Working draft

On full abstraction for PCF:

I. Models, observables and the full abstraction problem

II. Dialogue games and innocent strategies

III. A fully abstract and universal game model

J. M. E. Hyland[1]
Department of Pure Mathematics and Mathematical Statistics
University of Cambridge
16 Mill Lane, Cambridge CB2 1SB
United Kingdom

and

C.-H. L. Ong[2]
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD
United Kingdom

and

Department of Information Systems and Computer Science
National University of Singapore, Lower Kent Ridge Road, Singapore 0511
Republic of Singapore

April 1995

Keywords. Higher-type sequential computation, full abstraction, PCF, λ-calculus, game semantics.

[1]Partially supported by CEC ESPRIT Project CLICS-II. `martin@pmms.cam.ac.uk`

[2]Part of this work was carried out while at the University of Cambridge Computer Laboratory and supported by a research fellowship of Trinity College, Cambridge. On leave from the National University of Singapore. `Luke.Ong@comlab.ox.ac.uk`

Abstract

We present an order-extensional, order (or inequationally) fully abstract model for Scott's language PCF. The approach we have taken is very concrete and in nature goes back to Kleene [Kle78] and Gandy [Gan93] in one tradition, and to Kahn and Plotkin [KP93], and Berry and Curien [BC82] in another. Our model of computation is based on a kind of game in which each play consists of a dialogue of questions and answers between two players who observe the following Principles of Civil Conversation:

1. *Justification.* A question is asked only if the dialogue at that point "warrants" it. An answer is proffered only if a question expecting it has already been asked.
2. *Priority.* Questions pending in a dialogue are answered on a "last asked first answered" basis. This is equivalent to Gandy's "no dangling question mark" condition.

We analyse PCF-style computations *directly* in terms of partial strategies based on the information available to each player when he is about to move. Our players are required to play an *innocent strategy*: they play on the basis of their *view* which is that part of the "history" that interests them currently. Views are continually updated as the play unfolds. Hence our games are neither history-sensitive nor history-free. Rather they are view-dependent. These considerations give expression to what seems to us to be the nub of PCF-style higher-type sequentiality in a (dialogue) game-semantical setting.

Organization of the paper

This paper has three parts.

Part I begins with a brief survey of the full abstraction problem of PCF tracing its roots to old foundational problems in (higher-type) recursion theory and sequential computation considered by Platek and also by Kleene and others. We study the model theory of PCF in the light of standard ideas from both categorical logic and categorical type theory. We take a (concrete) model of PCF to be a Platek category (cartesian closed with conditionals and fixed points) equipped with what we call a simple object of numerals. In the same categorical spirit the notion of observational equivalence is analysed. Given a notion of observables on a symmetric monoidal closed category $\mathbb{C}$ (*e.g.* a model of PCF), we give a precise definition for the induced observational preorder (over homsets of $\mathbb{C}$) and study the associated quotient construction $\mathbb{C} \mapsto \hat{\mathbb{C}}$. These analyses yield a general categorical setting within which to articulate and reason about the standard (though hitherto concretely understood) properties of adequacy, order-extensionality (equivalently the context lemma) and full abstraction.

In **Part II** we formalize the class of dialogue games in which the two players involved are required to observe the disciplines of justification and priority mentioned above. We make a category $\mathbb{CA}$ out of such games: objects are computational arenas which are a kind of environment for such dialogue games, and maps are innocent strategies. The main result of Part II is that the category $\mathbb{CA}$ is cartesian closed and enriched over dI-domains. With respect to an intrinsic notion of observables, $\mathbb{CA}$ satisfies the context lemma; equivalently the associated observational quotient $\hat{\mathbb{CA}}$ is order-extensional.

The category $\mathbb{CA}$ is considered as a model of PCF and **P** – an extension of PCF by definition-by-cases constructs – in **Part III**. We prove a strong definability theorem: there is an order-isomorphism between compact elements of the model and a class of finite canonical forms of **P** (ordered by Ω-matching). As a corollary the observational quotient $\hat{\mathbb{CA}}$ of $\mathbb{CA}$ is an order-extensional, order fully abstract model for PCF. The strong definability result extends to a universality theorem for $\mathbb{CA}$: modulo observational equivalence, all recursive innocent strategies are PCF-definable. We conclude this paper with a discussion of and comparison with related work. Directions for further research are identified.

