identity on A in $K_!(\mathcal{G})$ is $\mathtt{der}_A : !A \to A$.

Exponential laws

Proposition 2.17 *1. There is a natural isomorphism* $e_{A,B} : !(A\&B) \cong !A\otimes!B$.

2. $!I = I$.

Proof

1. We define $e_{A,B}$ $:!(A\&B)\multimap!A\otimes!B$ as (the strategy induced by) the map which sends $\mathtt{inl}(a,i) \in !A\otimes!B$ to $(\mathtt{inl}(a),i) \in !(A\&B)$, $(\mathtt{inl}(a),i) \in !(A\&B)$ to $\mathtt{inl}(a,i) \in !A\otimes!B$ and similarly sends $\mathtt{inr}(b,i) \in !A\otimes!B$ to $(\mathtt{inr}(b),i) \in !(A\&B)$, $(\mathtt{inr}(b),i) \in !(A\&B)$ to $\mathtt{inr}(b,i) \in !A\otimes!B$.

   We define $e^{-1}_{A,B}$: $(!A\otimes!B)\multimap!(A\&B)$ as (the strategy induced by) the map which sends $\mathtt{inl}(a,2i) \in !A\otimes!B$ to $(\mathtt{inl}(a),i) \in !(A\&B)$, $(\mathtt{inl}(a),i) \in !(A\&B)$ to $\mathtt{inl}(a,2i) \in !A\otimes!B$ and $(\mathtt{inr}(b),i) \in !(A\&B)$ to $\mathtt{inr}(b,2i+1) \in !A\otimes!B$, $\mathtt{inr}(b,2i+1) \in !A\otimes!B$ to $(\mathtt{inr}(b),i) \in !(A\&B)$.

   It is straightforward to check that $e_{A,B}, e^{-1}_{A,B}$ are strategies. Let's prove that $e_{A,B}, e^{-1}_{A,B}$ define the required isomorphism.

   - For $e_{A,B}; e^{-1}_{A,B}$: $(!A_2\otimes!B_2)\multimap(!A_1\otimes!B_1)$ (we have used different subscripts for different copies of the same game) we have that $\mathtt{inl}(a,i) \in (!A_1\otimes!B_1)$ is sent to $\mathtt{inl}(a,2i) \in (!A_2\otimes B_2)$ and $\mathtt{inr}(b,j) \in (!A_1\otimes!B_1)$ is sent to $\mathtt{inr}(b,2j+1) \in (!A_2\otimes B_2)$. This strategy is equivalent to the identity. The automorphism which witnesses the equivalence is the map which sends i in $!A_1$ to $2i$ and j in $!B_1$ to $2j+1$ (and is the identity elsewhere).

- For $e^{-1}_{A,B}; e_{A,B}$ the same map as above witnesses the equivalence of $e^{-1}_{A,B}; e_{A,B}$ with the identity.

2. Immediate by definition.

□

Products in $K_!(\mathcal{G})$

Proposition 2.18 *I is terminal in $K_!(\mathcal{G})$.*

Proof For any game A there is only one strategy in $!A \multimap I$, namely $\{\epsilon\}$. This is because I has an empty set of moves and for any opening move a in $!A$ we have $\lambda_{!A \multimap I}(a) = P$ so that Opponent has no opening move in $!A \multimap I$. □

Proposition 2.19 *$A \xleftarrow{\pi_1} A\&B \xrightarrow{\pi_2} B$ is a product diagram in $K_!(\mathcal{G})$, where*

$$\begin{array}{rcl} \pi_1 & = & !(A\&B) \xrightarrow{\mathtt{der}} A\&B \xrightarrow{\mathtt{fst}} A \\ \pi_2 & = & !(A\&B) \xrightarrow{\mathtt{der}} A\&B \xrightarrow{\mathtt{snd}} B. \end{array}$$

If $\sigma : !C \multimap A, \tau : !C \multimap B$ then their pairing $\langle \sigma, \tau \rangle : !C \multimap A\&B$ is defined by

$$\langle \sigma, \tau \rangle = !C \xrightarrow{\mathtt{con}} !C \otimes !C \xrightarrow{\sigma^\dagger \otimes \tau^\dagger} !A \otimes !B \xrightarrow{e} !(A\&B) \xrightarrow{\mathtt{der}} A\&B.$$

In fact, we have:

Proposition 2.20 *$K_!(\mathcal{G})$ has countable products.*

Cartesian closure We define $A \Rightarrow B \equiv\ !A \multimap B$.

Proposition 2.21 *$K_!(\mathcal{G})$ is cartesian closed.*

Proof We already know that $K_!(\mathcal{G})$ has finite products. Also, we have the natural isomorphisms

$$\begin{array}{rcl} K_!(\mathcal{G})(A\&B, C) & = & \mathcal{G}(!(A\&B), C) \\ & \cong & \mathcal{G}(!A \otimes !B, C) \\ & \cong & \mathcal{G}(!A, !B \multimap C) \\ & = & K_!(\mathcal{G})(A, B \Rightarrow C). \end{array}$$

Thus $K_!(\mathcal{G})$ is cartesian closed, with "function spaces" given by $\Rightarrow$. □

We shall write $\mathcal{I} = K_!(\mathcal{G})$, since we think of this category as our intensional model.

2.9 Order-enrichment

There is a natural ordering on strategies on a game A given by set inclusion. It is easily seen that (history-free) strategies are closed under directed unions, and that $\{\epsilon\}$ is the least element in this ordering. However, morphisms in $\mathcal{G}$ are actually partial equivalence classes of strategies, and we must define an order on these partial equivalence classes.

We define:

$$[\sigma] \sqsubseteq_A [\tau] \text{ iff } \sigma \underset{\approx}{\sqsubset} \tau.$$

Proposition 2.22 *$\sqsubseteq_A$ is a partial order over $\hat{A}$. The least element in this partial order is $[\{\epsilon\}]$.*

We have not been able to determine whether $(\hat{A}, \sqsubseteq_A)$ is a cpo in general. However, a weaker property than cpo-enrichment suffices to model PCF, namely *rationality*, and this property can be verified for $K_!(\mathcal{G})$.

A *pointed poset* is a partially ordered set with a least element. A cartesian closed category $\mathbb{C}$ is *pointed-poset enriched* (ppo-enriched) if:

- Every hom-set $\mathbb{C}(A, B)$ has a ppo structure $(\mathbb{C}(A, B), \sqsubseteq_{A,B}, \perp_{A,B})$.
- Composition, pairing and currying are monotone.
- Composition is *left-strict*: for all $f : A \to B$,

$$\perp_{B,C} \circ f = \perp_{A,C}.$$

$\mathbb{C}$ is *cpo-enriched* if it is ppo-enriched, and moreover each poset

$$(\mathbb{C}(A, B), \sqsubseteq_{A,B})$$

is directed-complete, and composition preserves directed suprema. $\mathbb{C}$ is *rational* if it is ppo-enriched, and moreover for all $f : A \times B \to B$:

- The chain $(f^{(k)} \mid k \in \omega)$ in $\mathbb{C}(A, B)$ defined inductively by

$$f^{(0)} = \perp_{A,B}, \qquad f^{(k+1)} = f \circ \langle \mathtt{id}_A, f^{(k)} \rangle$$

  has a least upper bound, which we denote by f^{∇}.
- For all $g : C \to A$, $h : B \to D$,

$$g \circ f^{\nabla} \circ h = \bigsqcup_{k \in \omega} g \circ f^{(k)} \circ h.$$

Altough the standard definition of categorical model for PCF is based on cpo-enriched categories, in fact rational categories suffice to interpret PCF, as we will see in Section 2.10.

Strong completeness and continuity Let A be a game, and $(\Lambda, \leqslant)$ a directed set. A family $\{[\sigma_\lambda] \mid \lambda \in \Lambda\}$ is said to be *strongly directed* if there exist strategies σ'_λ for each $\lambda \in \Lambda$ such that $\sigma'_\lambda \in [\sigma_\lambda]$ and $\lambda \leqslant \mu \;\Rightarrow\; \sigma'_\lambda \subseteq \sigma'_\mu$.

Proposition 2.23 *A strongly directed family is $\sqsubseteq$-directed. Every strongly directed family has a $\sqsubseteq$-least upper bound.*

Now consider the constructions in $\mathcal{G}$ we have introduced in previous sections. They have all been given in terms of concrete operations on strategies, which have then been shown to be compatible with the partial preorder relation $\underset{\approx}{\sqsubset}$, and hence to give rise to well-defined operations on morphisms of $\mathcal{G}$. Say that an n-ary concrete operation Φ on strategies is *strongly continuous* if it is monotone with respect to $\underset{\approx}{\sqsubset}$, and monotone and continuous with respect to subset inclusion and directed unions:

- $\sigma_1 \underset{\approx}{\sqsubset} \tau_1, \ldots, \sigma_n \underset{\approx}{\sqsubset} \tau_n \implies \Phi(\sigma_1, \ldots, \sigma_n) \underset{\approx}{\sqsubset} \Phi(\tau_1, \ldots, \tau_n)$
- $\Phi(\bigcup S_1, \ldots, \bigcup S_n) = \bigcup \{\Phi(\sigma_i, \ldots, \sigma_n) \mid \sigma_i \in S_i,\ i \in 1, \ldots, n\}$

for directed $S_1, \ldots, S_n$. (Note that for $n = 0$, these properties reduce to $\Phi \approx \Phi$.)

Proposition 2.24 *Composition, tensor product, currying and promotion are strongly continuous.*

Proposition 2.25 *$K_!(\mathcal{G})$ is a rational cartesian closed category.*

2.10 The model of PCF

PCF is an applied simply-typed λ-calculus; that is, the terms in PCF are terms of the simply-typed λ-calculus built from a certain stock of constants. As such, they can be interpreted in any cartesian closed category once we have fixed the interpretation of the ground types and the constants. The constants of PCF fall into two groups: the ground and first-order constants concerned with arithmetic manipulation and conditional branching; and the recursion combinators $\mathbf{Y}_T : (T \Rightarrow T) \Rightarrow T$ for each type T. These recursion combinators can be canonically interpreted in any rational cartesian closed category $\mathbb{C}$. Indeed, given any object A in $\mathbb{C}$, we can define $\Theta_A : \mathbf{1} \times (A \Rightarrow A) \Rightarrow A \longrightarrow (A \Rightarrow A) \Rightarrow A$ by

$$\Theta_A = [\![F : (A \Rightarrow A) \Rightarrow A \vdash \lambda f^{A \Rightarrow A}.f(Ff) : (A \Rightarrow A) \Rightarrow A]\!].$$

Now define $\mathbf{Y}_A = \Theta_A^{\triangledown} : \mathbf{1} \longrightarrow (A \Rightarrow A) \Rightarrow A$. Note that

$$\mathbf{Y}_A = \bigsqcup_{k \in \omega} \Theta_A^{(k)} = \bigsqcup_{k \in \omega} [\![\mathbf{Y}_A^{(k)}]\!],$$

where

$$\mathbf{Y}_A^{(0)} = \lambda f^{A \Rightarrow A}.\bot_A \qquad \mathbf{Y}_A^{(k+1)} = \lambda f^{A \Rightarrow A}.f(\mathbf{Y}_A^{(k)} f).$$

These terms $\mathbf{Y}_A^{(k)}$ are the standard "syntactic approximants" to $\mathbf{Y}_A$.

Thus, given a rational cartesian closed category $\mathbb{C}$, a model $\mathcal{M}(\mathbb{C})$ of PCF can be defined by stipulating the following additional information:

- For each ground type of PCF, a corresponding object of $\mathbb{C}$. This suffices to determine the interpretation of each PCF type T as an object in $\mathbb{C}$, using the cartesian closed structure of $\mathbb{C}$. (For simplicity, we shall work with the version of PCF with a single ground type N.)
- For each ground constant and first-order function of PCF, say of type T, a morphism $x : \mathbf{1} \to A$ in $\mathbb{C}$, where $\mathbf{1}$ is the terminal object in $\mathbb{C}$, and A is the object in $\mathbb{C}$ interpreting the type T. (x is a "point" or "global element" of the type A.)

We say that $\mathcal{M}(\mathbb{C})$ is a *standard model* if $\mathbb{C}(\mathbf{1}, N) \cong \mathbb{N}_\bot$, the flat cpo of the natural numbers, and moreover the interpretation of the ground and first-order arithmetic constants agrees with the standard one. We cite an important result due to Berry [Ber79, BCL85].

Theorem 2.26 (Computational Adequacy) *If $\mathcal{M}(\mathbb{C})$ is a standard model, then it is computationally adequate;* i.e. *for all programs M and ground constants $\underline{c}$,*

$$M \longrightarrow^* \underline{c} \iff [\![M]\!] = [\![\underline{c}]\!]$$

and hence the model is sound*: for all terms $M, N : T$,*

$$[\![M]\!] \sqsubseteq [\![N]\!] \implies M \sqsubseteq^{obs} N.$$

(Berry stated his result for models based on cpo-enriched categories, but only used rational closure.)

Thus to obtain a model $\mathcal{M}(K_!(\mathcal{G}))$ it remains only to specify the ground types and first-order constants. The interpretation of N as **Nat** has already been given at the end of Section 2.1. It is readily seen that $\widehat{\mathbf{Nat}} \cong \mathbb{N}_\bot$.

Ground constants For each natural number n, there is a strategy $\overline{n} : I \to \mathbf{Nat}$, given by

$$\overline{n} = \{\epsilon, *\underline{n}\}.$$

Also, $\Omega_{\mathbf{Nat}} = [\{\epsilon\}]$.

Arithmetic functions For each number-theoretic partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ there is a strategy

$$\sigma^f = \{\epsilon, *_2 *_1\} \cup \{*_2 *_1 \underline{n}_1 \underline{m}_2 \mid f(n) \succeq m\}.$$

Conditionals The strategy κ interpreting $\texttt{if0} : N \Rightarrow N \Rightarrow N \Rightarrow N$ is defined as follows: in response to the initial question, κ interrogates its first argument; if the answer is 0, then it interrogates the second argument, and copies the reply to the output; if the answer is any number greater than 0, it interrogates the third argument, and copies the reply to the output.

Proposition 2.27 *$\mathcal{M}(K_!(\mathcal{G}))$ is a standard model of PCF.*

3 Intensional Full Abstraction

3.1 PCF*c*

In order to obtain our intensional full abstraction result, it turns out that we need to consider an extension of PCF. This extension is quite "tame", and does not change the character of the language. It consists of extending PCF with a family of first order constants

$$\texttt{case}_k : N \Rightarrow \underbrace{N \Rightarrow \cdots \Rightarrow N}_{k} \Rightarrow N$$

for each $k \in \omega$. The functions that these constants are intended to denote are defined by:

$$\begin{array}{llllll} \texttt{case}_k & \bot & n_0\, n_1 \ldots\, n_{k-1} & = & \bot \\ \texttt{case}_k & i & n_0\, n_1 \ldots\, n_{k-1} & = & n_i,\ 0 \le i < k \\ \texttt{case}_k & i & n_0\, n_1 \ldots\, n_{k-1} & = & \bot,\ i \ge k. \end{array}$$

The interpretation of $\texttt{case}_k$ as a strategy is immediate: this strategy responds to the initial question by interrogating its first input; if the response is i, with $0 \le i < k$, it interrogates the $i+1$'th input and copies the answer to the output; otherwise, it has no response.

To see how harmless this extension, which we call PCF*c*, is, note that each term in PCF*c* is observationally equivalent to one in PCF. Specifically,

$$\begin{array}{l} \texttt{case}_k \equiv^{\mathsf{obs}} \lambda x^N.\lambda y_0^N.\ldots.\lambda y_{k-1}^N. \\ \qquad \texttt{if0}\ x\ y_0 \\ \qquad\quad (\texttt{if0}\ (\texttt{pred}\ x)\ y_1 \\ \qquad\qquad \vdots \\ \qquad\qquad (\texttt{if0}\ (\underbrace{\texttt{pred}\,\texttt{pred}\,\texttt{pred}}_{k}\ x)\ y_{k-1}\ \Omega)\ldots). \end{array}$$

The point is that our intensional model is sufficiently fine-grained to distinguish between these observationally equivalent terms. However, note that our results in Section 4 apply directly to PCF.

3.2 Evaluation Trees

We shall now describe a suitable analogue of Böhm trees [Bar84] for PCF*c*. These give an (infinitary) notion of normal forms for PCF*c* terms, and provide a bridge between syntax and semantics.

We use Γ, Δ to range over type environments $x_1 : T_1, \ldots, x_k : T_k$. We define $\mathbf{FET}(\Gamma, T)$, the finite evaluation trees of type T in context Γ, inductively as follows:

- $\dfrac{M \in \mathbf{FET}(\Gamma, x : T, U)}{\lambda x^T.M \in \mathbf{FET}(\Gamma, T \Rightarrow U)}$
- $\dfrac{}{\Omega, \mathtt{n} \in \mathbf{FET}(\Gamma, N)}$
- $\dfrac{\begin{array}{l}\Gamma(x) = T_1 \Rightarrow \ldots T_k \Rightarrow N, \\ P_i \in \mathbf{FET}(\Gamma, T_i), 1 \leq i \leq k, \\ Q_n \in \mathbf{FET}(\Gamma, N), n \in \omega, \\ \exists n \in \omega.\ \forall m \geq n.\ Q_n = \Omega\end{array}}{\mathtt{case}(xP_1 \ldots P_k, (Q_n \mid n \in \omega)) \in \mathbf{FET}(\Gamma, N)}$

We regard these evaluation trees as defined "up to α–equivalence" in the usual sense. Note that if we identify each

$$\mathtt{case}(xP_1 \ldots P_k, (Q_n \mid n \in \omega))$$

with

$$\mathtt{case}_l(xP_1 \ldots P_k, Q_0, \ldots, Q_{l-1})$$

for the least l such that $Q_n = \Omega$ for all $n \geq l$, then every finite evaluation tree is a term in PCFc.

We order $\mathbf{FET}(\Gamma, T)$ by the "Ω–match ordering": $M \sqsubseteq N$ if N can be obtained from M by replacing occurrences of Ω by arbitrary finite evaluation trees.

Proposition 3.1 $(\mathbf{FET}(\Gamma, T), \sqsubseteq)$ *is a pointed poset with non-empty meets. Every principal ideal is a finite distributive lattice.*

Now we define $\mathbf{ET}(\Gamma, T)$, the space of evaluation trees, to be the ideal completion of $\mathbf{FET}(\Gamma, T)$. As an immediate consequence of proposition 3.1, we have

Proposition 3.2 $\mathbf{ET}(\Gamma, T)$ *is a dI-domain. The compact elements are terms of PCFc.*

Strictly speaking, the compact elements of $\mathbf{ET}(\Gamma, T)$ are principal ideals $\downarrow(M)$, where M is a finite evaluation tree, which can be identified with a term in PCFc as explained above.

3.3 The Bang Lemma

We now prove a key technical result. This will require an additional hypothesis on games. Say that a game A is *well-opened* if the opening moves of A can only appear in opening positions. That is, for all $a \in M_A$ if $a \in P_A$ then

$$sa \in P_A \Rightarrow s = \epsilon.$$

It is easy to see that N and I are well-opened, that if A and B are well-opened so is $A \& B$ and that if B is well-opened so is $A \Rightarrow B$. Here and henceforth we blur the distinction between the type N and the game it denotes. Thus the category of well-opened games is cartesian closed, and generates the same PCF model $\mathcal{M}(\mathcal{I})$.

Now let A be well-opened and consider $s \in P^{\mathsf{even}}_{!A \multimap !B}$. Using the switching condition, we see that s can be written uniquely as

$$s = \beta_1 \cdots \beta_k$$

where each "block" β_j has the form $(i_j, b_j)t_j$, i.e. starts with a move in $!B$; every move in $!B$ occurring in β_j has the form (i_j, b') for some b', i.e. has the same index as the opening move in β_j; if β_i, β_j are two adjacent blocks then $i \neq j$; and $|\beta_j|$ is even (so each block starts with an O-move). We refer to i_j as the *block index* for β_j. For each such block index i we define s_i to be the subsequence of s obtained by deleting all blocks with index $i' \neq i$.

Some further notation. For $s \in M^*_{!A \multimap !B}$, we define

$$\mathtt{FST}(s) = \{i \mid \exists a.(i,a) \text{ occurs in } s\}$$

i.e. the set of all indices of moves in $!A$ occurring in s. Also, we write $s{\upharpoonright}A, j$ for the projection of s to moves of the form (j,a), i.e. moves in $!A$ with index j; and similarly $s{\upharpoonright}B, j$.

Lemma 3.3 *For all $\sigma : !A \multimap !B$ with A well-opened, $s \in \sigma$, and block indices i, j occurring in s:*

(i) $s_i \in \sigma$,

(ii) $i \neq j$ *implies* $\mathtt{FST}(s_i) \cap \mathtt{FST}(s_j) = \varnothing$.

Proof By induction on $|s|$. The basis is trivial. For the inductive step, write $s = \beta_1 \ldots \beta_k \beta_{k+1}$, $t = \beta_1 \ldots \beta_k$, $umm' = \beta_{k+1}$. Let the index of β_{k+1} be i. We show firstly that $(tu)_i m \in P_{!A \multimap !B}$. By the induction hypothesis, for all $j \in \mathtt{FST}((tu)_i)$, $(tu)_i{\upharpoonright}A, j = tu{\upharpoonright}A, j$, while obviously $(tu)_i{\upharpoonright}B, i = tu{\upharpoonright}B, i$. Also, m is either a move in $!B$ with index i, or a move in $!A$. In the latter case, by the switching condition the index of m is in $\mathtt{FST}((tu)_i)$. Hence the projection conditions are satisfied by $(tu)_i m$. Moreover $(tu)_i m$ is well-formed by the Projection Lemma 2.4.4. Thus $(tu)_i m \in P_{!A \multimap !B}$ as required.

By induction hypothesis, $(tu)_i \in \sigma$, and since $\sigma = \sigma_f$ is a well-defined history-free strategy, with $f(m) = m'$ since $tumm' \in \sigma$ we conclude that $(tumm')_i = (tu)_i mm' \in \sigma$. Moreover, for $j \neq i$, $(tumm')_j = (tu)_j \in \sigma$ by induction hypothesis. This establishes *(i)*.

Now note that, if tu satisfies *(ii)*, so does tum by the switching condition. Suppose for a contradiction that $tumm'$ does not satisfy *(ii)*. This means that $m' = (j,a)$, where $j \in \mathtt{FST}((tu)_{i'})$ for some $i' \neq i$ and hence that $s{\upharpoonright}A, j = s'a$ where $s' \neq \epsilon$, so that a is a non-opening move in A. But we have just shown that $(tu)_i mm' \in \sigma \subseteq P_{!A \multimap !B}$ and hence that $(tu)_i mm'{\upharpoonright}A, j \in P_A$. By induction hypothesis

$$\mathtt{FST}((tu)_i) \cap \mathtt{FST}((tu)_{i'}) = \varnothing$$

and hence $(tu)_i mm'{\upharpoonright}A, j = a$. Thus a is both an opening and a non-opening move of A, contradicting our hypothesis that A is well opened. □

With the same notation as in lemma 3.3:

Corollary 3.4 *(i)* $\forall j \in \mathtt{FST}(s_i)\ s_i{\upharpoonright}A, j = s{\upharpoonright}A, j$.

(ii) $\forall j \notin \mathtt{FST}(s_i)\ s_i{\upharpoonright}A, j = \epsilon$.

(iii) $s_i{\upharpoonright}B, i = s{\upharpoonright}B, i$.

(iv) $j \neq i$ *implies* $s_i{\upharpoonright}B, j = \epsilon$.

Lemma 3.5 *Let $\sigma, \tau : !A \multimap !B$ with A well-opened. If $\sigma; \mathtt{der_B} \approx \tau; \mathtt{der_B}$ then $\sigma \approx \tau$.*

Proof We prove the contrapositive. Suppose $\sigma \not\approx \tau$. Then w.l.o.g. we can assume that there exist positions $sab, s'a'$ such that $sab \in \sigma$, $s' \in \tau$, $sa \approx s'a'$, and either $s'a' \notin \mathrm{dom}(\tau)$ or $s'a'b' \in \tau$ and $sab \not\approx s'a'b'$. Let the block index of a in sa be i, and of a' in $s'a'$ be i'. Note that the block index of b in sab must also be i.

By Lemma 3.3, $(sab)_i \in \sigma$ and $s'_{i'} \in \tau$. We claim that $(sa)_i \approx (s'a')_{i'}$. Indeed, if $s = \beta_1 \ldots \beta_k$, $s' = \beta'_1 \ldots \beta'_{k'}$, then by definition of $\approx_{!A \multimap !B}$ we must have $k = k'$ and the permutation $\pi = [\pi_A, \pi_B]$ witnessing $sa \approx s'a'$ must map the block index of each β_j to that of β'_j, so that in particular $sa{\upharpoonright}B, i \approx s'a'{\upharpoonright}B, i'$. Moreover, π_A must map $\mathtt{FST}((sa)_i)$ bijectively onto $\mathtt{FST}((s'a')_{i'})$. Using Corollary 3.4 for each $j \in \mathtt{FST}((sa)_i)$, $(sa)_i{\upharpoonright}A, j = sa{\upharpoonright}A, j \approx s'a'{\upharpoonright}A, \pi_A(j) = (s'a')_{i'}{\upharpoonright}A, \pi_A(j)$.

Now let tcd be defined by replacing each $(i,m) \in !B$ in $s_i ab$ by m; and $t'c$ be defined by replacing each $(i', m') \in !B$ in $s'_i a'$ by m'. Then $tcd \in \sigma; \mathrm{der_B}{}^i$, $t' \in \tau; \mathrm{der_B}{}^{i'}$ and

$tc \approx t'c'$. We wish to conclude that $tcd, t'c'$ witness the non equivalence $\sigma; \mathtt{der_B} \not\approx \tau; \mathtt{der_B}$. Suppose for a contradiction that for some d', $t'c'd' \in \tau; \mathtt{der_B}^{i'}$ and $tcd \approx t'c'd'$. This would imply that for some b', $s'_{i'}a'b' \in \tau$ and $s_i ab \approx s'_{i'}a'b'$. Since $s'a' \in P_{!A \multimap !B}$ and τ is a well-defined history-free strategy, this implies that $s'a'b' \in \tau$. Using Lemma 3.3 and Corollary 3.4 as above, $sab \approx s'a'b'$. This yields the required contradiction with our assumptions on $sab, s'a'$. □

Proposition 3.6 (The Bang Lemma) *For all $\sigma : !A \multimap !B$ with A well opened,*

$$\sigma \approx (\sigma; \mathtt{der_B})^{\dagger}.$$

Proof By the right identity law (Prop. 2.11 **(m3)**), $\sigma; \mathtt{der_B} \approx (\sigma; \mathtt{der_B})^{\dagger}; \mathtt{der_B}$. By Lemma 3.5, this implies that $\sigma \approx (\sigma; \mathtt{der_B})^{\dagger}$. □

3.4 The Decomposition Lemma

In this section we prove the key lemma for our definability result. We begin with some notational conventions. We will work mostly in the cartesian closed category $\mathcal{M}(K_!(\mathcal{G}))$. We write arrows in this category as $\sigma : A \Rightarrow B$ and composition e.g. of $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ as $\tau \circ \sigma$. We will continue to write composition in the Linear Category $\mathcal{G}$ in diagram order denoted by ; . We write

$$\mathtt{Ap} : (A \Rightarrow B) \& A \Rightarrow B$$

for the application in the cartesian closed category, and "linear" application in $\mathcal{G}$ as

$$\mathtt{LAPP} : (A \multimap B) \otimes A \to B$$

All games considered in this section are assumed to be well-opened. If $s \in M^*_{D \Rightarrow B}$, we write

$$\mathtt{FST}(s) = \{ i \mid \exists d \,.(i,d) \text{ occurs in } s \}$$

i.e. the set of all indices of moves in $!D$ occurring in s.

Now we define a strategy

$$\chi : N \& N^{\omega} \Rightarrow N$$

corresponding to the **case** construct. It will actually be most convenient to firstly define the affine version

$$\chi_a : N_1 \otimes N_2^{\omega} \multimap N_0$$

where we have tagged the occurrences of N for ease of identification;

$$\chi_a = \mathtt{Pref}\{ *_0 *_1 n_1 *_{2,n} m_{2,n} m_0 \mid n, m \in \omega \}$$

i.e. χ_a responds to the initial question by interrogating its first input; if it gets the response n it interrogates the n'th component of its second input, and copies the response as its answer to the initial question.

Now we define

$$\chi = !(N \& N^{\omega}) \xrightarrow{e_{N,N^{\omega}}} !N \otimes !N^{\omega} \xrightarrow{\mathrm{der_N} \otimes \mathrm{der_{N^{\omega}}}} N \otimes N^{\omega} \xrightarrow{\chi_a} N$$

We will now fix some notation for use in the next few lemmas. Let

$$\sigma : C \& (A \Rightarrow N_2) \Rightarrow N_1$$

be a strategy where we have tagged the two occurrences of N for ease of identification. We assume that σ's response to the initial question $*_1$ in N_1 is to interrogate its second input, i.e. to ask the initial question $*_2$ in N_2. Thus any non-empty position in σ must have the form $*_1 *_2 s$. Moreover by the stack discipline any *complete* position in σ, i.e. one containing an answer to the initial question $*_1$, must have the form

$$*_1 *_2 \; s \, n_2 \, t \, n_1$$

where n_2 is the answer corresponding to the question $*_2$ (this is the sole—albeit crucial—point at which the stack condition is used in the definability proof). Thus a general description of non-empty positions in σ is that they have the form

$$*_1 *_2 s \, n_2 \, t$$

where n_2 is the answer corresponding to $*_2$, or

$$*_1 *_2 s$$

where $*_2$ is not answered in s.

Lemma 3.7 *For all $*_1 *_2 s \, n_2 \, t \in \sigma$*

(i) $*_1 *_2 n_2 t \in \sigma$

(ii) $\mathtt{FST}(s) \cap \mathtt{FST}(t) = \varnothing$.

Proof By induction on $|t|$, which must be odd. (The proof follows very similar lines to that of Lemma 3.1 in the previous section). The basis is when $t = m$, and $f(n_2) = m$, where $\sigma = \sigma_f$. Then *(i)* follows because σ is a well-defined history-free strategy, and *(ii)* holds because otherwise $m = (j, d)$ where d is both a starting move, using $*_1 *_2 n_2 m \in \sigma$, and a non-starting move, using $*_1 *_2 s n_2 t \in \sigma$, contradicting well-openedness. If $t = umm'$, then we firstly show that

$$*_1 *_2 n_2 um \in P_{C\&(A\Rightarrow N)\Rightarrow N}$$

By the induction hypothesis and the switching conditions, for all $j \in \mathtt{FST}(um)$

$$*_1 *_2 n_2 um \restriction C\&(A \Rightarrow N), j = *_1 *_2 s n_2 um \restriction C\&(A \Rightarrow N), j$$

so $*_1 *_2 n_2 um$ satisfies the projection conditions because $*_1 *_2 s n_2 um$ does. Also, $*_2 s n_2$ is balanced so by the Parity Lemma 2.4.3 $*_1 t$ is well formed, and hence $*_1 *_2 n_2 um$ is well formed. Thus

$$*_1 *_2 n_2 um \in P_{C\&(A\Rightarrow N)\Rightarrow N}$$

Now since $\sigma = \sigma_f$ is a well-defined history-free strategy with $f(m) = m'$, and $*_1 *_2 n_2 u \in \sigma$ by induction hypothesis, we must have $*_1 *_2 n_2 umm' \in \sigma$, establishing *(i)*.

For *(ii)* suppose for a contradiction that $m' = (j, d)$ for $j \in \mathtt{FST}(s)$. Then $*_1 *_2 s n_2 t \restriction C\&(A \Rightarrow N), j = s'd \in P_{C\&(A\Rightarrow N)}$, where $s' \neq \epsilon$. On the other hand, by induction hypothesis $*_1 *_2 n_2 umm' \restriction C\&(A \Rightarrow N), j = d$, and by *(i)*, $d \in P_{C\&(A\Rightarrow N)}$. This contradicts our assumption that games are well-opened. $\square$

Now we define

$$\sigma' = \{*_1 *_2 s \, n_2 n_1 \mid *_1 *_2 s \, n_2 \in \sigma\} \cup \{*_1 *_2 s \mid *_1 *_2 s \in \sigma, \; *_2 \text{ not answered in } \sigma\}$$

and for all $n \in \omega$

$$\tau_n = \{*_1 t \mid *_1 *_2 n_2 \, t \in \sigma\}$$

Lemma 3.8 *$\sigma' : C\&(A \Rightarrow N) \Rightarrow N$ and $\tau_n : C\&(A \Rightarrow N) \Rightarrow N$ ($n \in \omega$) are valid strategies.*

Proof The fact that each τ_n is a set of valid positions follows from Lemma 3.7. That σ', τ_n are history-free and satisfy the partial equivalence relation follows directly from their definitions and the fact that σ is a valid strategy. $\square$

Lemma 3.9 $\sigma \approx \mathtt{con}_C ; \sigma' \otimes \langle \tau_n \mid n \in \omega \rangle ; \chi_a$.

Proof Unpacking the definition of the RHS $\tau = \mathtt{con}_C ; \sigma' \otimes \langle \tau_n \mid n \in \omega \rangle ; \chi_a$ we see that the second and third moves of χ_a synchronize and cancel out with the first and last moves of σ' respectively, and the fourth and fifth moves of χ_a cancel out with the first and last moves of the appropriate τ_n. Thus positions in τ have the form

$$*_1 *_2 s' n_2 t' \text{ or } *_1 *_2 s'$$

where $*_1 *_2 s n_2 t$, $*_1 *_2 s$ are positions in σ, and s', t' are bijectively reindexed versions of s and t, with the property that $\mathtt{FST}(s') \cap \mathtt{FST}(t') = \varnothing$. However, by Lemma 3.7 we know that $\mathtt{FST}(s) \cap \mathtt{FST}(t) = \varnothing$, and hence

$$*_1 *_2 s' n_2 t' \approx *_1 *_2 s n_2 t$$

and $\sigma \approx \tau$ as required. □

Lemma 3.10 $\sigma \approx \chi \circ \langle \sigma', \langle \tau_n \mid n \in \omega \rangle \rangle$

Proof

$$\begin{array}{ll}
\chi \circ \langle \sigma', \langle \tau_n \mid n \in \omega \rangle \rangle & = \text{definition} \\
(\mathtt{con}; (\sigma'^{\dagger} \otimes \langle \tau_n \mid n \in \omega \rangle^{\dagger}; e; \mathtt{der})^{\dagger}; e^{-1}; \mathtt{der} \otimes \mathtt{der}; \chi_a & \approx \text{ Bang Lemma} \\
\mathtt{con}; (\sigma'^{\dagger} \otimes \langle \tau_n \mid n \in \omega \rangle^{\dagger}); e; e^{-1}; \mathtt{der} \otimes \mathtt{der}; \chi_a & \approx \\
\mathtt{con}; (\sigma'^{\dagger} \otimes \langle \tau_n \mid n \in \omega \rangle^{\dagger}); \mathtt{der} \otimes \mathtt{der}; \chi_a & \approx \\
\mathtt{con}; (\sigma'^{\dagger}; \mathtt{der} \otimes \langle \tau_n \mid n \in \omega \rangle^{\dagger}; \mathtt{der}); \chi_a & \approx \\
\mathtt{con}; (\sigma' \otimes \langle \tau_n \mid n \in \omega \rangle); \chi_a & \approx \text{Lemma 3.9} \\
\sigma. &
\end{array}$$

□

We continue with our decomposition, and define

$$\sigma'' = \{ s \mid *_1 *_2 s \in \sigma, \ *_2 \text{ not answered in } s \}$$

Lemma 3.11 $\sigma'' : C\&(A \Rightarrow N) \Rightarrow !A$ *is a well-defined strategy, and*

$$\sigma' \approx \mathtt{con}_{C\&(A \Rightarrow N)}; \pi_2 \otimes \sigma''; \mathtt{LAPP}. \quad (\dagger)$$

Proof We must firstly explain how moves in σ'' can be interpreted as being of type $C\&(A \Rightarrow N) \Rightarrow !A$. Let the index in $!(C\&(A \Rightarrow N))$ of the response by σ to the initial question $*_1$ be i_0. Then we regard all moves in $s \in \sigma''$ with index i_0 as moves in the target $!A$, and all moves with index $i \neq i_0$ as moves in the source $!(C\&(A \Rightarrow N))$. The projection conditions and stack discipline are easily seen to hold for s with respect to this type. The fact that σ'' is history-free and satisfies the partial equivalence relation follows directly from its definition and the fact that σ' is a valid strategy.