Contents

Part I

Models, observables and the full abstraction problem

1 Introduction

Historical remarks

In the early 1940's, Gödel considered a notion of primitive recursive functionals of finite type, which we now call Gödel's System **T**, in connection with what came to be known as the Dialectica Interpretation [Göd58, Göd90]. Gödel presented his results as a contribution to a *liberalised* version of Hilbert's programme.[3] Gödel's work was later extended to the Bar Recursive Functionals by Spector [Spe62] who used them to give a constructive consistency proof for classical analysis. However the first full-blown generalization of ordinary Recursion Theory to higher types was made by Kleene in the late 1950's (see [Kle59, Kle63] for a formulation in terms of computation schemes). In Kleene's theory, a notion of partial recursive function of higher type is defined over a total type structure. Recursion is introduced by a computation scheme which essentially encapsulates the Second Recursion Theorem. In this theory, partial recursive functions are not closed under substitution, and a natural formulation of the First Recursion Theorem fails. Kleene exhibited these features in [Kle63] where he also observed in passing that a theory involving application of (partial) functions to partial functions might be possible. A succinct account of Kleene's theory and of its attendant difficulties is contained in Gandy [Gan67].

An attempt was made by Platek in his thesis [Pla66] to develop a recursion theory on partial functions of higher type which avoids the problems of the Kleene theory. The type structure Platek considers is that of hereditarily order-preserving functions over what we now call the flat CPO of natural numbers. Platek couched his theory in terms of computation schemes, and recursion is introduced by a scheme amounting to the First Recursion Theorem. The essentials of the theory are definition by cases and least fixed points, and Platek does introduce a λ-calculus formulation which can be regarded as a precursor to PCF.

The formal system PCF was introduced by Scott in a famous paper [Sco93] which remained an unpublished manuscript for a long time until its recent appearance in the Böhm Festschrift. The syntax[4] of PCF is simple enough: it is essentially the simply-typed lambda calculus augmented by general recursion in the form of a fixed-point operator at every type and definition by cases at ground types, and further augmented by basic arithmetic operations. Scott intended PCF to be a "logical calculus (or algebra)" for studying program equivalence and other algebraic and logical properties of programs by using (simple) type theory [Chu40].

A major theme of Scott's work is the relationship between the *logical* types which are the higher types, and the data types which are the ground and first-order types. The former are used to study the latter; the theory of data types requires the higher-type objects — the computable functionals — for its formalization, as emphasized by Scott. Of course, Scott had in mind a semantics in terms of what we now call Scott continuous functions and for this interpretation, a quite straightforward operational semantics is appropriate: we can think of all the computations as being finite. Hence the theory introduced by Scott is in principle implementable and has been the focus of much attention in Computer Science. Another theme in Scott's paper is completeness about which he asked several questions. One such question concerns the power of expression of the language with respect to the continuous function space model – in a word, definability. What came to be known as the full

[3] Hilbert had set out to justify classical mathematics systematically in terms of notions which should be as intuitively clear as possible. A focus of his programme was the consistency of classical number theory; he wanted to find these basic notions in the domain of "finitary mathematics". Bernays subsequently pointed out that in order to prove the consistency of classical number theory, it was necessary to extend Hilbert's finitary standpoint by admitting "abstract concepts" of a certain kind in addition to the combinatorial concepts relating to symbols. Gödel introduced System **T**, which is essentially the simply-typed λ-calculus augmented by primitive recursion, as a vehicle for expressing the "abstract notion". He believed that System **T** would be the key to making Hilbert's programme viable in the modified sense.

[4] In [Sco93] Scott considered a version of PCF based on typed **S**- and **K**-combinators.

abstraction problem for PCF was adumbrated: Scott observed that parallel or is not definable in the language and that an implementation on a "sequential machine" would require a dovetailing strategy. System **T** and PCF (as a formal system in the sense of Scott's original presentation as opposed to a programming language) are similar in various ways, though there is an important difference: Scott's approach admits partial functions whereas Gödel's is only concerned with total functions.

1.1 The programming language PCF

In [Plo77] Plotkin presented PCF explicitly as a programming language and studied the relationship between its operational semantics and denotational semantics which is based on the Scott continuous function space model. Types of the language are just Church's simple types [Chu40]. In the following we shall also refer to them as PCF-types. They are defined as follows:

$$\begin{array}{rcll} A & ::= & \iota & \text{natural numbers} \\ & | & o & \text{booleans} \\ & | & A \Rightarrow A & \text{arrow or function type.} \end{array}$$