Now write τ for the RHS of (†). We diagram τ, tagging occurrences of the types for ease of reference.

$$\begin{tikzcd}
!(C_0 \& (!A_0 \multimap N_0)) \arrow[d, "\mathtt{con}"'] \\
!(C_1 \& (!A_1 \multimap N_1)) \otimes !(C_2 \& (!A_2 \multimap N_2)) \arrow[d, "\pi_2 \otimes \sigma''"'] \\
(!A_3 \multimap N_3) \otimes\ !A_4 \arrow[d, "\mathtt{LAPP}"'] \\
N_5
\end{tikzcd}$$

From the definitions LAPP plays copy-cat strategies between N_3 and N_5 and $!A_3$ and $!A_4$; π_2 plays a copy-cat strategy between $!A_3 \multimap N_3$ and a single index i_0 in $!(C_1 \& (!A_1 \multimap N_1))$; con splits $!(C_0 \& (!A_0 \multimap N_0))$ into two disjoint address spaces $!(C_0 \& (!A_0 \multimap N_0))_L$ and $!(C_0 \& (!A_0 \multimap N_0))_R$ and plays copy-cat strategies between $!(C_0 \& (!A_0 \multimap N_0))_L$ and $!(C_2 \& (!A_2 \multimap N_2))$ and between $!(C_0 \& (!A_0 \multimap N_0))_R$ and $!(C_2 \& (!A_2 \multimap N_2))$. Thus we see that the opening move in N_5 is copied to $(i_0, N_0)_L$ via N_3 and (i_0, N_1), and any response in $(i_0, N_0)_L$ is copied back to N_5. Similarly, O's moves $(i_0, !A_0)_L$ are copied to $!A_4$ via $(i_0, !A_1)$ and $!A_3$; and P's responses in $!A_4$ following σ'' are copied back to $(i_0, !A_0)_L$. Finally, O's moves in $!(C_0 \& (!A_0 \multimap N_0))_R$ are copied to $!(C_2 \& (!A_2 \multimap N_2))$, and P's responses following σ'' are copied back to $!(C_0 \& (!A_0 \multimap N_0))_R$.

As regards sequencing, the initial move $*_5$ is copied immediately as $*_{i_0,L}$. Opponent may now either immediately reply with $n_{i_0,L}$, which will be copied back as n_5, completing the play; or move in $(i_0, !A_0)_L$— the only other option by the switching condition. Play then proceeds following σ'' transposed to

$$\underline{\sigma}'' : !(C_0 \& (\ !A_0 \multimap N_0))_R \rightarrow (i_0,\ !A_0)_L,$$

until Opponent replies with some $n_{i_0,L}$ to $*_{i_0,L}$. Thus positions in τ have the form

$$*_5\ *_{i_0,L}\ \ s'\ n_{i_0,L}\ n_5 \text{ or } \ *_5\ \ *_{i_0,L}\ s'$$

where s' is a bijectively reindexed version of $s \in \sigma''$, with $s \approx s'$. Clearly $\sigma'' \approx \underline{\sigma}''$, and hence $\sigma' \approx \tau$.

□

We now prove a useful general lemma.

Lemma 3.12 *For all strategies $\gamma : C \Rightarrow (A \Rightarrow B), \delta : C \Rightarrow A$*

$$\mathtt{Ap} \circ \langle \gamma, \delta \rangle \approx \mathtt{con}_C; (\gamma \otimes \delta^\dagger); \mathtt{LAPP}.$$

Proof

$$\begin{array}{ll}
\mathtt{Ap} \circ \langle \gamma, \delta \rangle & = \text{definition} \\
(\mathtt{con}_C; \gamma^\dagger \otimes \delta^\dagger; e; \mathtt{der}_{(A \Rightarrow B) \otimes A})^\dagger; e^{-1}; \mathtt{der}_{A \Rightarrow B} \otimes \mathtt{id}_A; \mathtt{LAPP} & \approx \text{Bang Lemma} \\
\mathtt{con}_C; \gamma^\dagger \otimes \delta^\dagger; e; e^{-1}; \mathtt{der}_{A \Rightarrow B} \otimes \mathtt{id}_A; \mathtt{LAPP} & \approx \\
\mathtt{con}_C; \gamma \otimes \delta^\dagger; \mathtt{LAPP}. &
\end{array}$$

□

Now consider a game

$$(A_1 \& \ldots \& A_k) \Rightarrow N$$

where

$$A_i = (B_{i,1} \& \ldots \& B_{i,l_i}) \Rightarrow N, \quad 1 \leq i \leq k.$$

Let $\tilde{A} = A_1 \& \ldots \& A_k$, $\tilde{B}_i = B_{i,1} \& \ldots \& B_{i,l_i}$, $1 \leq i \leq k$.

We define $\bot_{\tilde{A}} : \tilde{A} \Rightarrow N$ by $\bot_{\tilde{A}} = \{\epsilon\}$ and $\mathbf{K}_{\tilde{A}}n : \tilde{A} \Rightarrow N \quad (n \in \omega)$ by $\mathbf{K}_{\tilde{A}}n = \{\epsilon, *n\}$.Thus $\bot_{\tilde{A}}$ is the completely undefined strategy of type $\tilde{A} \Rightarrow N$ while $\mathbf{K}_{\tilde{A}}n$ is the constant strategy which responds immediately to the initial question in N with the answer n.

Finally, if $1 \leq i \leq k$, and for each $1 \leq j \leq l_i$

$$\sigma_j : \tilde{A} \Rightarrow B_{i,j}$$

and for each $n \in \omega$

$$\tau_n : \tilde{A} \Rightarrow N$$

we define

$$\check{\mathbf{C}}_{\mathbf{i}}(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) : \tilde{A} \Rightarrow N$$

by

$$\check{\mathbf{C}}_{\mathbf{i}}(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) = \chi \circ \langle \mathtt{Ap} \circ \langle \pi_i, \langle \sigma_1, \ldots, \sigma_{l_i} \rangle \rangle, \langle \tau_n \mid n \in \omega \rangle \rangle.$$

Lemma 3.13 (The Decomposition Lemma (uncurried version)) *Let $\sigma : (A_1 \& \ldots \& A_n) \Rightarrow N$ be any strategy, where*

$$A_i = (B_{i,1} \& \ldots \& B_{i,l_i}) \Rightarrow N, \quad 1 \leq i \leq k.$$

Then exactly one of the following three cases applies:

(i) $\sigma = \bot_{\tilde{A}}$.

(ii) $\sigma = \mathbf{K}_{\tilde{A}}n$ *for some* $n \in \omega$.

(iii) $\sigma \approx \check{\mathbf{C}}_{\mathbf{i}}(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$
where $1 \leq i \leq k$, $\sigma_j : \tilde{A} \Rightarrow B_{i,j}$, $1 \leq j \leq l_i$, $\tau_n : \tilde{A} \Rightarrow N$, $n \in \omega$.

Proof By cases on σ's response to the initial question. If it has no response, we are in case *(i)*. If its response is an immediate answer n for some $n \in \omega$, we are in case *(ii)*. Otherwise, σ must respond with the initial question in the i'th argument, for some $1 \leq i \leq k$. In this case, write $C = A_1 \& \ldots \& A_{i-1} \& A_{i+1} \& \ldots \& A_k$. We have the natural isomorphism

$$\alpha : !(C \& A_i) \cong !\tilde{A} : \alpha^{-1}$$

so we can apply Lemma 3.10 to conclude that

$$\alpha ; \sigma \approx \chi \circ \langle \sigma', \langle \tau_n \mid n \in \omega \rangle \rangle$$

By Lemma 3.11

$$\sigma' \approx \mathtt{con} ; \pi_2 \otimes \sigma'' ; \mathtt{LAPP}$$

where $\sigma'' : C \& A_i \Rightarrow !\tilde{B}_i$. By the Bang Lemma,

$$\sigma'' \approx (\sigma'' ; \mathtt{der})^{\dagger}.$$

Moreover

$$\sigma'' ; \mathtt{der}_B : C \& A_i \Rightarrow (B_{i,1} \& \ldots \& B_{i,l_i})$$

so by the universal property of the product,

$$\sigma''; \mathtt{der}_B \approx \langle \sigma_1, \ldots, \sigma_{l_i} \rangle$$

where $\sigma_j : C \& A_i \Rightarrow B_{i,j}, \;\; 1 \leq j \leq l_i$.

Thus $\sigma' \approx \mathtt{con}; \pi_2 \otimes \langle \sigma_1, \ldots, \sigma_{l_i} \rangle^\dagger; \mathtt{LAPP}$ and by Lemma 3.12,

$$\sigma' \approx \mathtt{Ap} \circ \langle \pi_2, \langle \sigma_1, \ldots, \sigma_{l_i} \rangle \rangle$$

Thus

$$\begin{array}{rcl} \sigma & \approx & \alpha^{-1}; \alpha; \sigma \\ & \approx & \alpha^{-1}; (\chi \circ \langle \mathtt{Ap} \circ \langle \pi_2, \langle \sigma_1, \ldots, \sigma_{l_i} \rangle \rangle, \langle \tau_n \mid n \in \omega \rangle \rangle \\ & \approx & \chi \circ \langle \mathtt{Ap} \circ \langle \pi_i, \langle \alpha^{-1}; \sigma_1, \ldots, \alpha^{-1}; \sigma_{l_i} \rangle \rangle, \langle \alpha^{-1}; \tau_n \mid n \in \omega \rangle \rangle \\ & = & \check{\mathbf{C}}_i(\alpha^{-1}; \sigma_1, \ldots, \alpha^{-1}; \sigma_{l_i}, (\alpha^{-1}; \tau_n \mid n \in \omega)). \end{array}$$

□

The Decomposition Lemma in its uncurried version is not sufficiently general for our purposes. Suppose now that we have a game

$$(A_1 \& \ldots \& A_k) \Rightarrow N$$

where

$$A_i = B_{i,1} \Rightarrow \ldots B_{i,l_i} \Rightarrow N, \quad (1 \leq i \leq l_i).$$

If for some $1 \leq i \leq k$ and each $1 \leq j \leq l_i$ we have

$$\sigma_j : \tilde{A} \Rightarrow B_{i,j}$$

and for each $n \in \omega$

$$\tau_n : \tilde{A} \Rightarrow N$$

then we define

$$\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) : \tilde{A} \Rightarrow N$$

by

$$\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) = \chi \circ \langle \mathtt{Ap} \circ \langle \ldots \mathtt{Ap} \circ \langle \pi_i, \sigma_1 \rangle, \ldots, \sigma_{l_i} \rangle, \langle \tau_n \mid n \in \omega \rangle \rangle.$$

To relate $\mathbf{C}_i$ and $\check{\mathbf{C}}_i$, consider the canonical isomorphisms

$$\alpha_i : B_{i,1} \Rightarrow \ldots B_{i,l_i} \Rightarrow N \cong (B_{i,1} \& \ldots \& B_{i,l_i}) \Rightarrow N : \alpha_i^{-1} \;\; (1 \leq i \leq k)$$

Let $\tilde{\alpha} = !(\alpha_1 \& \ldots \& \alpha_k)$ so

$$\tilde{\alpha} : !(A_1 \& \ldots \& A_k) \cong !(A_1^u \& \ldots \& A_k^u)$$

where $A_i^u = (B_{i,1} \& \ldots \& B_{i,l_i}) \Rightarrow N$ is the uncurried version of A_i. Then

$$\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) \approx \tilde{\alpha}; \check{\mathbf{C}}_i(\tilde{\alpha}; \sigma_1, \ldots, \tilde{\alpha}; \sigma_{l_i}, (\tilde{\alpha}; \tau_n \mid n \in \omega)) \quad (1)$$

In terms of $\lambda-$calculus, this just boils down to the familiar equations

$$\mathtt{Curry}(f)xy = f(x, y)$$

$$\mathtt{Uncurry}(g)(x, y) = gxy$$

To see the relationship between the combinators $\bot, \mathbf{K}n$ and $\mathbf{C}$ and the syntax of PCF, we use the combinators to write the semantics of finite evaluation trees.

Given $P \in \mathbf{FET}(\Gamma, T)$ where $\Gamma = x_1 : T_1, \ldots, x_k : T_k$, we will define

$$\mathcal{S}(\Gamma \vdash P : T) : (\mathcal{S}(T_1) \& \ldots \& \mathcal{S}(T_k)) \Rightarrow \mathcal{S}(T)$$

- $\mathcal{S}(\Gamma \vdash \lambda x^T.P : T \Rightarrow U) = \Lambda(\mathcal{S}(\Gamma, x : T \vdash P : U))$
- $\mathcal{S}(\Gamma \vdash \Omega : N) = \perp_{\tilde{T}}$
- $\mathcal{S}(\Gamma \vdash n : N) = \mathbf{K}_{\tilde{T}} n$
- $\mathcal{S}(\Gamma \vdash \mathtt{case}(x_i P_1 \ldots P_{l_i}, (Q_n \mid n \in \omega)) : N) = \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$ where

$$T_i = U_{i,1} \Rightarrow \ldots U_{i,l_i} \Rightarrow N,$$
$$\sigma_j = \mathcal{S}(\Gamma \vdash P_j : U_{i,j}), \ \ 1 \le j \le l_i,$$
$$\tau_n = \mathcal{S}(\Gamma \vdash Q_n : N), \ \ n \in \omega.$$

We can now prove the general form of the Decomposition Lemma:

Proposition 3.14 (Decomposition Lemma) *Let $\sigma : (A_1 \& \ldots \& A_p) \Rightarrow (A_{p+1} \Rightarrow \ldots A_q \Rightarrow N)$ be any strategy, where*

$$A_i = B_{i,1} \Rightarrow \ldots B_{i,l_i} \Rightarrow N, \ \ 1 \le i \le q$$

We write $\tilde{C} = A_1, \ldots, A_p$, $\tilde{D} = A_{p+1}, \ldots, A_q$. (Notation : if $\tau : \tilde{C}, \tilde{D} \Rightarrow N$, then $\Lambda_{\tilde{D}}(\tau) : \tilde{C} \Rightarrow (A_{p+1} \Rightarrow \cdots \Rightarrow A_q \Rightarrow N)$.)

Then exactly one of the following three cases applies.

(i) $\sigma = \Lambda_{\tilde{D}}(\perp_{\tilde{C},\tilde{D}})$.

(ii) $\sigma = \Lambda_{\tilde{D}}(\mathbf{K}_{\tilde{C},\tilde{D}} n)$ *for some* $n \in \omega$.

(iii) $\sigma = \Lambda_{\tilde{D}}(\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)))$, *where* $1 \le i \le q$, *and*

$$\begin{array}{rcll} \sigma_j : \tilde{C}, \tilde{D} & \Rightarrow & B_{i,j}, & 1 \le j \le l_i, \\ \tau_n : \tilde{C}, \tilde{D} & \Rightarrow & N, & n \in \omega. \end{array}$$

Proof Let $\alpha_i : A_i \cong A_i^u : \alpha^{-1}$ be the canonical isomorphism between A_i and its uncurried version

$$A_i^u = (B_{i,1} \& \ldots \& B_{i,l_i}) \Rightarrow N$$

for each $1 \le i \le q$.

Let

$$\tilde{\alpha} = !(\alpha_1 \& \ldots \& \alpha_p \& \alpha_{p+1} \& \ldots \& \alpha_q).$$

Note that

$$\begin{array}{rcll} \perp_{\tilde{C},\tilde{D}} & = & \tilde{\alpha}; \perp_{\tilde{C}^u,\tilde{D}^u} & (2) \\ \mathbf{K}_{\tilde{C},\tilde{D}} n & = & \tilde{\alpha}; \mathbf{K}_{\tilde{C}^u,\tilde{D}^u} n & (3). \end{array}$$

We can apply Lemma 3.13 to $\breve{\sigma} = \tilde{\alpha}^{-1}; \Lambda_{\tilde{D}}^{-1}(\sigma) : \tilde{C}^u, \tilde{D}^u \Rightarrow N$. The result now follows from equations (1)–(3) since

$$\sigma \approx \Lambda_{\tilde{D}}(\tilde{\alpha}; \breve{\sigma}).$$

□

With the same notations as in the Decomposition Lemma:

Lemma 3.15 (Unicity of Decomposition) *(i) If $\sigma \approx \perp_{\tilde{C},\tilde{D}}$ then $\sigma = \perp_{\tilde{C},\tilde{D}}$.*

(ii) If $\sigma \approx \mathbf{K}_{\tilde{C},\tilde{D}} n$ then $\sigma = \mathbf{K}_{\tilde{C},\tilde{D}} n$.