We use the meta-variable β to range over ground types ι and o. As usual $\Rightarrow$ associates to the right: $A_1 \Rightarrow A_2 \Rightarrow A_3$ is read as $A_1 \Rightarrow (A_2 \Rightarrow A_3)$. Note that with $n \geqslant 0$, each simple type can be uniquely expressed as $A_1 \Rightarrow A_2 \cdots \Rightarrow A_n \Rightarrow \beta$, which we abbreviate as $(A_1, \cdots, A_n, \beta)$, where β is a ground type. For example the type $((\iota \Rightarrow \iota) \Rightarrow \iota \Rightarrow \iota) \Rightarrow \iota \Rightarrow \iota$ is abbreviated as $(((\iota, \iota), \iota, \iota), \iota, \iota)$. The *height* $\mathsf{ht}(A)$ of a type A is defined by recursion as follows:

$$\begin{array}{rcl} \mathsf{ht}(\beta) & \stackrel{\text{def}}{=} & 0 \\ \mathsf{ht}(A \Rightarrow B) & \stackrel{\text{def}}{=} & \max(\mathsf{ht}(A) + 1, \mathsf{ht}(B)). \end{array}$$

We say that an object is *type-n* if it has type of height n. Intuitively height measures how "higher-order" a type really is. In general the mathematical difficulties associated with the higher-order objects stem from nesting of the arrow on the left.

For each type A, we fix a denumerable set of variables. Raw PCF-terms are defined by the following grammar:

$$\begin{array}{rcll} s & ::= & \Omega^A & \text{undefined term} \\ & | & c^A & \text{constant} \\ & | & x & \text{variable} \\ & | & (s \cdot s) & \text{application} \\ & | & (\lambda x : A.s) & \text{abstraction} \\ & | & \mathbf{Y}^A(s) & \text{general recursive term, or Y-term;} \end{array}$$

where c^A ranges over the set $\mathbf{A}$ of basic arithmetic constants which we will introduce shortly. Whenever type information is irrelevant, we omit type labels and write Ω^A, c^A, $x : A$ and $\mathbf{Y}^A(-)$ simply as Ω, c, x and $\mathbf{Y}(-)$ respectively. We shall write $(s \cdot t)$ simply as st. As usual, application associates to the left: $st_1 \cdots t_n$ abbreviates $(\cdots((st_1)t_2) \cdots t_n)$, and we routinely omit as many parentheses as we safely can. PCF-*terms* are raw terms that are well-typed. The phrase $s : A$ means that the type of the term s is A, derived according to the following rules:

$$\Omega^A : A \qquad\qquad c^A : A$$

$$\frac{s : A \Rightarrow A}{\mathbf{Y}^A(s) : A} \quad \frac{s : A_1 \Rightarrow A_2 \qquad t : A_1}{(s \cdot t) : A_2} \quad \frac{s : A_2}{(\lambda x : A_1 . s) : A_1 \Rightarrow A_2}$$

The set $\mathbf{A}$ of basic arithmetic constants is presented together with their types as follows:

n	:	ι	numerals, for each natural number $n \geqslant 0$
t, f	:	o	booleans: truth and falsity
succ	:	$\iota \Rightarrow \iota$	successor
pred	:	$\iota \Rightarrow \iota$	predecessor
$\mathsf{zero?}$	:	$\iota \Rightarrow o$	test for zero
cond^ι	:	$o \Rightarrow \iota \Rightarrow \iota \Rightarrow \iota$	natural number conditional
cond^o	:	$o \Rightarrow o \Rightarrow o \Rightarrow o$	boolean conditional.

The notion of free and bound variables is completely standard; a closed term is a term without any free variables. We write the term substitution (as opposed to context substitution) operation as $s[t/x]$ which means "in s, substitute the term t for every free occurrence of x", taking care to rename bound variables where necessary so as to avoid variable capture. See, for example, [Bar84, p. 27] for a formal definition.

Operational Semantics *Programs* of PCF are just closed terms of ground type. (For this reason we shall often refer to a ground type as a *program type*.) Values are λ-abstractions and constants (less Ω); values are ranged over by the meta-variable v. Following the function paradigm, to compute a program in PCF is to evaluate it. We present the operational semantics of PCF in terms of a Martin-Löf style evaluation relation. Formally we define a relation $\Downarrow$ between closed terms and values inductively over the following rules. We read $s \Downarrow v$ as "the closed term s evaluates to the value v".