(iii) If $\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) \underset{\approx}{\sqsubseteq} \mathbf{C}_i(\sigma'_1, \ldots, \sigma'_{l_i}, (\tau'_n \mid n \in \omega))$ then

$$\begin{array}{ll} \sigma_j \underset{\approx}{\sqsubseteq} \sigma'_j, & 1 \le j \le l_i, \\ \tau_n \underset{\approx}{\sqsubseteq} \tau'_n, & n \in \omega. \end{array}$$

Proof *(i)* and *(ii)* are trivial.

For *(iii)* write $\sigma = \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$ and $\tau = \mathbf{C}_i(\sigma'_1, \ldots, \sigma'_{l_i}, (\tau'_n \mid n \in \omega))$.

Suppose firstly that $s \in \tau_n$. Then $*_1 *_2 n_2 s \in \sigma$, so since $\sigma \underset{\approx}{\sqsubseteq} \tau$, for some t, $*_1 *_2 n_2 t \in \tau$ and $*_1 *_2 n_2 s \approx *_1 *_2 n_2 t$. This implies that $t \in \tau'_n$ and $s \approx t$. We conclude that $\tau_n \underset{\approx}{\sqsubseteq} \tau'_n$.

Now suppose that $s \in \sigma_j$. Then $*_1 *_2 s' \in \sigma$ where s' is a reindexed version of s with $s \approx s'$. Since $\sigma \underset{\approx}{\sqsubseteq} \tau$, there exists t' such that $*_1 *_2 t' \in \tau$ and $*_1 *_2 s' \approx *_1 *_2 t'$. This implies that there exists $t \in \sigma'_j$ with $s \approx t$. We conclude that $\sigma_j \underset{\approx}{\sqsubseteq} \sigma'_j$. □

3.5 Approximation Lemmas

The Decomposition Lemma provides for one step of decomposition of an arbitrary strategy into a form matching that of the semantic clauses for evaluation trees. However, infinite strategies will not admit a well-founded inductive decomposition process. Instead, we must appeal to notions of continuity and approximation, in the spirit of Domain Theory [AJ94b].

We define a PCF *type-in-context* ([Cro94]) to be a type of the form

$$(T_1 \& \ldots \& T_p) \Rightarrow U$$

where $T_1, \ldots, T_p, U$ are PCF types. Given such a type-in-context T, we will write $\mathtt{Str}(T)$ for the set of strategies on the game $\mathcal{S}(T)$.

The Unicity of Decomposition Lemma says that decompositions are unique up to partial equivalence. Referring to the Decomposition Lemma, Prop. 3.14, note that the proof of the decomposition

$$\sigma \approx \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$$

involved defining specific strategies $\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)$ from the given σ. If we also fix specific pairing and tagging functions and dereliction indices in the definition of promotion, dereliction, contraction etc.(and hence in the $\mathcal{M}(\mathcal{I})$ operations of composition, pairing, currying etc.), we obtain an operation Φ on strategies such that

$$\Phi(\sigma) = \begin{cases} 1 & \text{in case } \textit{(i)} \\ (2, n) & \text{in case } \textit{(ii)} \\ (3, \sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)) & \text{in case } \textit{(iii)} \end{cases}$$

according to the case of the Decomposition Lemma which applies to σ. We shall use Φ to define a family of functions

$$p_k : \mathtt{Str}(T) \rightarrow \mathtt{Str}(T) \ \ (k \in \omega)$$

inductively as follows:

- $p_0(\sigma) = \Lambda_{\tilde{U}}(\bot_{\tilde{T},\tilde{U}})$
- $$p_{k+1}(\sigma) = \begin{cases} \Lambda_{\tilde{U}}(\bot_{\tilde{T},\tilde{U}}), & \Phi(\sigma) = 1 \\ \Lambda_{\tilde{U}}(\mathbf{K}_{\tilde{T},\tilde{U}} n), & \Phi(\sigma) = (2, n) \\ \Lambda_{\tilde{U}}(\mathbf{C}_i(p_k(\sigma_1), \ldots, p_k(\sigma_{l_i}), (\tau'_n \mid n \in \omega))), & \Phi(\sigma) = \sigma_0 \end{cases}$$

  where
  $$\sigma_0 = (3, \sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$$
  and
  $$\tau'_n = \begin{cases} p_k(\tau_n), & 0 \leq n \leq k \\ \Lambda_{\tilde{U}}(\bot_{\tilde{T},\tilde{U}}), & n > k. \end{cases}$$

The principal properties of these functions are collected in the following Lemma.

Lemma 3.16 (Approximation Lemma for Strategies) *For all $k \in \omega$:*

(i) $\sigma \subseteq \tau$ *implies* $p_k(\sigma) \subseteq p_k(\tau)$

(ii) *If* $\sigma_0 \subseteq \sigma_1 \subseteq \ldots$ *is an increasing sequence,*

$$p_k(\bigcup_{l \in \omega} \sigma_l) = \bigcup_{l \in \omega} p_k(\sigma_l)$$

(iii) $\sigma \lesssim \tau$ *implies* $p_k(\sigma) \lesssim p_k(\tau)$

(iv) $p_k(\sigma) \lesssim \sigma$

(v) $\forall s \in \sigma.\ |s| \leq 2k \Rightarrow \exists t \in p_k(\sigma).\ s \approx t$

(vi) $p_k(\sigma) \subseteq p_{k+1}(\sigma)$

(vii) $\bigcup_{l\in\omega} p_l(\sigma) \approx \sigma$

(viii) $p_k(p_k(\sigma)) \approx p_k(\sigma)$

Proof Firstly, consider the operation $\Phi(\sigma)$. In case *(iii)*, where

$$\Phi(\sigma) = (3, \sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$$

$\Phi(\sigma)$ is obtained by firstly defining σ' and the τ_n from σ, then σ'' from σ', and finally

$$\sigma_j = (\sigma''; \mathtt{der})^\dagger; \pi_j.$$

Note that σ';σ'' and the τ_n are defined *locally*, i.e. by operations on positions applied pointwise to σ and σ' respectively. Together with the $\subseteq$ −monotonicity and continuity of Promotion, Dereliction, Contraction etc. (Proposition 2.9.4) this implies *(i)* and *(ii)*. Now note that $\mathbf{C}_i$ is $\subseteq$ − and $\lesssim_A$ monotonic by Proposition 2.9.3. A straightforward induction using $\lesssim$−monotonicity and $\subseteq$ −monotonicity of $\mathbf{C}_i$ respectively and the Unicity of Decomposition Lemma yelds *(iii)*. Similarly routine inductions using $\lesssim$−monotonicity and $\subseteq$ −monotonicity of $\mathbf{C}_i$ respectively prove *(iv)* and *(vi)*.

We prove *(v)* by induction on k. The basis is trivial as are cases *(i)* and *(ii)* of the Decomposition Lemma at the inductive step. Suppose we are in case *(iii)*, with

$$\sigma \approx \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))$$

Consider firstly $s \in \sigma$ where $s = *_1 *_2 s'$ with $*_2$ not answered in s'. Then $s' \in \sigma''$ where σ'' is derived from σ' and σ' from σ as in the proof of the Decomposition Lemma. Since $\langle \sigma_1, \ldots, \sigma_{l_i}\rangle^\dagger \approx \sigma''$, s' can be decomposed into subsequences $s_{j,1}, \ldots, s_{j,p_j}$ with $s'_{j,q} \approx s_{j,q} \in \sigma_j$, $1 \leq j \leq l_i$, $1 \leq q \leq p_j$.

Since $|s_{j,q}| < |s|$, we can apply the induction hypothesis to conclude that $s_{j,q} \approx u_{j,q} \in p_k(\sigma_j)$, and hence that there is $*_1 *_2 u \in p_{k+1}(\sigma)$ with $s \approx *_1 *_2 u$. The case where $s = *_1 *_2 s' n_2 t$ is similar.

To prove *(vii)*, note firstly that the union $\bigcup_{l\in\omega} p_l(\sigma)$ is well-defined by *(vi)*. Now $\bigcup_{l\in\omega} p_l(\sigma) \lesssim \sigma$ follows from *(iv)*, while $\sigma \lesssim \bigcup_{l\in\omega} p_l(\sigma)$ follows from *(v)*.

Finally *(viii)* can be proved by induction on k and *(iii)* using the Unicity of Decomposition Lemma. □

We now turn to evaluation trees. Let $\Gamma = x_1 : T_1, \ldots, x_k : T_k$. We define a family of functions

$$q_k : \mathbf{ET}(\Gamma, U) \to \mathbf{ET}(\Gamma, U)\ (k \in \omega)$$

inductively by

$$\begin{array}{lcl}
q_0(P) & = & \lambda\tilde{x}^{\tilde{U}}.\Omega \\
q_{k+1}(\lambda\tilde{x}^{\tilde{U}}.\Omega) & = & \lambda\tilde{x}^{\tilde{U}}.\Omega \\
q_{k+1}(\lambda\tilde{x}^{\tilde{U}}.n) & = & \lambda\tilde{x}^{\tilde{U}}.n \\
q_{k+1}(\lambda\tilde{x}^{\tilde{U}}.\mathtt{case}(x_i P_1 \ldots p_{l_i}, (Q_n \mid n \in \omega))) & & \\
= \lambda\tilde{x}^{\tilde{U}}.\mathtt{case}(x_i q_k(P_1) \ldots q_k(P_{l_i}), (Q'_n \mid n \in \omega)) & &
\end{array}$$

where

$$Q'_n = \begin{cases} q_k(Q_n), & 0 \leq n \leq k \\ \lambda\tilde{x}^{\tilde{U}}.\Omega, & n > k \end{cases}$$

The following is then standard:

Lemma 3.17 (Approximation Lemma for Evaluation Trees) *The* $(q_k \mid k \in \omega)$ *form as increasing sequence of continuous functions with* $\bigsqcup_{k\in\omega} q_k = \mathtt{id}_{\mathbf{ET}(\Gamma,U)}$. *Each* q_k *is idempotent and has finite image.*

3.6 Main Results

We are now equipped to address the relationship between strategies and evaluation trees directly. Let $\Gamma = x_1 : T_1, \ldots, x_k : T_k$. We define a map

$$\varsigma : \mathbf{FET}(\Gamma, U) \to \mathtt{Str}(\tilde{T} \Rightarrow U)$$

this map is a concrete version of the semantic map defined in section 2.4. That is, we fix choices of pairing functions etc. as in the definition of Φ in 2.5, and define $\varsigma(\Gamma \vdash P : U)$ as a specific representative of the partial equivalence class $\mathcal{S}(\Gamma \vdash P : U)$. Thus we will have

$$\mathcal{S}(\Gamma \vdash P : U) = [\varsigma(\Gamma \vdash P : U)].$$

We were sloppy about this distinction in 2.4; we give the definition of ς explicitly for emphasis:

$$\begin{array}{lcr}
\varsigma(\Gamma \vdash \lambda x^T.P : T \Rightarrow U) & = & \Lambda(\varsigma(\Gamma, x : T \vdash P : U)) \\
\varsigma(\Gamma \vdash \Omega : N) & = & \bot_{\tilde{T}} \\
\varsigma(\Gamma \vdash n : N) & = & \mathbf{K}_{\tilde{T}} n \\
\varsigma(\Gamma \vdash \mathtt{case}(x_i P_1 \ldots P_{l_i}, (Q_n \mid n \in \omega)) & = & \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))
\end{array}$$

where

$$\begin{array}{rcr}
T_i & = & B_{i,1} \Rightarrow \ldots \Rightarrow B_{i,l_i} \Rightarrow N, \\
\sigma_j & = & \varsigma(\Gamma \vdash P_j : B_{i,j}),\ 1 \le j \le l_i, \\
\tau_n & = & \varsigma(\Gamma \vdash Q_n : N),\ n \in \omega.
\end{array}$$

Lemma 3.18 *If* $P \sqsubseteq Q$ *then* $\varsigma(\Gamma \vdash P : U) \subseteq \varsigma(\Gamma \vdash Q : U)$

Proof By induction on the construction of P, using $\subseteq$–monotonicity of $\mathbf{C}_i$. □

Let $\tilde{T} = T_1, \ldots, T_l$, and $\mathtt{Con}(\tilde{T})$ be the set of all $\tilde{T}$–contexts $x_1 : T_1, \ldots, x_p : T_p$. For each $k \in \omega$, we define a map

$$\eta_k : \mathtt{Str}(\tilde{T} \Rightarrow U) \to \Pi_{\Gamma \in \mathtt{Con}(\tilde{T})} \mathbf{FET}(\Gamma, U)$$

inductively by:

$$\eta_0(\sigma)\Gamma \quad = \quad \lambda \tilde{y}^{\tilde{U}}.\Omega$$

$$\eta_{k+1}(\sigma)\Gamma = \begin{cases} \lambda \tilde{y}^{\tilde{U}}.\Omega, \sigma = \Lambda_{\tilde{U}}(\bot_{\tilde{T},\tilde{U}}) \\ \lambda \tilde{y}^{\tilde{U}}.n, \sigma = \Lambda_{\tilde{U}}(\mathbf{K}_{\tilde{T},\tilde{U}} n) \\ \lambda \tilde{y}^{\tilde{U}}.\mathtt{case}(z_i P_1 \ldots P_{l_i}, (Q_n \mid n \in \omega)), \\ \qquad\qquad \sigma \approx \Lambda_{\tilde{U}}(\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega))) \end{cases}$$

where

$$\begin{array}{rcr}
\Gamma & = & x_1 : T_1, \ldots, x_p : T_p, \\
\Delta & = & y_1 : U_1, \ldots, y_q : U_q \\
\tilde{z} & = & x_1, \ldots, x_p, y_1, \ldots, y_q, \\
P_j & = & \eta_k(\sigma_j)\Gamma, \Delta,\ \ 1 \le j \le l_i
\end{array}$$

and

$$Q_n = \begin{cases} \eta_k(\sigma_j)\Gamma, \Delta, & 0 \le n \le k \\ \Omega & n > k \end{cases}$$

Lemma 3.19 *For all $k \in \omega$:*

(i) $\sigma \underset{\approx}{\sqsubseteq} \tau$ *implies* $\eta_k(\sigma)\Gamma \sqsubseteq \eta_k(\tau)\Gamma$.

(ii) If $\sigma_0 \subseteq \sigma_1 \subseteq \ldots$ *is an increasing sequence,*

$$\eta_k(\bigcup_{l\in\omega} \sigma_l)\Gamma = \bigsqcup_{l\in\omega} \eta_k(\sigma_l)\gamma.$$

(iii) $\eta_k(\sigma)\Gamma \sqsubseteq \eta_{k+1}(\sigma)\Gamma$

(iv) $q_k(\eta_l(\sigma)\Gamma) = \eta_k(\sigma)\gamma, \;\; l \geq k$

Proof *(i)* is proved similarly to part *(iii)* of the Approximation Lemma for strategies; *(ii)* is proved similarly to part *(ii)*; and *(iii)* to part *(vi)*; *(iv)* is proved by a routine induction on k. $\Box$

Lemma 3.20 *For all* $P \in \mathbf{FET}(\Gamma, U)$, $\sigma \in \mathtt{Str}(\tilde{T} \Rightarrow U)$, $k \in \omega$:

(i) $\eta_k(\varsigma(\Gamma \vdash P : U))\Gamma = q_k(P)$

(ii) $\varsigma(\Gamma \vdash (\eta_k(\sigma)\Gamma) : U) \approx p_k(\sigma)$

Proof Both parts are proved by induction on k. The induction bases are trivial as are cases *(i)* and *(ii)* of the Decomposition Lemma at the inductive step, and the corresponding cases on the construction of P

(i)

$$\begin{array}{l} \eta_{k+1}(\varsigma(\Gamma \vdash \lambda\tilde{y}^{\tilde{U}}.\mathtt{case}(z_i P_1 \ldots P_{l_i}, (Q_n \mid n \in \omega)))) \quad = \\ \lambda\tilde{y}^{\tilde{U}}.\mathtt{case}(z_i P'_1 \ldots P'_{l_i}, (Q'_n \mid n \in \omega)) \end{array}$$

where

$$\begin{array}{rcr} P'_j & = & \eta_k(\varsigma(\Gamma, \Delta \vdash P_j : B_{i,j}))\Gamma, \Delta \\ & = \text{ind.hyp} & q_k(P_j) \end{array}$$

$$\begin{array}{rcl} Q'_n & = & \left\{ \begin{array}{cc} \eta_k(\varsigma(\Gamma, \Delta \vdash Q_n : N))\Gamma, \Delta, & 0 \leq n \leq k \\ \Omega & n > k \end{array} \right. \\ & = \text{ind.hyp} & \left\{ \begin{array}{cc} q_k(\Omega) & 0 \leq n \leq k \\ \Omega & n > k \end{array} \right. \end{array}$$

(ii)

$$\begin{array}{l} \varsigma(\Gamma \vdash \eta_{k+1}(\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)))\Gamma : U \quad \approx \\ \Lambda_{\tilde{U}}(\mathbf{C}_i(\sigma'_1, \ldots, \sigma'_{l_i}, (\tau'_n \mid n \in \omega))) \end{array}$$

where

$$\begin{array}{rcl} \sigma'_j & \approx & \varsigma(\Gamma, \Delta \vdash (\eta_k(\sigma_j)\Gamma, \Delta) : U) \\ & \approx_{\text{ind.hyp}} & p_k(\sigma_j) \end{array}$$

$$\begin{array}{rcl} \tau'_n & \approx & \left\{ \begin{array}{cc} \varsigma(\Gamma, \Delta \vdash (\eta_k(\tau_n)\Gamma, \Delta) : N) & 0 \leq n \leq k \\ \perp_{\tilde{T}, \tilde{U}} & n > k \end{array} \right. \\ & \approx_{\text{ind.hyp}} & \left\{ \begin{array}{cc} p_k(\tau_n) & 0 \leq n \leq k \\ \perp_{\tilde{T}, \tilde{U}} & n > k \end{array} \right. \end{array}$$

$\Box$

Now we define functions

$$\mathcal{S} : \mathtt{ET}(\Gamma, U) \to \mathtt{Str}(\tilde{T}, U)$$
$$\mathcal{E} : \mathtt{Str}(\tilde{T}, U) \to \mathtt{ET}(\Gamma, U)$$

by:

$$\mathcal{S}(P) = \bigcup_{k\in\omega} \varsigma(\Gamma \vdash q_k(P) : U)$$

$$\mathcal{E}(\sigma) = \bigsqcup_{k\in\omega} \eta_k(\sigma)\Gamma$$

By Lemma 3.18 and the Approximation Lemma for evaluation trees, $(\varsigma(\Gamma \vdash q_k(P) : U) \mid k \in \omega)$ is an $\subseteq$ −increasing sequence of strategies, so $\mathcal{S}$ is well-defined. Similarly, by Lemma 3.19 $\mathcal{E}$ is well-defined.