$$v \Downarrow v \qquad \frac{u[t/x] \Downarrow v}{(\lambda x.u)t \Downarrow v} \qquad \frac{s \Downarrow v \quad vt \Downarrow v'}{st \Downarrow v'}$$

$$\frac{s \Downarrow \mathsf{t} \qquad u \Downarrow v}{\mathsf{cond}^\beta s u u' \Downarrow v} \qquad \frac{s \Downarrow \mathsf{f} \qquad u' \Downarrow v}{\mathsf{cond}^\beta s u u' \Downarrow v}$$

$$\frac{s\mathbf{Y}^A(s) \Downarrow v}{\mathbf{Y}^A(s) \Downarrow v}$$

$$\frac{s \Downarrow n}{\mathsf{succ} s \Downarrow n+1} \qquad \frac{s \Downarrow n+1}{\mathsf{pred} s \Downarrow n} \qquad \frac{s \Downarrow 0}{\mathsf{pred} s \Downarrow 0}$$

$$\frac{s \Downarrow 0}{\mathsf{zero?} s \Downarrow \mathsf{t}} \qquad \frac{s \Downarrow n+1}{\mathsf{zero?} s \Downarrow \mathsf{f}}$$

We further define: for any program s

$$s\Downarrow \stackrel{\text{def}}{=} \exists v. s \Downarrow v,$$
$$s\Uparrow \stackrel{\text{def}}{=} \neg[s\Downarrow].$$

Remark 1.1.1 Evaluation may be implemented by a process of one-step reduction. (Indeed we could have presented the operational semantics equivalently in terms of a "small-step", Plotkin-style transition relation.) According to this notion of reduction, terms are reduced following the left-most reduction strategy, β-contraction is carried out in a call-by-name fashion, and no reduction is permitted "under a lambda".

There is an operational notion of program equivalence which programmers understand well: two program fragments are equivalent if they can always be *interchanged* without affecting the visible or observable outcome of the computations. This criterion of sameness which is called *observational equivalence* is expressed in terms of invariance of observable outcome under all program contexts. Let s and t be PCF-terms of the same type. We say that *s observationally approximates t*, written $s \lesssim t$, if for every type-compatible program context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, and for any value v, if $C[s] \Downarrow v$ then $C[t] \Downarrow v$. (To our knowledge the idea of a preorder on terms defined by a universal quantification over contexts goes back to Morris' thesis [Mor68].) Two program fragments s and t are said to be observationally equivalent if both $s \lesssim t$ and $t \lesssim s$. We write program contexts as $C[X]$ where the "hole" represented by X is to be thought of as a kind of meta-variable. As usual, $C[s]$ means "the term which is obtained from the context $C[X]$ by substituting s for every occurrence of X in $C[X]$". Note that variable capture is possible (and intended) in context substitution.

1.2 The type theory PCF

The operational semantics for PCF encapsulates a deterministic reduction or computation strategy for the programming language; but also it reflects an intuitive understanding of the meaning of the terms. In this view the reductions are justified as the replacement of a term by an equal term. Thus the intuitive semantics can be given expression in an equational theory. In the case of PCF this amounts to a type theory related to Scott's original formulation. Our (core) type theory for PCF $\mathbb{T}$ is given as follows. We take the typing rules already given, and define a relation $s = t$ on typed terms (in context) by taking, in addition to the usual rules for equality, the following:

$$
\boxed{\begin{array}{c}
(\lambda x : A.s)t = s[t/x] \qquad \lambda x : A.sx = s \quad (\text{if } x \text{ not free in } s) \\[4pt]
\mathsf{cond}^{\beta}\mathbf{t}st = s \qquad \mathsf{cond}^{\beta}\mathbf{f}st = t \\[4pt]
s(\mathbf{Y}^{A}(s)) = \mathbf{Y}^{A}(s) \\[4pt]
\mathsf{succ}\ n = n+1 \qquad \mathsf{pred}\ n+1 = n \qquad \mathsf{pred}\ 0 = 0 \\[4pt]
\mathsf{zero?}0 = \mathsf{t} \qquad \mathsf{zero?}n+1 = \mathsf{f}
\end{array}}
$$

It is important that there be a good relation between the reduction relation $\Downarrow$ of the operational semantics and the equality of the type theory. This is given by the following proposition.

Proposition 1.2.1 *For any programs s and t, if s and t are equal in the type theory $\mathbb{T}$ then for any ground value v*

$$s \Downarrow v \quad \Longleftrightarrow \quad t \Downarrow v.$$

Proof A more or less straightforward application of the Church-Rosser theorem and a standardization theorem following, for example, the treatment in [Plo75]. $\square$

Corollary 1.2.2 *For any program s in the type theory and ground value v,*

$$s = v \text{ in the type theory if and only if } s \Downarrow v.$$

$\square$

What is commonly called a denotational semantics for PCF is essentially some kind of interpretation of (model for) the type theory which we have just introduced. The usual form of a model for PCF is that the types are interpreted as domains and the terms as continuous (or stable continuous) maps between domains. In this introduction we shall restrict attention to these traditional models. Later in the paper however we shall introduce a more abstract categorical notion of model for PCF; this provides a more appropriate context in which to understand our results.