We now prove the key result on definability.

Theorem 3.21 (Isomorphism Theorem) *(i) For all $P \in \mathtt{ET}(\Gamma, U)$*

$$\mathcal{E} \circ \mathcal{S}(P) = P$$

(ii) For all $\sigma \in \mathtt{Str}(\tilde{T} \Rightarrow U)$,

$$\mathcal{S} \circ \mathcal{E}(\sigma) \approx \sigma$$

(iii) Let $T = \tilde{T} \Rightarrow U$. Then there is an order-isomorphism

$$\mathcal{S}_{\approx} : \mathtt{ET}(\Gamma, U) \simeq \mathcal{S}(\hat{T}) : \mathcal{E}_{\approx}$$

where $\mathcal{S}_{\approx}(P) = [\mathcal{S}(P)]$ (i.e. the partial equivalence class of $\mathcal{S}(P)$), and $\mathcal{E}_{\approx}([\sigma]) = \mathcal{E}(\sigma)$.

Proof

(i)

$$\begin{array}{lll} \mathcal{E} \circ \mathcal{S}(P) & & \\ & = \text{definition} & \bigsqcup_{k\in\omega} \eta_k(\bigcup_{l\in\omega} \varsigma(\Gamma \vdash q_l(P) : U))\Gamma \\ & = \text{Lemma 3.19}\textit{(ii)} & \bigsqcup_{k\in\omega} \bigsqcup_{l\in\omega} \eta_k(\varsigma(\Gamma \vdash q_l(P) : U))\Gamma \\ & = & \bigsqcup_{n\in\omega} \eta_n(\varsigma(\Gamma \vdash q_n(P) : U))\Gamma \\ & = \text{Lemma 3.20} & \bigsqcup_{n\in\omega} q_n \circ q_n(P) \\ & = \text{Lemma 3.17} & P. \end{array}$$

(ii)

$$\begin{array}{lll} \mathcal{S} \circ \mathcal{E}(\sigma) & & \\ & = & \bigcup_{k\in\omega} \varsigma(\Gamma \vdash q_k(\bigsqcup_{l\in\omega} \eta_l(\sigma)\Gamma) : U) \\ & = \text{continuity of } q_k & \bigcup_{k\in\omega} \varsigma(\Gamma \vdash \bigsqcup_{l\in\omega} q_k(\eta_l(\sigma)\Gamma) : U) \\ & = \text{Lemma 3.19(iv)} & \bigcup_{k\in\omega} \varsigma(\Gamma \vdash (\eta_k(\sigma)\Gamma) : U) \\ & \approx \text{Lemma 3.20} & \bigcup_{k\in\omega} p_k(\sigma) \\ & \approx \text{Lemma 3.16} & \sigma. \end{array}$$

(iii) Firstly $\mathcal{E}_{\approx}$ is well-defined and monotone by Lemma 3.19*(i)*. Also, $\mathcal{S}_{\approx}$ is monotone by Lemma 3.18. By *(i)* and *(ii)*, $\mathcal{E}_{\approx} = \mathcal{S}_{\approx}^{-1}$. □

As an immediate corollary of the Isomorphism Theorem and Proposition 3.2.2:

Proposition 3.22 *For each PCF type T, $\mathcal{S}(\hat{T})$ is a dI-domain. Hence $\mathcal{M}(\mathcal{I})$ is an algebraic cpo-based model.*

Thus although a priori we only knew that $\mathcal{M}(\mathcal{I})$ was a rational model, by virtue of the Isomorphism theorem we know that the carrier at each PCF type is an algebraic cpo. Hence the notion of *intensional full abstraction* makes sense for $\mathcal{M}(\mathcal{I})$. Recall from the introduction that a model is intensionally fully abstract for a language $\mathcal{L}$ if every compact element of the model is denoted by a term of $\mathcal{L}$.

We can now prove the culminating result of this section.

Theorem 3.23 (Intensional Full Abstraction) *$\mathcal{M}(\mathcal{I})$ is intensionally fully abstract for PCFc.*

Proof Consider any PCF type T. By the Isomorphism Theorem, the compact elements of $\mathcal{S}(T)$ are the image under $\mathcal{S}_{\approx}$ of the compact elements of $\mathbf{ET}(\Gamma_0, T)$ (where Γ_0 is the empty context). But the compact elements of $\mathbf{ET}(\Gamma_0, T)$ are just the finite evaluation trees $\mathbf{FET}(\Gamma_0, T)$ and the restriction of $\mathcal{S}_{\approx}$ to $\mathbf{FET}(\Gamma_0, T)$ is the semantic map $\mathcal{S}(.)$ on finite evaluation trees *qua* terms of PCF*c*. □

4 Extensional Full Abstraction

4.1 The Intrinsic Preorder

We define the *Sierpinski game* Σ to be the game

$$\Sigma = (\{q, a\}, \{(q, OQ), (a, PA)\}, \{\epsilon, q, qa\}, \mathtt{id}_{P_\Sigma})$$

with one initial question q, and one possible response a. Note that $\hat{\Sigma}$ is indeed the usual Sierpinski space. i.e. the two-point lattice $\bot < \top$ with $\bot = \{\epsilon\}, \top = \{\epsilon, qa\}$.

Now for any game A we define the *intrinsic preorder* $\lesssim_A$ on $\mathtt{Str}(A)$ by:

$$x \lesssim_A y \iff \forall \alpha : A \to \Sigma.\ x;\alpha \mathrel{\underset{\approx}{\sqsubseteq}} y;\alpha$$

Note that if we write $x{\downarrow} \equiv x = \top$ and $x{\uparrow} \equiv x = \bot$, then:

$$x \lesssim_A y \iff \forall \alpha : A \to \Sigma.\ x;\alpha{\downarrow} \supset y;\alpha{\downarrow}$$

It is trivially verified that $\lesssim_A$ is a preorder.

Lemma 4.1 (Point Decomposition Lemma) *(i)* $\forall x \in \mathtt{Str}(!A).\ x \approx (x;\mathtt{der}_A)^\dagger = !(x;\mathtt{der}_A)$

(ii) $\forall x \in \mathtt{Str}(A\&B).\ x \approx \langle x;\mathtt{fst}, x;\mathtt{snd}\rangle$

(iii) $\forall x \in \mathtt{Str}(A \otimes B).\ \exists y \in \mathtt{Str}(A), z \in \mathtt{Str}(B).\ x \approx y \otimes z$

Proof Firstly we must explain the notation. We think of a strategy σ in A indifferently as having the type $\sigma : I \to A$. Now since $!I = I$, we can regard $!\sigma : !I \to !A$ as in $\mathtt{Str}(!A)$. Similarly, since $I \otimes I = I$, we can regard $\sigma \otimes \tau$ as in $\mathtt{Str}(A \otimes B)$, where $\sigma \in \mathtt{Str}(A), \tau \in \mathtt{Str}(B)$. Finally, using $!I = I$ again we can form $\langle \sigma, \tau\rangle \in \mathtt{Str}(A\&B)$ where $\sigma \in \mathtt{Str}(A), \tau \in \mathtt{Str}(B)$.

Next we note that *(i)* is a special case of the Bang Lemma, while *(ii)* follows from the universal property of the product.

Finally, we prove *(iii)*. Given $x \in \mathtt{Str}(A \otimes B)$, write $x = \sigma_f$, where $f : M_A^O + M_B^O \rightharpoonup M_A^P + M_B^P$. By the switching condition, we can decompose f as $f = g + h$, where $g : M_A^O \rightharpoonup M_A^P$, and $h : M_B^O \rightharpoonup M_B^P$. Now define $y = \sigma_g, z = \sigma_h$. It is clear that y and z are well-defined strategies, and

$$x = \sigma_f = \sigma_{g+h} \approx \sigma_g \otimes \sigma_h = y \otimes z.$$

□

Now we characterise the intrinsic preorder on the Linear types. The general theme is that "intrinsic = pointwise". This is analogous to results in Synthetic Domain Theory and PER models, although the proofs are quite different, and remarkably enough no additional hypotheses are required.

Lemma 4.2 (Extensionality for Tensor) *For all* $x \otimes y, x' \otimes y' \in \mathtt{Str}(A \otimes B)$

$$x \otimes y \lesssim_{A \otimes B} x' \otimes y' \iff x \lesssim_A x' \wedge y \lesssim_B y'$$

Proof $(\Rightarrow)$. If $x \otimes y \lesssim_{A \otimes B} x' \otimes y'$ and $x;\alpha\!\downarrow$, then $x \otimes y; \beta\!\downarrow$ where

$$\beta = A \otimes B \xrightarrow{\mathtt{id}_A \otimes \bot_{B,I}} A \otimes I \xrightarrow{\sim} A \xrightarrow{\alpha} \Sigma,$$

$\bot_{B,I} = \{\epsilon\}$. This implies that $x \otimes y; \beta\!\downarrow$, and hence that $x';\alpha\!\downarrow$. This shows that $x \lesssim_A x'$; the proof that $y \lesssim_B y'$ is similar.

$(\Leftarrow)$. Suppose that $x \lesssim_A x', y \lesssim_B y'$ and $x \otimes y; \gamma\!\downarrow$ where $\gamma : A \otimes B \to \pm$. Then define $\alpha : A \to \Sigma$ by:

$$\alpha = A \xrightarrow{\sim} A \otimes I \xrightarrow{\mathtt{id}_A \otimes y} A \otimes B \xrightarrow{\gamma} \Sigma$$

Then $x;\alpha \approx x \otimes y;\gamma\!\downarrow$, so $x';\alpha \approx x' \otimes y;\gamma\!\downarrow$ since $x \lesssim_A x'$. This shows that $x \otimes y \lesssim_{A \otimes B} x' \otimes y$. A similar argument shows that $x' \otimes y \lesssim_{A \otimes B} x' \otimes y'$, and so

$$x \otimes y \lesssim_{A \otimes B} x' \otimes y \lesssim_{A \otimes B} x' \otimes y'.$$

□

Lemma 4.3 (Extensionality for Product) *For all* $\langle x, y\rangle, \langle x', y'\rangle \in \mathtt{Str}(A \& B)$

$$\langle x, y\rangle \lesssim_{A \& B} \langle x', y'\rangle \iff x \lesssim_A x' \wedge y \lesssim_B y'$$

Proof By the definition of $A\&B$, any $\gamma : A\&B \to \Sigma$ must factor as

$$\gamma = A\&B \xrightarrow{\mathtt{fst}} A \xrightarrow{\alpha} \Sigma$$

or as

$$\gamma = A\&B \xrightarrow{\mathtt{snd}} B \xrightarrow{\beta} \Sigma$$

This shows the right-to-left implication. Conversely, given $\alpha : A \to \Sigma$ we can form

$$A\&B \xrightarrow{\mathtt{fst}} A \xrightarrow{\alpha} \Sigma$$

and similarly for $\beta : B \to \Sigma$. □

Lemma 4.4 (Linear Function Extensionality) *For all* $f, g \in \mathtt{Str}(A \multimap B)$

$$f \lesssim_{A \multimap B} g \iff \forall x \in \mathtt{Str}(A),\ x; f \lesssim_B x; g$$

Proof $(\Rightarrow)$ Suppose $f \lesssim_{A \multimap B} g, x \in \mathtt{Str}(A), \beta : B \to \Sigma$ and $x; f; \beta\!\downarrow$. Then we define $\gamma : (A \multimap B) \to \Sigma$ by

$$\gamma = (A \multimap B) \xrightarrow{\sim} (A \multimap B) \otimes I \xrightarrow{\mathtt{id}_{A \multimap B} \otimes x} (A \multimap B) \otimes A \xrightarrow{\mathtt{LAPP}} B \xrightarrow{\beta} \Sigma$$

For all $h \in \mathtt{Str}(A \multimap B), h;\gamma \approx x;h;\beta$, so $x;g;\beta \approx g;\gamma\!\downarrow$ since $f \lesssim_{A \multimap B} g$ and $f;\gamma\!\downarrow$.

$(\Leftarrow)$ Suppose $f;\gamma\!\downarrow$ where $\gamma : (A \multimap B) \to \Sigma$. From the switching condition we know that γ can respond to the initial move in Σ only in B or Σ; to a move in B only in B or Σ and to a move in A only in A or Σ. Moreover, whenever Player is to move in A the number of moves played in B is odd, hence there is an unanswered question in B which must have been asked more recently than the opening question in Σ. By the stack discipline γ can in fact only respond in A to a move in A. Thus if $\gamma \in \sigma_f$ where $f : M_A^O + M_B^P + M_{\mathcal{O}}^O \rightharpoonup M_A^P + M_B^O + M_\Sigma^P$ we can decompose f as $f = g + h$ where: $g : M_A^O \rightharpoonup M_A^P, h : M_B^P + M_\Sigma^O \rightharpoonup M_B^O + M_\Sigma^P$. If we now define $x = \sigma_g, \beta = \sigma_h$ then:

(i) $x \in \texttt{Str}(A)$.

(ii) $\beta : B \to \Sigma$.

(iii) $\forall h \in \texttt{Str}(A \multimap B). h; \gamma \approx x; h; \beta$.

Now

$$\begin{array}{rll} f; \gamma\downarrow & \supset & x; f; \beta\downarrow \\ & \supset \text{ by assumption} & x; g; \beta\downarrow \\ & \supset & g; \gamma\downarrow \end{array}$$

as required. □

This argument can be understood in terms of Classical Linear Logic. If we think of $A \multimap \Sigma$ as "approximately $A^{\perp}$", then

$$(A \multimap B) \multimap \Sigma \approx (A \multimap B)^{\perp} = A \otimes B^{\perp} \approx A \otimes (B \multimap \Sigma).$$

To prove our final extensionality result, we will need an auxiliary lemma.

Lemma 4.5 (Separation of head occurrence) *For all $\sigma : !A \to \Sigma$, for some $\sigma' : !A \otimes A \to \Sigma$:*

$$\sigma \approx !A \xrightarrow{\texttt{con}_A} !A \otimes !A \xrightarrow{\texttt{id}_{!A} \otimes \texttt{der}_A} !A \otimes A \xrightarrow{\sigma'} \Sigma$$

Proof If $\sigma = \perp_{!A,\Sigma}$ or $\sigma = K_{!A}\top$, the result is trivial. If σ responds to the initial question q with a move (i, a) in $!A$ we define σ' by interpreting the index i as a separate tensorial factor rather than an index in $!A$. The only non-trivial point is to show that $\sigma' \approx \sigma'$. If $q(i,a)sm \approx q(i,a)s'm'$ where $q(i,a)s, q(i,a)s' \in \sigma'$, then any permutation π witnessing the equivalence must satisfy $\pi(i) = i$. Let the response of σ' to m be (j_1, a_1) and to m' (j_2, a_2). Since $\sigma \approx \sigma$ we must have $q(i,a)sm(j_1,a_1) \approx_{!A \multimap \Sigma} q(i,a)s'm'(j_2,a_2)$, and hence either $j_1 = j_2 = i$ or $j_1 \neq j_2 \neq i$. In either cases, $q(i,a)sm(j_1,a_1) \approx_{!A \otimes A \multimap \Sigma} q(i,a)s'm'(j_2,a_2)$, as required. □

Lemma 4.6 (Bang Extensionality) *For all $x, y \in \texttt{Str}(A)$.*

$$x \lesssim_A y \Longleftrightarrow !x \lesssim_{!A} !y$$

Proof ($\Leftarrow$) If $!x \lesssim_{!A} !y$ and $x; \alpha\downarrow$ then $!x; (\texttt{der}_A; \alpha)\downarrow$, so $!y; (\texttt{der}_A; \alpha)\downarrow$, and hence $y; \alpha\downarrow$ as required.

($\Rightarrow$) If $!x; \alpha\downarrow$, define $|\alpha|$ to be the number of indices in $!A$ occurring in $!x \| \alpha$. We show that, for all $\alpha : !A \to \Sigma$ such that $!x; \alpha\downarrow, !y; \alpha\downarrow$, by induction on $|\alpha|$. For the basis, note that $!x; \alpha\downarrow$ and $|\alpha| = 0$ implies that $\alpha = K_{!A}\top$. For the inductive step, let $|\alpha| = k + 1$. By Lemma 4.5, for some $\beta : !A \otimes A \to \Sigma$, $\alpha \approx \texttt{con}_A; \texttt{id}_{!A} \otimes \texttt{der}_A; \beta$. For all $z \in \texttt{Str}(A)$. $!z; \texttt{con}_A; \texttt{id}_{!A} \otimes \texttt{der}_A \approx !z \otimes z$, so $!z \otimes z; \beta \approx !z; \alpha$.

Now define

$$\gamma = !A \xrightarrow{\sim} !A \otimes I \xrightarrow{\texttt{id}_{!A} \otimes x} !A \otimes A \xrightarrow{\beta} \Sigma$$

For all $z \in \texttt{Str}(A)$, $!z; \gamma \approx !z \otimes x; \beta$. In particular, $!x; \gamma \approx !x \otimes x; \beta \approx !x; \alpha\downarrow$. Since $|\alpha| > 0$, there is a first index i_0 in $!A$ used by α. By the definition of γ, $!x \| \gamma$ is $!x \| \alpha$ with all moves at index i_0 deleted. Hence $|\gamma| < |\alpha|$, and by induction hypothesis $!y; \gamma\downarrow$.

Define $\delta : A \to \Sigma$ by

$$\delta = A \xrightarrow{\sim} I \otimes A \xrightarrow{!y \otimes \texttt{id}_A} !A \otimes A \xrightarrow{\beta} \Sigma.$$

Then for all $z \in \texttt{Str}(A)$. $z; \delta \approx !y \otimes z; \beta$. In particular, $x; \delta \approx !y \otimes x; \beta \approx !y; \gamma\downarrow$. By the assumption that $x \lesssim_A y, y; \delta\downarrow$. This implies that $!y; \alpha \approx !y \otimes y; \beta\downarrow$, as required. □

Lemma 4.7 (Intuitionistic Function Extensionality)

$$\sigma \lesssim_{A \Rightarrow B} \tau \Longleftrightarrow \forall x : 1 \Rightarrow A,\ \beta : B \Rightarrow \Sigma.\ \beta \circ \sigma \circ x{\downarrow} \supset \beta \circ \tau \circ x{\downarrow}.$$

Proof

$$\begin{array}{rcl} \sigma \lesssim_{A \Rightarrow B} \tau & \Longleftrightarrow & \forall z \in \mathtt{Str}(!A).z;\sigma \lesssim_B z;\tau \\ & & \text{Linear Function Extensionality} \\ & \Longleftrightarrow & \forall x \in \mathtt{Str}(A).x^\dagger;\sigma \lesssim_B x^\dagger;\tau \\ & & \text{Bang Lemma, } !I = I \\ & \Longleftrightarrow & \forall x \in \mathtt{Str}(A).\ x^\dagger;\sigma^\dagger \lesssim_{!B} x^\dagger;\tau^\dagger \\ & & \text{Bang Extensionality, } \mathtt{der}_I = \mathtt{id}_I \\ & \Longleftrightarrow & \forall x \in \mathtt{Str}(A), \beta :!B \to \Sigma.\ x^\dagger;\sigma^\dagger;\beta{\downarrow} \supset x^\dagger;\tau^\dagger;\beta{\downarrow} \\ & \Longleftrightarrow & \forall x : 1 \Rightarrow A, \beta : B \Rightarrow \Sigma.\ \beta \circ \sigma \circ x{\downarrow} \supset \beta \circ \tau \circ x{\downarrow}. \end{array}$$

□

Lemma 4.8 (Congruence Lemma) *(i)* $\sigma \lesssim_{A \Rightarrow B} \sigma' \wedge \tau \lesssim_{B \Rightarrow C} \tau' \supset \tau \circ \sigma \lesssim_{A \Rightarrow C} \tau' \circ \sigma'$

(ii) $\sigma \lesssim_{C \Rightarrow A} \sigma' \wedge \tau \lesssim_{C \Rightarrow B} \tau' \supset \langle \sigma, \tau \rangle \lesssim_{C \Rightarrow A\&B} \langle \sigma', \tau' \rangle$.

(iii) $\sigma \lesssim_{A\&B \Rightarrow C} \tau \supset \Lambda(\sigma) \lesssim_{A \Rightarrow (B \Rightarrow C)} \Lambda(\tau)$.

Proof *(i)*

$$\begin{array}{rcll} \beta \circ \tau \circ \sigma \circ x{\downarrow} & \supset & \beta \circ \tau' \circ \sigma \circ x{\downarrow} & \tau \lesssim_{B \Rightarrow C} \tau' \\ & \supset & \beta \circ \tau' \circ \sigma' \circ x{\downarrow} & \sigma \lesssim_{A \Rightarrow B} \sigma'. \end{array}$$

(ii) For all $x : 1 \Rightarrow C$, $\langle \sigma, \tau \rangle \circ x \approx \langle \sigma \circ x, \tau \circ x \rangle : I \to A\&B$; and similarly, $\langle \sigma', \tau' \rangle \circ x \approx \langle \sigma' \circ x, \tau' \circ x \rangle$. By *(i)*, $\sigma \circ x \lesssim_A \sigma' \circ x$ and $\tau \circ x \lesssim_B \tau' \circ x$. The result now follows by Product Extensionality.

(iii) Identifying morphisms with points of arrow types,

$$\begin{array}{rcll} \gamma \circ \Lambda(\sigma) \circ x \circ y{\downarrow} & \supset & \gamma \circ \sigma \circ \langle x, y \rangle{\downarrow} & \\ & \supset & \gamma \circ \tau \circ \langle x, y \rangle{\downarrow} & \sigma \lesssim_{A\&B \Rightarrow C} \tau \\ & \supset & \gamma \circ \Lambda(\tau) \circ x \circ y{\downarrow}. & \end{array}$$

□

Finally we consider the relationship between the intrinsic and intensional preorders.

Lemma 4.9 *(i) If $\sigma \sqsubseteq\!\!\approx_A \tau$, then $\sigma \lesssim_A \tau$.*

(ii) If $\sigma_o \subseteq \sigma_1 \subseteq \ldots$ is an increasing sequence, and for all n, $\sigma_n \lesssim_A \tau_n$, then $\bigcup_{n \in \omega} \sigma_n \lesssim_A \tau_n$.

Proof *(i)* By $\sqsubseteq\!\!\approx$−monotonicity of composition (Proposition 2.9.3) if $\sigma \sqsubseteq\!\!\approx_A \tau$ and $\sigma;\alpha = \top$ then $\top = \sigma;\alpha \sqsubseteq\!\!\approx_A \tau;\alpha$ and hence $\tau;\alpha = \top$.

(ii) By $\subseteq$ −continuity of composition (Proposition 2.9.3), similarly to *(i)*. □

By Lemma 4.9, $\sigma \approx \tau$ implies $\sigma \simeq \tau$ where $\simeq$ is the equivalence induced by the preorder $\lesssim$. Thus each $\simeq$ −equivalence class is a union of $\approx$ −classes. Henceforth, when we write $[\sigma]$ we shall mean the $\simeq$ −equivalence class of σ.

We can define the notion of *strong chain* of $\simeq$ −equivalence classes, just as we did for $\approx$ −classes: a sequence

$$(\dagger) \quad [\sigma_0] \lesssim [\sigma_1] \lesssim \ldots$$

such that there are $(\sigma'_n \mid n \in \omega)$ with $\sigma'_n \in [\sigma_n]$ and $\sigma'_n \subseteq \sigma'_{n+1}$ for all $n \in \omega$.

Lemma 4.10 *Every strong $\lesssim$ −chain has a $\lesssim$ −least upper bound.*

Proof Given a strong chain (†), take $\bigsqcup_{n \in \omega}[\sigma_n] = [\sigma']$ where $\sigma' = \bigcup_{n \in \omega} \sigma'_n$. For all n, $\sigma_n \simeq \sigma'_n \subseteq \sigma'$, so by Lemma 4.9*(i)*, $[\sigma']$ is un upper bound for $([\sigma_n] \mid n \in \omega)$.

Finally, if $[\tau]$ is another upper bound, then for all n, $\sigma'_n \lesssim \tau$; so by Lemma 4.9*(ii)*, $\sigma' \lesssim \tau$. □

4.2 The Extensional Category

We begin with some general considerations on quotients of rational cartesian closed categories. Let $\mathcal{C}$ be a rational CCC. A *precongruence* on $\mathcal{C}$ is a family $\lesssim = \{\lesssim_{A,B} \mid A, B \in \mathtt{Obj}(\mathcal{C})\}$ of relations $\lesssim_{A,B} \subseteq \mathcal{C}(A,B) \times \mathcal{C}(A,B)$ satisfying the following properties:

(r1) each $\lesssim_{A,B}$ is a preorder

(r2) $f \lesssim_{A,B} f'$ and $g \lesssim_{B,C} g'$ implies $g \circ f \lesssim_{A,C} g' \circ f'$

(r3) $f \lesssim_{C,A} f'$ and $g \lesssim_{C,B} g'$ implies $\langle f, g \rangle \lesssim_{C,A\times B} \langle f', g' \rangle$

(r4) $f \lesssim_{A\times B,C} g$ implies $\Lambda(f) \lesssim_{A,B\Rightarrow C} \Lambda(g)$

(r5) $\sqsubseteq_{A,B} \subseteq \lesssim_{A,B}$

(r6) for all $f : A \times B \to B, g : C \to A, h : B \to D$:

$$(\forall n \in \omega.\ h \circ f^{(n)} \circ g \lesssim_{C,D} k) \supset h \circ f^{\nabla} \circ g \lesssim_{C,D} k.$$

Given such a precongruence, we define a new category $\mathcal{C}/{\lesssim}$ as follows. The objects are the same as those of $\mathcal{C}$;

$$\mathcal{C}/{\lesssim}(A,B) = (\mathcal{C}(A,B)/\simeq_{A,B}, \leq_{A,B}).$$

That is, a morphism in $C/\lesssim_{(A,B)}$ is a $\simeq_{A,B}$ −equivalence class $[f]$, where $\simeq_{A,B}$ is the equivalence relation induced by $\lesssim_{A,B}$. The partial ordering is then the induced one:

$$[f] \leq_{A,B} [g] \iff f \lesssim_{A,B} g.$$

Note that by (r5), $[\bot_{A,B}]$ is the least element with respect to this partial order. By (r2)–(r4), composition, pairing and currying are well-defined on $\simeq$ −equivalence classes by

$$\begin{array}{lcl} [g] \circ [f] & = & [g \circ f], \\ \langle [f], [g] \rangle & = & [\langle f, g \rangle], \\ \Lambda([f]) & = & [\Lambda(f)]\ . \end{array}$$

It is then immediate by (r5) and the fact that $\mathcal{C}$ is a rational (and hence in particular a ppo-enriched) CCC that $\mathcal{C}/{\lesssim}$ is a ppo-enriched CCC. It remains to verify rationality for $\mathcal{C}/{\lesssim}$. By (r2) and (r5), for any $f : A \times B \to B, g : C \to A, h : B \to D$, the sequence $([h \circ f^{(n)} \circ g] \mid n \in \omega)$ is a $\leq_{C,D}$-chain. By (r5) and (r6), $[h \circ f^{\nabla} \circ g]$ is the $\leq_{C,D}$ −least upper bound of this chain. In particular , taking $g = \mathtt{id}_A$ and $h = \mathtt{id}_B$, $[f^{\nabla}]$ is the least upper bound of $([f^{(n)}] \mid n \in \omega)$.

We record this result, which is a variant of [ADJ76], as

Lemma 4.11 (Rational Quotient) *If $\lesssim$ is a precongruence on a rational CCC $\mathcal{C}$, then $\mathcal{C}/{\lesssim}$ is a rational CCC.*

Now we define a family $\lesssim = \{\lesssim_{A\Rightarrow B} \mid A, B \in \mathtt{Obj}(K_!(\mathcal{G}))\}$.

Lemma 4.12 *$\lesssim$ is a precongruence on $K_!(\mathcal{G})$.*

Proof The fact that $\lesssim_{A\to B}$ is a preorder has already been noted. (r2)–(r4) are the pre-congruence Lemma 4.1.8. (r5) is Lemma 4.1.9*(i)*. Finally, we verify (r6). Let $\sigma : A\&B \Rightarrow B, \tau : C \Rightarrow A, \theta : B \Rightarrow D$. As already explained, since $\underset{\approx}{\sqsubseteq} \subseteq \lesssim$, we work directly with $\simeq$ −classes of strategies, rather that $\simeq$ −classes of $\approx$ −classes of strategies. Now $(\theta \circ \sigma^{(n)} \circ \tau \mid n \in \omega)$ is a $\subseteq$ −chain (using $\subseteq$ −monotonicity of composition), and we can apply Lemma 4.1.9*(ii)* to yeld (r6). $\square$

Now we define $\mathcal{E} = K_!(\mathcal{G})/\lesssim$.

Proposition 4.13 *$\mathcal{E}$ is a rational CCC. Moreover, $\mathcal{E}$ is well-pointed in the order-enriched sense:*

$$f \leq_{A,b} g \Leftrightarrow \forall x : 1 \to A.\ \ f \circ x \leq_{A,B} g \circ x\ .$$

Proof $\mathcal{E}$ is a rational CCC by Lemma 4.11 and 4.12. It is well-pointed by Intuitionistic Function Extensionality (Lemma 4.1.7). □

Now we define the PCF model $\mathcal{M}(\mathcal{E})$ with the same interpretation of **Nat** as in $\mathcal{M}(K_!(\mathcal{G}))$. The ground and first-order constants of PCF are interpreted by the $\simeq$ $-$equivalence classes of their interpretations in $\mathcal{M}(K_!(\mathcal{G}))$.

Proposition 4.14 *$\mathcal{M}(\mathcal{E})$ is an order-extensional standard model of PCF.*

Proof $\mathcal{M}(\mathcal{E})$ is an order-extensional model of PCF by Proposition 4.2.3. It is standard because $\mathcal{M}(K_!(\mathcal{G}))$ is, and $\lesssim_{\mathbf{Nat}}=\sqsubseteq_{\mathbf{Nat}}$. □

4.3 An alternative view of $\mathcal{E}$

We now briefly sketch another way of looking at $\mathcal{E}$, which brings out its extensional character more clearly, although technically it is no more than a presentational variant of the above description. Given a game A, define

$$\mathcal{D}(A) = (\{[x]_{\simeq} \mid x \in \mathtt{Str}(A)\}, \leq_A)$$

Then $\mathcal{D}(A)$ is a pointed poset. Given $\sigma : A \Rightarrow B$, define $\mathcal{D}(\sigma) : \mathcal{D}(A) \to \mathcal{D}(B)$ as the (monotone) function defined by:

$$\mathcal{D}(\sigma)([x]) = [\sigma \circ x]$$

Write $f : A \to_{\mathcal{E}} B$ if $f : \mathcal{D}(A) \to \mathcal{D}(B)$ is a monotone function such that $f = \mathcal{D}(\sigma)$ for some $\sigma : A \Rightarrow B$. In this case we say that f is *sequentially realised* by σ, and write $\sigma \Vdash f$.

Note that there are order-isomorphisms

- $\mathcal{D}(I) \cong 1$
- $\mathcal{D}(A\&B) \cong \mathcal{D}(A) \times \mathcal{D}(B)$
- $\mathcal{D}(A \Rightarrow B) \cong \mathcal{D}(A) \Rightarrow_{\mathcal{E}} \mathcal{D}(B)$

Here $\mathcal{D}(A) \times \mathcal{D}(B)$ is the cartesian product of the posets $\mathcal{D}(A), \mathcal{D}(B)$, with the pointwise order; while $\mathcal{D}(A) \Rightarrow_{\mathcal{E}} \mathcal{D}(B)$ is the set of all functions $f : A \to_{\mathcal{E}} B$, again with the pointwise order.

Now note that, with respect to the representations of $\mathcal{D}(A\&B)$ as a cartesian product and $\mathcal{D}(A \Rightarrow B)$ as a "function space", the interpretations of composition, pairing and projections, and currying and application in $\mathcal{E}$ are the usual set-theoretic operations on functions *in extenso*. That is,

$$\begin{array}{lcl}
\mathcal{D}(\tau \circ \sigma) & = & \mathcal{D}(\tau) \circ \mathcal{D}(\sigma) \\
\mathcal{D}(\langle \tau, \sigma \rangle) & = & \langle \mathcal{D}(\sigma), \mathcal{D}(\tau) \rangle \\
\mathcal{D}(\pi_1) & = & \pi_1 \\
\mathcal{D}(\pi_2) & = & \pi_2 \\
\mathcal{D}(\Lambda(\sigma)) & = & \Lambda(\mathcal{D}(\sigma)) \\
\mathcal{D}(\mathtt{Ap}) & = & \mathtt{Ap}
\end{array}$$

where the operations on the right hand sides are defined as in the category of sets (or any concrete category of domains).

Thus an equivalent definition of $\mathcal{E}$ is as follows:

`Objects`	as in $K_!(\mathcal{G})$
`Arrows`	$f : A \to_{\mathcal{E}} B$
`Composition`	function composition

The rôle of the intensional structure, that is of the use of the game A to represent the abstract space $\mathcal{D}(A)$, is to cut down the function spaces to the sequentially realisable functions. Specifically, note the use of A and B in the definition of $\mathcal{D}(A) \Rightarrow_{\mathcal{E}} \mathcal{D}(B)$.

4.4 Full Abstraction

We recall that a model $\mathcal{M}$ is fully abstract for a language $\mathcal{L}$ if, for all types T and closed terms $M, N : T$

$$\mathcal{M}[\![M]\!] \sqsubseteq \mathcal{M}[\![N]\!] \Leftrightarrow M \sqsubseteq_{\mathtt{obs}} N \quad (\dagger)$$

where

$$\begin{array}{lll} M \sqsubseteq_{\mathtt{obs}} N & \Leftrightarrow & \forall \text{ program context } C[.] \\ & & C[M]\Downarrow n \supset C[N]\Downarrow n \end{array}$$

Here a program context $C[.]$ is one for which $C[P]$ is a closed term of type N for any closed term $P : T$; and $\Downarrow$ is the operational convergence relation. The left—to—right implication in $(\dagger)$ is known as *soundness* and the converse as *completeness*. It is standard that soundness is a consequence of computational adequacy [Cur93]; thus by Proposition 2.10.1, standard models are sound. Also, full abstraction for closed terms is easily seen to imply the corresponding statement $(\dagger)$ for open terms.