The standard model In [Sco93] Scott gave a denotational semantics for PCF. Program types (booleans and the natural numbers) are interpreted by the respective flat CPOs, and function types by Scott continuous function space. We shall call this model the *standard* (continuous) model of PCF. Continuity is used to determine the way fixed point operators are interpreted *i.e.* standardly as the least upper bound of the ω-increasing chain of successive iterates, see *e.g.* [Tar55]. Note that the standard model is *order-extensional*[5] in the sense that function types are interpreted by sets of functions which are ordered pointwise.

Adequacy and full abstraction More generally, writing the denotation of a program s as $[\![s]\!]$, we say that the denotational semantics $[\![-]\!]$ is *adequate* if for every pair of type-compatible terms s and t,

$$[\![s]\!] \sqsubseteq [\![t]\!] \qquad \Longrightarrow \qquad s \precsim t.$$

If, in addition, the converse is also valid, that is to say,

$$[\![s]\!] \sqsubseteq [\![t]\!] \qquad \Longleftrightarrow \qquad s \precsim t,$$

then the denotational semantics is said to be order (or inequationally) *fully abstract* for the language. To our knowledge the notion of full abstraction is due to Milner [Mil75], though it seems implicit in work in the pure lambda calculus by Plotkin, Morris [Mor68], Wadsworth [Wad76, Wad78], Hyland [Hyl76] and others. (The definition of adequacy and full abstraction which we have just given is the traditional one. In the sequel, we shall present the same notions in a more general, categorical setting.) Adequacy and full abstraction tell us how well the operational and the denotational views of program equivalence relate to each other. They are indications of how reliable or how “fitting” the denotational model is in relation to the language. More specifically, adequacy assures us that the model is reliable enough for affirming observational equivalence between two terms since denotational equality suffices; but the model is not necessarily reliable for refuting equivalence for which we need full abstraction. Adequacy is often easy to establish, but this is not so for full abstraction. A model is not fully abstract usually because it is in some sense too rich a structure for the language: it contains semantic objects which “cannot be computed” by the programming language. Conversely, a model which is fully abstract for a language provides a very satisfactory characterization of (the observational equivalence of) the language in terms of the denotational model.

Plotkin showed in [Plo77] that the standard model is adequate but not fully abstract for PCF. He also pointed out the reason for the failure of full abstraction. The mismatch may be explained, in a nutshell, by the fact that while PCF-programs correspond to “sequential” algorithms, the standard Scott-continuous function space model contains “parallel” functions, or more precisely, functions which can only be implemented by parallel algorithms (e.g. parallel or). This point was made explicit by Plotkin in [Plo77] (see also [Saz75]) as follows.

Theorem 1.2.3 (Plotkin) *The standard Scott-continuous function space model is fully abstract for the programming language which is obtained by extending* PCF *by a parallel conditional constant.* □

[5]More precisely, this means that for any partially ordered domains D^{A_1} and D^{A_2} which interpret the types A_1 and A_2 respectively, and for any elements f, g of $D^{A_1 \Rightarrow A_2}$, $f \leqslant g$ if and only if $f(a) \leqslant g(a)$ for every $a \in D^{A_1}$.

An important conceptual advance was made by Plotkin and Milner in understanding full abstraction. They identified a necessary and sufficient condition for full abstraction.

Theorem 1.2.4 (Plotkin-Milner) *Any continuous, order-extensional model of* PCF *which follows the standard*[6] *interpretation is a system of Scott domains. Further, such a model is fully abstract if and only if all compact elements of the model are* PCF*-definable.* □

Plotkin and Milners' result leaves open the question of whether there is a denotational model which is fully abstract for PCF proper. This was quickly answered by Milner [Mil77]:

Theorem 1.2.5 (Milner) *There is a unique (up to isomorphism) order-extensional, continuous fully abstract model for* PCF. □

1.3 The full abstraction problem for PCF

While there seems to be a consensus that the full abstraction problem for PCF is difficult, there is much less agreement on what the problem is. At one level this question seems superfluous: for we already know that there is a unique fully abstract model for PCF — witness Milner's construction. In our view the thrust of the problem has to do with the (philosophical) question of what a good model is. A good model enlightens; it gives a new perspective on the behaviour or operational semantics of the programming language in question. There is no doubt that Milner's result settles an important question and his construction is a valuable contribution, at least from a mathematical point of view. Nonetheless because his construction is essentially a term model, it does not much increase our understanding of PCF beyond what can already be gleaned directly from the syntax. One way to formulate the problem which, we believe, strikes at the root of the issue is the following:

> **The full abstraction problem for** PCF *Give an abstract, synthetic account of the unique order-extensional, fully abstract model of* PCF *as identified by Milner.* □

It is worth expanding on the two operative words. By an *abstract* model, we mean a model which is constructed without recourse to the syntax or operational semantics of the language. In fact the more computationally neutral the model is in its conception, the more apposite it is as a solution. By *synthetic* description is meant a constructive, axiomatic explanation of the function space which interprets the PCF function types (in terms of the respective interpretation of the components). For example, these criteria rule out Milner's construction even though the model is fully abstract. In contrast the interpretation of a program as a continuous function is evidently abstract. The synthetic description of the Scott continuous function space model of PCF is also satisfactory in every way: the category of Scott domains (say) and continuous functions is cartesian closed and may be presented constructively.

Since the crux of the full abstraction problem is the characterization of *sequential* computations, we may reformulate the full abstraction problem for PCF as the problem of finding an abstract, synthetic characterization of higher-type, sequential, PCF-definable functionals. Formulated in this way, we highlight the epistemological difficulties inherent in the problem, for we do not have a proper definition of higher-type sequentiality from first principles. At any rate, to date there is certainly no notion of higher-type sequentiality which can be said to be canonical in any sense. In fact it is unclear whether

[6] An interpretation of PCF is said to be *standard* if the ground types are interpreted as the respective flat CPOs with the constants interpretated in the standard way.

there are various inequivalent notions of higher-type sequentiality, all of them equally appealing; or whether as is the case for effective computability, there is just one notion under different guises.

Some criteria The full abstraction problem for PCF in the above qualitative sense is by its nature incapable of being precisely specified because the underlying considerations are philosophical and so more or less subjective in nature. Therefore, it seems all the more important to lay down a few criteria which should be as objective as possible so that progress in understanding the problem may to some extent be calibrated and be seen in perspective.

A continuous model of PCF is a CPO-enriched cartesian closed category of a certain kind (the exact nature is spelt out in the sequel). In view of Theorem 7.3.4, we might say that one weak form of the full abstraction problem for PCF boils down to the following:

> **Observational abstraction for PCF** *Find a* CPO*-enriched cartesian closed category of Scott domains (providing a standard interpretation of* PCF*) all of whose compact elements are* PCF*-definable.* □

Note that there is no intrinsic reason why the denotation of a PCF-program in such a model must be a set-theoretic function.

In [JS93] Jung and Stoughton seek "a weak but precise minimal condition that a semantic solution of the full abstraction should satisfy". The second criterion, which we call the *Jung-Stoughton criterion*, imposes an effectivity constraint on the way the fully abstract model is presented. It seeks an effective construction of the fully abstract model restricted to *finitary* PCF *i.e.* that part of PCF which is generated from the boolean base type.

The third criterion is the hardest to satisfy. It asks for an axiomatic characterization of higher-type, PCF-sequential functions. By way of comparison, if it is right to think of the first two criteria as contributions to the representation theory of higher-type sequentiality, then the third is in the business of giving it a *definition*. One appealing way to characterize PCF-definable functions is to express it in terms of an appropriate preservation property in an order-theoretic framework, for example, in the style of Bucciarelli and Ehrhards' strongly stable functions [BE91, BE93]. Another way is to characterize it topologically say, as a refinement of the Scott topology. Such an approach is likely to be very hard, if it is at all feasible.

1.4 Quest for a solution: a survey

This sets the scene for a line of research motivated by the quest for a solution to the full abstraction problem (in the qualitative sense) for PCF. As Plotkin already intimated in [Plo77], the key to the solution is an abstract characterization of sequential computation. To give that, one needs a proper understanding of sequentiality. The matter is straightforward in the case of first-order computation. Milner and others have already obtained satisfactory abstract description of first-order sequential functions. Intuitively the meaning of sequential computation is clear enough: it is to do "one thing at a time" at any intermediate stage of the computation, and possibly in a specific order. The real difficulty lies in describing sequential, functional computation at higher types.

The first major contribution was made by Kahn and Plotkin and reported in a technical report written in French. Like the papers of Scott and Plotkin mentioned in the preceding, a revised version of the paper [KP93] in English has also appeared in the recently published Böhm Festschrift. They introduce a class of mathematical structures known as *concrete data structures* (CDS). A CDS is an elaborate structure specially designed to articulate sequential computations. The framework of CDSs and Kahn-Plotkin sequential functions is a highly innovative conceptual advance in understanding higher-type sequentiality. Their framework does not give rise to a cartesian closed category. (This is hardly surprising since it was not their aim to carry out a systematic analysis of higher-type functional computation in that paper.) Its primary objective was to examine the behaviour of stream-like computation.

The search for a cartesian closed category of "sequential functions" became the focus of research. Historically the research bifurcated at this point. The crux of the matter is the abstract characterization of sequential, functional computation at higher type. The sticking point lies in an apparent tradeoff between the two essential features: on the one hand, sequential computation which is an inherently intensional notion; and on the other, the requirement that such computations interact with each other in a functional, or extensional way. So to characterize sequential functions is to find an appropriate setting in which both properties can be held in tension. Unfortunately, based on the work of Berry and Curien in the late 1970's, it would seem that in order to get a cartesian closed category of "sequential functions", one of the two criteria has to give.

One major effort consisted in relaxing the constraints of sequentiality but staying within the framework of functions. This led Berry to the notion of *stability* [Ber78]. The appropriate maps are *stable functions* which are continuous functions that preserve greatest lower bound of consistent (or "upper bounded") subsets; and the objects are *dI-domains* — Scott domains which satisfy a distributivity property and axiom (**I**) meaning that every compact element can only dominate finitely many elements. Stable functions are not ordered by the standard extensional (or point-wise) ordering[7] but by a new ordering called stable ordering. A major result is that the category of dI-domains and stable functions is cartesian closed.

The other approach builds on the central ideas behind the framework of CDS and Kahn-Plotkin sequential function but sacrifices extensionality. Thus, Berry and Curien introduced *sequential algorithms* over CDSs [BC82] (see also Curien's book [Cur93] for a comprehensive introduction). Sequential algorithms may be thought of as intensional refinements or "implementations" of Kahn-Plotkin sequential functions. There are two reasons why this way of thinking is appropriate. First it is possible to express each sequential algorithm as a pair of the form (f, ϕ) where f is just a Kahn-Plotkin sequential function, and ϕ, referred to as the associated computation strategy, is a partial function that picks out a sequentiality index at each stage of the computation. Secondly it is a theorem that the quotient of the CPO of sequential algorithms by the extensional equality is isomorphic to the CPO of Kahn-Plotkin sequential functions with respect to the stable ordering. Remarkably, unlike Kahn-Plotkin sequential functions, sequential algorithms do give rise to a cartesian closed category.

Each of the approaches gives rise to a CPO-enriched cartesian closed category and provides a continuous model for PCF but none leads to a solution of the full abstraction problem for PCF. In the case of the stable function space model, a simple reason[8] is that the ordering in question is not the extensional ordering but rather the stable ordering. In the case of the model associated with sequential algorithms, the morphisms are not even functions.

Recently, drawing on their intuitions as programmers, Cartwright and Felleisen [CF92, CCF94] introduced a continuous, order-extensional model for PCF which is based on what they call *observably sequential functions*. Curien [Cur92a] immediately realised that the observably sequential functions were a natural *extensional* refinement of sequential algorithms. This is remarkable because the sequen-

[7] Care should be taken not to confuse the two concepts: *extensional* object and *order-extensional* functions. We use the adjective *extensional* simply to mean the property of being a function as opposed to, say, an algorithm which is an intensional thing. However, even if the maps of an order-enriched category are extensional, they are not necessarily *order-extensional* i.e. ordered extensionally. The category of dI-domains and stable functions is a case in point.

[8] A "deeper" explanation has to do with a subtle point about the extensional way in which PCF functionals interact with function arguments. Curien has shown that there is no PCF-term of type $(o \Rightarrow o \Rightarrow o) \Rightarrow o$ which distinguishes between the left- or and right-or (say). However, sequential algorithms are more intensional: there is a sequential algorithm which discriminates between two computations which only differ *intensionally* in the above sense.

tial algorithms being considered in the extended setting, which are called *observable algorithms*, are still very much intensional in nature, and are most succinctly represented as a kind of decision trees. The key to this surprising development is that the concrete data structures are now equipped with "error values". To ensure a well-behaved mechanism of function application, observable algorithms are required to "percolate errors to the top" when they are applied to arguments. A main result is that the category of DCDSs with error values and observable algorithms is cartesian closed. The associated model is not fully abstract for PCF, but it is for a language called SPCF which is PCF extended by error values and escape handling control facilities much resembling the `catch` facility in some versions of the programming language Lisp.