Theorem 4.15 *$\mathcal{M}(\mathcal{E})$ is fully abstract for PCF.*

Proof Firstly, $\mathcal{M}(\mathcal{E})$ is a standard model by Proposition 4.14, and hence sound. We shall prove the contrapositive of completeness. Suppose M, N are closed terms of PCF of type $T = T_1 \Rightarrow \ldots T_k \Rightarrow \mathbf{Nat}$ and

$$\mathcal{M}(\mathcal{E})[\![M]\!] \not\lesssim_{[\![T]\!]} \mathcal{M}(\mathcal{E})[\![N]\!].$$

Let $[\sigma] = \mathcal{M}(\mathcal{E})[\![M]\!], [\tau] = \mathcal{M}(\mathcal{E})[\![N]\!]$. By Intuitionistic Function Extensionality, for some $x_1 \in \mathtt{Str}([\![T_1]\!]), \ldots, x_k \in \mathtt{Str}([\![T_k]\!])$,

$$\beta : !N \to \Sigma, \beta \circ \sigma \circ x_1 \circ \ldots \circ x_k \downarrow \text{ and } \beta \circ \tau \circ x_1 \circ \ldots \circ x_k \uparrow.$$

By $\underset{\approx}{\sqsubset}$−monotonicity of composition, this implies that $\sigma \circ x_1 \circ \ldots \circ x_k \not\underset{\approx}{\sqsubset}_{\mathbf{Nat}} \tau \circ x_1 \circ \ldots \circ x_k$, and hence that $\sigma \circ x_1 \circ \ldots \circ x_k = n$ for some $n \in \omega$, and $\tau \circ x_1 \circ \ldots \circ x_k \neq n$. By $\subseteq$ −continuity of composition and the properties of the projections p_k given in the Approximation Lemma 3.5.1, for some $m \in \omega$, $\sigma \circ p_m(x_1) \circ \ldots \circ p_m(x_k) = n$, while by $\subseteq$ −monotonicity of composition, $\tau \circ p_m(x_1) \circ \ldots \circ p_m(x_k) \neq n$. By Lemma 3.6.3, there are finite evaluation trees,and hence PCFc terms $P_1, \ldots, P_k$ such that $[\![P_i]\!] = [p_m(x_i)]$, $1 \leq i \leq k$. This means that $[\![MP_1 \ldots P_k]\!] = n$, while $[\![NP_1 \ldots P_k]\!] \neq n$. By computational adequacy, this implies that $MP_1 \ldots P_k \Downarrow n$ and $\neg(NP_1 \ldots P_k \Downarrow n)$. By Lemma 3.1.1, each PCFc term is observationally congruent to a PCF term. Hence there is a PCF context $C[.] = [.]Q_1 \ldots Q_k$, where $Q_i \cong_{\mathtt{obs}} P_i$, $1 \leq i \leq k$, such that $C[M]\Downarrow n$ and $\neg(C[N]\Downarrow n)$. This implies that $M \not\sqsubseteq_{\mathtt{obs}} N$, as required. □

As an instructive consequence of this proof, we have:

Corollary 4.16 (Context Lemma) *For all closed $M, N : T_1 \Rightarrow \ldots T_k \Rightarrow$ **Nat**,*

$$\begin{array}{rcl} M \sqsubseteq_{\mathtt{obs}} N & \Leftrightarrow & \forall \textit{ closed } P_1 : T_1, \ldots, P_k : T_k \\ & & MP_1 \ldots P_k \Downarrow n \supset NP_1 \ldots P_k \Downarrow n \end{array}$$

Proof The left-to-right implication is obvious, by considering applicative contexts $[.]P_1 \ldots P_k$. The converse follows from the proof of the Full Abstraction Theorem, since if $M \not\sqsubseteq_{\mathtt{obs}} N$, then $[\![M]\!] \not\leq [\![N]\!]$ by soundness, and then by the argument for completeness this can be translated back into an applicative context separating M and N. $\square$

The point of reproving this well-known result is that a semantic proof falls out of the Full Abstraction Theorem. By contrast, Milner had to prove the Context Lemma directly, as a necessary preliminary to his syntactic construction of the fully abstract model. Moreover, the direct syntactic proof, particularly for the $\lambda-$calculus formulation of PCF [Cur93], is quite subtle. This gives some immediate evidence of substance in our "semantic analysis".

5 Universality

The definability result we have achieved so far refers only to compact strategies. Our aim in this section is to characterize precisely which strategies are (extensionally) definable in PCF, and in fact to construct a fully abstract model in which *all* strategies are definable.

5.1 Recursive Strategies

We shall develop effective versions of $\mathcal{G}$ and $\mathcal{E}$. Our treatment will be very sketchy, as the details are lengthy and tedious, but quite routine. We refer to standard texts such as [Soa87] for background.

We say that a game A is *effectively given* if there is a surjective map $e_A : \omega \to M_A$ with respect to which λ_A (with some coding of $\{P, O, Q, A\}$) and the characteristic functions of P_A and $\approx_A$ (with some coding of finite sequences) are tracked by recursive functions. A strategy σ on A is then said to be *recursive* if σ is a recursively enumerable subset of P_A (strictly speaking, if the set of codes of positions in σ is r.e.).

Lemma 5.1 *$\sigma = \sigma_f$ is recursive iff f is tracked by a partial recursive function. There are recursive functions taking an index for σ to one for f, and vice versa.*

Proof The predicate $f(a) \simeq b \Leftrightarrow \exists s.sab \in \sigma$ is clearly r.e. in σ, hence f has an r.e. graph and is partial recursive

Conversely, given f define a predicate $G(s, n)$ by:

$$\begin{array}{lcl} G(s, 0) & = & s = \epsilon, \\ G(s, n+1) & = & \exists a, b, t.\ s = tab \wedge s \in P_A \wedge G(t, n) \wedge f(a) \simeq b. \end{array}$$

Clearly G is r.e. and hence so is

$$\sigma = \mathtt{graph}(f) = \{s \mid \exists n.G(s, n)\}.$$

These constructions are defined via simple syntactic transformations and yield effective operations on indices. $\square$

If A and B are effectively given, one can verify that the effective structure lifts to $A \otimes B$, $A \multimap B, A \& B$ and $!A$. Also, I and **Nat** are evidently effectively given. The most interesting point which arises in verifying these assertions is that $\approx_{!A}$ is recursive. This requires the observation that, in checking $s \approx_{!A} t$, it suffices to consider permutations $\pi \in S(\omega)$ of *bounded* (finite) support, where the bound is easily computed from s and t.

Similarly, one can check that all operations on strategies defined in Section 2 effectivize. For example, it is easily seen that the definition of $\sigma; \tau$ in terms of sets of positions is r.e. in σ and τ; or, we can give an algorithm for computing $EX(f, g)$. This algorithm simply consists of applying f and g alternately starting from whichever is applicable to the input, until an "externally visible" output appears. Note that it is *not* the case in general that unions of $\subseteq$ $-$chains of recursive strategies are recursive. For example every strategy of type $N \multimap N$ is a union of an increasing chain of finite and hence recursive strategies. However, given a recursive $\sigma : A \& B \Rightarrow B$, $\sigma^{\nabla} = \bigcup_{n \in \omega} \sigma^{(n)}$ is recursive, since it can be enumerated uniformly effectively in n ("r.e. unions of r.e. sets are r.e.").

Thus we can define a category $\mathcal{G}_{\mathtt{rec}}$ with objects effectively given games, and morphisms (partial equivalence classes of) recursive strategies. Also, the interpretations of PCF constants in $\mathcal{M}(K_!(\mathcal{G}))$ are clearly recursive strategies.

Proposition 5.2 *(i)* $\mathcal{G}_{\mathtt{rec}}$ *is a Linear category*

(ii) $K_!(\mathcal{G}_{\mathtt{rec}})$ *is a rational cartesian closed category*

(iii) $\mathcal{M}(K_!(\mathcal{G}_{\mathtt{rec}}))$ *is a standard model of PCF*

We can now consider the extensional quotient $\mathcal{E}_{\mathtt{rec}} = K_!(\mathcal{G}_{\mathtt{rec}})/\lesssim$ where $\lesssim$ is defined just as for $K_!(\mathcal{G})$, but of course with respect to recursive tests, i.e. recursive strategies $A \multimap \Sigma$. All the results of section 4 go through with respect to recursive tests.

Proposition 5.3 $\mathcal{E}_{\mathtt{rec}}$ *is a well-pointed rational CCC.* $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ *is a fully abstract model of PCF.*

Proof The result does require a little care, since the Isomorphism Theorem 3.6.4 is not valid for $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$. However, the Isomorphism Theorem was not used in the proof of the Full Abstraction Theorem 4.3.1, but rather the finitary version Lemma 3.6.3, which is valid in $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$. $\square$

It is worth remarking that a nice feature of our definition of model in terms of rationality rather than cpo-enrichment is that the recursive version $\mathcal{E}_{\mathtt{rec}}$ is again a model in exactly the same sense as $\mathcal{E}$. By contrast, in the cpo-enriched setting one must either modify the definition of model explicitly (by only requiring completeness with respect to r.e. chains), or implicitly by working inside some recursive realizability universe.

5.2 Universal Terms

The fact that $\mathcal{M}(K_!(\mathcal{G}_{\mathtt{rec}}))$ and $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ are models shows that all PCF terms denote recursive strategies, as we would expect. Our aim now is to prove a converse; every recursive strategy is, up to extensional equivalence, the denotation of a PCF term, and hence every functional in the extensional model $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ is definable in PCF.

More precisely our aim is to define, for each PCF type T, a "universal term" U_T : **Nat** $\Rightarrow T$, such that

$$\mathcal{E}[\![U_T \ulcorner \sigma \urcorner]\!] = [\sigma]$$

for each recursive σ. These universal terms will work by simulating the evaluation tree corresponding to σ.

Firstly, we recall some notations from recursion theory. We fix an acceptable numbering of the partial recursive functions [Soa87] such that ϕ_n is the n'th partial recursive function and W_n is the n'th r.e. set. We also fix a recursive pairing function $\langle -, - \rangle : \omega \times \omega \to \omega$ and a recursive coding of finite sequences.

A recursive strategy σ is regarded as being given by a code (natural number) $\lceil \sigma \rceil$. By virtue of Lemma 5.1.1 we use such a code indifferently as determining σ by

$$\sigma = \sigma_f \text{ where } f = \phi_{\lceil \sigma \rceil}$$

or

$$W_{\lceil \sigma \rceil} = \{\lceil s \rceil \mid s \in \sigma\}$$

The following lemma is a recursive refinement of the Decomposition Lemma, and assumes the notations of Section 3.4.

Lemma 5.4 (Decomposition Lemma (Recursive Version)) *For each PCF type T there are partial recursive functions*

$$D_T, H_T : \omega \rightharpoonup \omega \textit{ and } B_T : \omega \times \omega \rightharpoonup \omega$$

such that, if σ is a recursive strategy on T

$$D_T \lceil \sigma \rceil = \begin{cases} \textit{undefined}, & \sigma = \bot_{\tilde{T}} \\ \langle 2, n \rangle, & \sigma = K_{\tilde{T}} n \\ \langle 3, i \rangle, & R(\sigma) \end{cases}$$

$$H_T \lceil \sigma \rceil = \begin{cases} \langle \lceil \sigma_1 \rceil, \ldots, \lceil \sigma_{l_i} \rceil \rangle, & R(\sigma) \\ \textit{undefined}, & \textit{otherwise} \end{cases}$$

$$B_T(\lceil \sigma \rceil, n) = \begin{cases} \lceil \tau_n \rceil, & R(\sigma) \\ \textit{undefined}, & \textit{otherwise} \end{cases}$$

where $R(\sigma)$ stands for

$$\Phi(\sigma) = (3, i, \sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)).$$

Proof $\mathcal{D}_T \lceil \sigma \rceil$ is computed by applying $\phi_{\lceil \sigma \rceil}$ to the (code of) the initial question. The extraction of τ_n from σ, $\tau_n = \{*_1 s \mid *_1 *_2 ns \in \sigma\}$, is obviously r.e. in σ, uniformly effectively in n. Hence we obtain an r.e. predicate $s \in B_T(\lceil \sigma \rceil, n)$, and by an application of the S-m-n theorem we obtain the index for "$B_T \lceil \sigma \rceil n = \lceil \tau_n \rceil$".

Similarly the extraction of σ' from σ is r.e. in σ, and that of σ'' for σ' is r.e. in σ'; while $\sigma_1, \ldots, \sigma_{l_i}$ are obtained from σ'' by composition, dereliction and projection, which are computable operations by Proposition 5.1.2. Hence applying the S-m-n theorem again we obtain the codes for $\sigma_1, \ldots, \sigma_{l_i}$. $\Box$

Given a PCF type T, we define the subtypes of T to be the PCF types occurring as subformulas of T, e.g. $(N \Rightarrow N)$ and N are subtypes of $(N \Rightarrow N) \Rightarrow N$. Let $S_1, \ldots, S_q$ be a listing of all the (finitely many) subtypes of T, where we write

$$S_i = S_{i,1} \Rightarrow \ldots S_{i,l_i} \Rightarrow N$$

To aid the presentation, we will use an abstract datatype $\texttt{Ctxt}_T$ of "T-contexts", which we will show later how to implement in PCF. We will make essential use of the fact that, while contexts can grow to arbitrary size in the recursive unfolding of an evaluation tree of type T, the types occurring in the context can only be subtypes of T.

$\texttt{Ctxt}_T$ comes with the following operations:

- $\texttt{emptycontext}_T : \texttt{Ctxt}_T$

- $\mathtt{get}_S : N \Rightarrow \mathtt{Ctxt}_T \Rightarrow S$ for each subtype S of T
- $\mathtt{extend}_{S_i} : \mathtt{Ctxt}_T \Rightarrow S_{i,1} \Rightarrow \ldots S_{i,l_i} \Rightarrow \mathtt{Ctxt}_T$ for each subtype S_i of T
- $\mathtt{map}_T : N \Rightarrow \mathtt{Ctxt}_T \Rightarrow N$.

If $\Gamma = x_1 : U_1, \ldots, x_n : U_n$, then Γ^i is the subsequence of all entries of type S_i, $1 \leq i \leq q$ and $\Gamma_j = x_j : U_j$

The idea is that, if Γ is an "abstract context",

- $\mathtt{extend}_{S_i}\Gamma x_1^{S_{i,1}} \ldots x_{l_i}^{S_{i,l_i}} = \Gamma, x_1 : S_{i,1}, \ldots, x_{l_i} : S_{i,l_i}$
- $\mathtt{map}_T\ i\ \Gamma = \langle i_1, i_2 \rangle$ where $\Gamma_i = x : S_{i_1} = \Gamma^{i_1}_{i_2}$
- $\mathtt{get}_{S_i} j \Gamma = \Gamma^i_j$.

Now we use the standard fact that every partial recursive function $\phi : \omega \rightharpoonup \omega$ can be represented by a closed PCF term $M : N \Rightarrow N$ in the sense that, for all $n \in \omega$

$$Mn \Downarrow m \Leftrightarrow \phi n \simeq m.$$

This obviously implies that partial recursive functions of two arguments can be represented by closed terms of type $N \Rightarrow N \Rightarrow N$. Specifically, we fix terms $\mathbf{D_T}, \mathbf{H_T} : N \Rightarrow N$ and $\mathbf{B_T} : N \Rightarrow N \Rightarrow N$ which represent D_T, H_T and B_T respectively.

Now we define a family of functions

$$F_S : \mathtt{Ctxt}_T \Rightarrow N \Rightarrow S$$

for each subtype $S = U_1 \Rightarrow \ldots U_k \Rightarrow N$ of T, by the following mutual recursion:

```
$F_S = \lambda k^N.\lambda\Gamma^{\mathrm{Ctxt}}_T.\lambda x_1^{U_1} \ldots \lambda x_k^{U_k}$
 let$\langle k_1, k_2\rangle = D_T k$ in
  if $k_1 = 2$ then $k_2$ else
    if $k_1 = 3$ then
     let $\Delta = \mathtt{extend}_S\ \Gamma x_1 \ldots x_k$ in
      let $\langle i_1, i_2\rangle = \mathtt{map}_T\ k_2\ \Delta$ in
       case $i_1$ of
        $1 : \ldots$
        $\vdots$
        $i :$ let $\langle k_1, \ldots, k_{l_i}\rangle = H_S k$ in
             let $n = (\mathtt{get}_{S_i} i_2\Delta)(F_{S_{i,1}} k_1 \Delta) \ldots (F_{S_{i,l_i}} k_{l_i} \Delta)$
                    in $F_N(B_S kn)\Delta$
        $i+1 : \ldots$
        $\vdots$
        $q : \ldots$
      otherwise : $\Omega$
     endcase
    else $\Omega$
```

These functions have been defined using some "syntactic sugar". Standard techniques can be used to transform these definitions into PCF syntax. In particular Bekič's rule [Win93] can be used to transform a finite system of simultaneous recursion equations into iterated applications of the $\mathbf{Y}$ combinator. The universal term U_T can then be defined by

$$U_T = F_T\ \mathtt{emptycontext}_T.$$

It remains to be shown how $\mathtt{Ctxt}_T$ can be implemented in PCF. To do this, we assume two lower-level data-type abstractions, namely product types $T \times U$ with pairing and projections, and list types $\mathtt{list}(T)$ for each PCF type T, with the usual operations:

- $\mathtt{empty}_T : \mathtt{list}(T) \Rightarrow N$
- $\mathtt{cons}_T : T \Rightarrow \mathtt{list}(T) \Rightarrow \mathtt{list}(T)$
- $\mathtt{hd}_T : \mathtt{list}(T) \Rightarrow T$
- $\mathtt{tl}_T : \mathtt{list}(T) \Rightarrow \mathtt{list}(T)$
- $\mathtt{nil}_T : \mathtt{list}_T$

We write $l{\downarrow}i$ for the i'th component of a list.

We represent an abstract context Γ by the $q+1-$tuple $(l_1, \ldots, l_q, \mathtt{mlist})$ where $l_i : \mathtt{list}(S_i), 1 \leq i \leq q$ and $\mathtt{mlist} : \mathtt{list}(N)$. The idea is that $l_i = \Gamma^i$, while

$$\mathtt{mlist}{\downarrow}i = \langle i_1, i_2 \rangle = \mathtt{map}_T\, i\, \Gamma.$$

It is straightforward to implement the operations on contexts in terms of this representation.

- $\mathtt{emptycontext}_T = ([], \ldots, [], [])$
- $\mathtt{map}_T i(l_1, \ldots, l_q, \mathtt{mlist}) = \mathtt{mlist}{\downarrow}i$
- $\mathtt{get}_{S_i} j(l_1, \ldots, l_q, \mathtt{mlist}) = l_i{\downarrow}j$
- $\mathtt{extend}_{S_i}(l_1, \ldots, l_q, \mathtt{mlist}) x_1 \cdots x_{l_i} = L$
  where
  $$L = \mathtt{extend1}_{S_{i,l_i}}(\cdots(\mathtt{extend1}_{S_{i,2}}(\mathtt{extend1}_{S_{i,1}}(l_1, \ldots, l_q, \mathtt{mlist})x_1)x_2)\cdots)x_{l_i}$$
  and $\mathtt{extend1}_{S_{i,j}}(l_1, \ldots, l_q, \mathtt{mlist})x$ equals
  $$(l_1, \ldots, l_j + +[x], \ldots, l_q, \mathtt{mlist} + +[\langle j, \mathtt{length}_{S_j}(l_j) + 1 \rangle])$$
  where $- + +-$ is list concatenation.

Finally, we show how to represent lists and products in PCF. We represent lists by

$$\mathtt{List}(T) = (N \Rightarrow T) \times N$$

where e.g.

- $\mathtt{cons}_T =$

```
λx^T.λl : List(T)
 let (f, n) = l      in (g, n + 1)
                     where
                     g = λi^N.      if i = 0   then x
                                               else  f(i − 1)
```

- $\mathtt{empty}_T(f, n) = n = 0.$

A function taking an argument of product type

$$T \times U \Rightarrow V$$

can be replaced by its curried version

$$T \Rightarrow U \Rightarrow V$$

while a function returning a product type can be replaced by the two component functions.

This completes our description of the universal term U_T.

For each PCF type T, we define a relation $M \mathrel{\mathcal{R}_T} a$ between closed PCF terms of type T and strategies $a \in \mathtt{Str}(T)$ by

$$M \mathrel{\mathcal{R}_T} a \iff [\![M]\!] \simeq a.$$

This is extended to sequences $\tilde{M} \mathrel{\mathcal{R}_{\tilde{T}}} \tilde{a}$ in the evident fashion.

We fix a type T with subtypes $S_1, \ldots, S_q$ as in the previous discussion.

Lemma 5.5 *Let $\tilde{T} \Rightarrow S$ be a PCF type-in-context and $\sigma \in \mathtt{Str}(\tilde{T} \Rightarrow S)$ a compact strategy, where $\tilde{T}, S$ are subtypes of T. Let Γ be a closed expression of type $\mathtt{Ctxt}_T$ (which we will regard as a sequence of closed terms), and $\tilde{a}$ a sequence of strategies. Then*

$$\Gamma \mathrel{\mathcal{R}}_{\tilde{T}} \tilde{a} \Rightarrow (F_S\lceil\sigma\rceil\Gamma) \mathrel{\mathcal{R}}_S (\sigma\tilde{a}).$$

Proof By induction on the height of the finite evaluation tree corresponding to σ under Theorem 3.21 , and by cases on the Decomposition Lemma for σ. The cases for $\sigma = \Lambda_{\tilde{S}}(\bot_{\tilde{T},\tilde{S}})$ and $\sigma = \Lambda_{\tilde{S}}(\mathbf{K}_{\tilde{T},\tilde{S}}n)$ are clear.

Suppose

$$\sigma \approx \mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}, (\tau_n \mid n \in \omega)).$$

By Intuitionistic Function Extensionality Lemma, it suffices to show that, for all closed $\tilde{M}$ and strategies $\tilde{b}$ such that $\tilde{M} \mathrel{\mathcal{R}}_{\tilde{S}} \tilde{b}$

$$F_S\lceil\sigma\rceil\Gamma\tilde{M} \mathrel{\mathcal{R}}_N \sigma\tilde{a}\tilde{b}.$$

Let $\Delta = \mathtt{extend}_S\Gamma\tilde{M},\ \tilde{c} = \tilde{a}, \tilde{b}$. Then $\Delta \mathrel{\mathcal{R}}_{\tilde{T},\tilde{S}} \tilde{c}$, so by induction hypothesis,

$$F_{S_{i,j}}\lceil\sigma_j\rceil\Delta \mathrel{\mathcal{R}}_{S_{i,j}} \sigma_j\tilde{c}, \quad 1 \leq j \leq l_i$$

Hence if we define

$$\begin{aligned} M &= \Delta_i(F_{S_{i,1}}\lceil\sigma_1\rceil\Delta)\cdots(F_{S_{i,l_i}}\lceil\sigma_{l_i}\rceil\Delta) \\ &= \Delta^{i_1}_{i_2}(F_{S_{i,1}}\lceil\sigma_1\rceil\Delta)\cdots(F_{S_{i,l_i}}\lceil\sigma_{l_i}\rceil\Delta) \end{aligned}$$

where $\langle i_1, i_2\rangle = \mathtt{map}\, i\, \Delta$, then $M \mathrel{\mathcal{R}}_N c_i(\sigma_1\tilde{c})\cdots(\sigma_{l_i}\tilde{c})$. Thus if $c_i(\sigma_1\tilde{c})\cdots(\sigma_{l_i}\tilde{c}) = \bot_N$, then $[\![M]\!] = \bot_n$, while if $c_i(\sigma_1\tilde{c})\cdots(\sigma_{l_i}\tilde{c}) = n$ then $[\![M]\!] = n$.

In the former case,

$$[\![F_S\lceil\sigma\rceil\Gamma\tilde{M}]\!] \simeq \bot_N \simeq \sigma\tilde{c}.$$

In the latter case,

$$\begin{aligned} [\![F_S\lceil\sigma\rceil\Gamma\tilde{M}]\!] &\simeq [\![F_N(B\lceil\sigma\rceil n)\Delta]\!] \\ &\simeq [\![F_N\lceil\tau_n\rceil\Delta]\!], \end{aligned}$$

while $\sigma\tilde{c} \simeq \tau_n\tilde{c}$, and by induction hypothesis $F_N\lceil\tau_n\rceil\Delta \mathrel{\mathcal{R}}_N \tau_n\tilde{c}$. □

Now we define a family of relations $(\preceq_k \mid k \in \omega)$, where $\preceq_k \subseteq \omega \times \omega$, inductively as follows:

$$\begin{array}{rcl} \preceq_0 & = & \omega \times \omega \\ n \preceq_{k+1} m & \Longleftrightarrow & (D_n = \langle 2, p\rangle \Rightarrow D_m = \langle 2, p\rangle) \\ & \wedge & (D_n = \langle 3, i\rangle \Rightarrow D_m = \langle 3, i\rangle \\ & \wedge & [Hn = \langle k_1, \ldots, k_{l_i}\rangle \Rightarrow \\ & & Hm = \langle k'_1, \ldots, k'_{l_i}\rangle \wedge \bigwedge_{j=1}^{l_i} k_j \preceq_k k'_j] \\ & \wedge & \forall p : 0 \preceq p \preceq k.\ Bnp \preceq_k Bmp). \end{array}$$

We can read $n \preceq_k m$ as: the stategy coded by m simulates the strategy coded by n for all behaviours of size $\leq k$.

We write

$$n \preceq m \iff \forall k \in \omega . n \preceq_k m.$$

Lemma 5.6 *For all PCF types T, $\sigma \in \mathtt{Str}(T), k \in \omega$:*

(i) $p_k(\sigma) \preceq \sigma$.

(ii) $\sigma \preceq_k p_k(\sigma)$

Lemma 5.7 *With $S, \Gamma, \tilde{M}$ as in Lemma 5.5, and σ any strategy in $\mathtt{Str}(S)$:*

$$[\![F_S\lceil\sigma\rceil\Gamma\tilde{M}]\!] = n \iff \exists k \in \omega .\ [\![F_S\lceil p_k(\sigma)\rceil\Gamma\tilde{M}]\!] = n$$

Proof ($\Leftarrow$) By Lemma 5.6*(i)*.

($\Rightarrow$) By Lemma 5.6*(ii)*, using continuity, and hence the fact that only finitely many calls to D, H and B are made in evaluating $F_S\lceil\sigma\rceil\Gamma\tilde{M}$. (This can be made precise using Berry's Syntactic Approximation Lemma for PCF [BCL85]). □

Theorem 5.8 (Universality Theorem) *For all PCF types T and recursive strategies $\sigma \in \mathtt{Str}(T)$ with $n = \lceil \sigma \rceil$,*

$$\mathcal{M}(K_!(\mathcal{G}))[\![U_T n]\!] \simeq_T \sigma.$$

Thus every functional in $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ (equivalently, every functional in $\mathcal{M}(\mathcal{E})$ realised by a recursive strategy) is definable in PCF.

Proof For all closed $\tilde{M} : \tilde{T}$.

$$\begin{array}{lll} [\![U_T \lceil \sigma \rceil \tilde{M}]\!] = n & \Longleftrightarrow & \exists k \in \omega.\ [\![U_T \lceil p_k(\sigma) \rceil \tilde{M}]\!] = n \\ & & \text{by Lemma 5.7} \\ & \Longleftrightarrow & \exists k \in \omega.\ p_k(\sigma)[\![\tilde{M}]\!] = n \\ & & \text{by Lemma 5.5} \\ & \Longleftrightarrow & \sigma[\![\tilde{M}]\!] = n \\ & & \text{by the Approximation Lemma for strategies.} \end{array}$$

By the Intuitionistic Function Extensionality Lemma this shows that $[\![U_T \lceil \sigma \rceil]\!] \simeq \sigma$. □

In the case of cpo-enriched models, an important result due to Milner is that the fully-abstract order extensional model is unique up to isomorphism. For rational models, the situation is not quite so rigid. For example, both $\mathcal{M}(\mathcal{E})$ and $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ are fully abstract, but $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ is properly contained in $\mathcal{M}(\mathcal{E})$. To see this, note that all monotonic functions of type $N \Rightarrow N$ are sequentially realised and hence live in $\mathcal{M}(\mathcal{E})$, while only the recursive ones live in $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$. We can, however, give a very satisfactory account of the canonicity of $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$. We define a category $\mathtt{FAMOD(PCF)}$ with objects the fully abstract (rational) models of PCF. A homomorphism $F : \mathcal{M}(\mathcal{C}) \to \mathcal{M}(\mathcal{D})$ is a functor from the full cartesian closed sub category of $\mathcal{C}$ generated by the interpretation of N in $\mathcal{M}(\mathcal{C})$ to the corresponding subcategory of $\mathcal{D}$. F is additionally required to be a rational CCC functor, and to preserve the interpretation of N and of the PCF ground and first-order constants.

Theorem 5.9 (Extensional Initiality Theorem) *$\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ is initial in* $\mathtt{FAMOD(PCF)}$.

Proof Let $\mathcal{N}$ be any fully abstract model. By the Universality Theorem, there is only one possible definition of $F : \mathcal{M}(\mathcal{E}_{\mathtt{rec}}) \to \mathcal{N}$, given by

$$F(\mathcal{M}(\mathcal{E}_{\mathtt{rec}})[\![M]\!]) = \mathcal{N}[\![M]\!]$$

for all closed terms M of PCF. Since $\mathcal{M}(\mathcal{E}_{\mathtt{rec}})$ and $\mathcal{N}$ are both fully abstract,

$$\begin{array}{rrcl} & \mathcal{M}(\mathcal{E}_{\mathtt{rec}})[\![M]\!] & \leq & \mathcal{M}(\mathcal{E}_{\mathtt{rec}})[\![N]\!] \\ \Leftrightarrow & M & \sqsubseteq_{\mathtt{obs}} & N \\ \Leftrightarrow & \mathcal{N}[\![M]\!] & \leq & \mathcal{N}[\![N]\!] \end{array}$$

so this map is well-defined, and preserves and reflects order. It is a homomorphism by the compositional definition of the semantic function. □

References

[AP90] M. Abadi and G. Plotkin. A PER model of polymorphism and recursive types. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1990.

[Abr94] Samson Abramsky. Proofs as processes. *Theoretical Computer Science*, 135:5–9, 1994.

[AJ93] S. Abramsky and R. Jagadeesan. Game semantics for exponentials. Announcement on the types mailing list, 1993.

[AJ94a] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543 – 574, June 1994. Also appeared as Technical Report 92/24 of the Department of Computing, Imperial College of Science, Technology and Medicine.

[AJ94b] Samson Abramsky and Achim Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science Volume 3*, pages 1–168. Oxford University Press, 1994.

[AJM94] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF (extended abstract). In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software. International Symposium TACS'94*, number 789 in Lecture Notes in Computer Science, pages 1–15, Sendai, Japan, April 1994. Springer-Verlag.

[AM95] Samson Abramsky and Guy McCusker. Games and full abstraction for the lazy λ-calculus. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 234–243. IEEE Computer Society Press, 1995.

[AM99] Samson Abramsky and Guy McCusker. Game Semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, pages 1–56, Springer-Verlag 1999.

[Abr97] Samson Abramsky. Games in the Semantics of Programming Languages. In *Proceedings of the 1997 Amsterdam Colloquium.*

[Abr00] Samson Abramsky. Axioms for Definability and Full Completeness. In G. Plotkin, C. Stirling and M. Tofte, editors, *Proof, Language and Interaction: Essays in honour of Robin Milner*, pages 55–75, MIT Press 2000.

[Bar84] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, revised edition, 1984.

[Ber79] Gérard Berry. *Modeles completement adequats et stable de lambda-calculs.* PhD thesis, Universite Paris VII, 1979.

[BC82] Gérard Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[BCL85] Gérard Berry, P.-L. Curien, and Jean-Jacques Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and John Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1985.

[BE91] A. Bucciarelli and T. Ehrhard. Extensional embedding of a strongly stable model of PCF. In *Proc. ICALP*, number 510 in LNCS, pages 35–46. Springer, 1991.

[CF92] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proc. POPL*, pages 328–342. ACM Press, 1992.

[Cro94] Roy Crole. *Categories for Types.* Cambridge University Press, 1994.

[Cur92a] Pierre-Louis Curien. Observable sequential algorithms on concrete data structures. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 432–443. IEEE Computer Science Press, 1992.

[Cur92b] Pierre-Louis Curien. Sequentiality and full abstraction. In P. T. Johnstone M. Fourman and A. M. Pitts, editors, *Applications of Categories in Computer Science*, pages 66–94. Cambridge University Press, 1992.

[Cur93] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Progress in Theoretical Computer Science. Birkhauser, 1993.

[DP90] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[Ehr] Thomas Ehrhard. Projecting sequential algorithms on the strongly stable functions. *Annals of Pure and Applied Logic* 77(3):201–244, 1996.

[Gan93] R. O. Gandy. Dialogues, Blass games and sequentiality for objects of finite type. Unpublished manuscript, 1993.

[GM00] Dan R. Ghica and Guy McCusker. Reasoning about Idealized Algol Using Regular Languages. In *Proceedings of ICALP 2000*, Springer Lecture Notes in Computer Science, 2000.

[Gir88] Jean-Yves Girard. Geometry of interaction 2: Deadlock-free algorithms. In P. Martin-Löf and G. Mints, editors, *International Conference on Computer Logic, COLOG 88*, pages 76–93. Springer-Verlag, 1988. Lecture Notes in Computer Science 417.

[Gir89a] Jean-Yves Girard. Geometry of interaction 1: Interpretation of System F. In R. Ferro et al., editor, *Logic Colloquium 88*, pages 221–260. North Holland, 1989.

[Gir89b] Jean-Yves Girard. Towards a geometry of interaction. In J. W. Gray and Andre Scedrov, editors, *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 69–108. American Mathematical Society, 1989.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[JS93] Achim Jung and Allen Stoughton. Studying the fully abstract model of PCF within its continous function model. In *Proc. Int. Conf. Typed Lambda Calculi and Applications*, pages 230–245, Berlin, 1993. Springer-verlag. Lecture Notes in Computer Science Vol. 664.

[Loa96] Ralph Loader. Finitary PCF is undecidable. *Theoretical Computer Science*, to appear, 2000.

[Lor60] P. Lorenzen. Logik und agon. In *Atti del Congresso Internazionale di Filosofia*, pages 187–194, Firenze, 1960. Sansoni.

[Lor61] P. Lorenzen. Ein dialogisches Konstruktivitätskriterium. In *Infinitistic Methods*, pages 193–200, Warszawa, 1961. PWN. Proceed. Symp. Foundations of Math.

[Mal93] Pasquale Malacaria. Dalle macchine a ambienti alla geometria dell'interazione. Unpublished manuscript, 1993.

[MH99] Pasquale Malacaria and Chris Hankin. Non-Deterministic Games and Program Analysis: an application to security. In *Proceedings of LiCS '99*, pages 443–453, 1999.

[Man76] E. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer-Verlag, 1976.

[Mil77] Robin Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[Mul87] K. Mulmuley. *Full Abstraction and Semantic Equivalence*. MIT Press, 1987.

[Nic94] H. Nickau. Hereditarily sequential functionals. In *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg*, Lecture notes in Computer Science. Springer, 1994.

[OR95] Peter W. O'Hearn and Jon G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.

[Plo77] Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Soa87] R. I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[Sto88] A. Stoughton. *Fully abstract models of programming languages.* Pitman, 1988.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages.* Foundations of Computing. The MIT Press, Cambridge, Massachusetts, 1993.