Kahn-Plotkin sequentiality and Berry and Curiens' sequential algorithm are both formulated within the rather concrete setting of CDS. Kahn, Plotkin, Winskel [Win80, Win87] and others have proved various representation theorems. One result shows that this approach apply to a (rather) restricted class of CPOs known as *concrete domains*. Can these two leading ideas in the understanding of higher-type sequentiality be generalized to a more abstract setting? In a series of papers [BE91, BE93, Ehr94], Bucciarelli and Ehrhard set out to answer this question systematically. They propose an abstract framework called *sequential structure* which is a pair $\langle X_*, X^* \rangle$ where

- X_*, the collection of "data" or "answers", is a dI-domain, and
- X^* is the collection of (a kind of) linear maps ("questions") from X_* to the two-point dI-domain ($\bot < \top$). An element of X^* should be thought of as a linear property of elements of X_*.

Think of a sequential structure as a concrete data structure made abstract. Their key idea was to replace cells with a class of linear maps. States of a CDS then correspond to points of the data space X_*. Remarkably, in this abstract setting, sequential algorithms can be defined quite naturally as pairs (f, ϕ) where f, a sequential function, describes the input-output behaviour of the algorithm; and ϕ, a partial function, describes its intensional properties. The enabling relation in a CDS which formalises a notion of "immediate reachability" or "adjacency in the ordering" also has a natural, abstract representation in the setting of sequential structure. Ehrhard and Bucciarelli show that a cartesian closed category of sequential structures with enabling and sequential algorithms can be constructed; and furthermore, into this category, the category of DCDSs and sequential algorithms can be fully and faithfully embedded. Thus the goal of extending sequential algorithms to an abstract setting is achieved.

Bucciarelli and Ehrhard [BE91, Buc93] also introduced the notion of *strong stability*. They were motivated by the observation that for DCDSs, Kahn-Plotkin sequential functions can be given an equivalent description in more algebraic terms. According to this definition, a sequential function is a continuous function preserving a certain class of meets. They then cast this idea in a more abstract setting. The "domains" are dI-domains D equipped with a collection $\mathbf{C}(D)$ of finite subsets of D satisfying a number of axioms. Call the collection $\mathbf{C}(D)$ a *coherence* and any of its elements a *coherence property*. A continuous function $f : D \longrightarrow E$ between dI-domains with coherence is said to be *strongly stable* if

- it preserves coherence properties *i.e.* $f(A)$ is in $\mathbf{C}(E)$ whenever A is in $\mathbf{C}(D)$, and
- it preserves greatest lower bound of coherence properties *i.e.* $f(\sqcap A) = \sqcap f(A)$ for any A in $\mathbf{C}(D)$.

Their result is that the category of dI-domains with coherence and strongly stable functions is cartesian closed. We know that the associated model is not fully abstract for PCF; but how close does it model "sequential functions"? At first order, strong stability coincides with Kahn-Plotkin sequentiality. However, at higher order, we find ourselves at a loss conceptually for we are faced with a fundamental question: is there a *standard* or canonical definition for higher-type sequentiality?

In [Ehr94] Ehrhard shows that any strongly stable function which arises from the model is the "extensional component" of a sequential algorithm. More precisely a cartesian closed category is constructed whose objects are triples $\langle E, X, \pi \rangle$. In such a triple, E is a sequential structure, X is a hypercoherence, and π is a function from E_* (the space of points of E) to $\mathrm{qD}(X)$, the qualitative domain induced by X. The function π is required to be linear, strongly stable (with respect to both the linear coherence induced by E^* on E_*, as well as the coherence induced by the hypercoherence X on $\mathrm{qD}(X)$) and onto. (Hypercoherence (see [Ehr93]) is a simplified framework for dealing with strong stability. A hypercoherence is a hypergraph that gives rise naturally to a qualitative domain equipped with a coherence.) The intuition is this: E_* is the space of sequential algorithms, $\mathrm{qD}(X)$ is a space of strongly stable functions, and π is the "forgetful" operation which sends any sequential algorithm onto its generalized extensional component. In this set up, the force of the function π being onto is that any strongly stable function is in some sense the extensional component of a sequential algorithm.

Brookes and Geva [BG92] have adopted a topological approach in an attempt to characterize sequentiality. They propose a general definition of sequential functions on Scott domains, characterized by a generalized notion of topology (along the lines of [Lam93]). This notion of sequential function turns out to coincide with the Kahn-Plotkin notion of sequential function when restricted to distributive concrete domains, but it considerably expands the class of domains for which sequential functions may be defined. Ordered stably, the sequential functions between two dI-domains form a dI-domain (the analogous property fails for Kahn-Plotkin sequential functions). However the category of dI-domains and sequential functions is not cartesian closed because application is not sequential.

Kleene's approach

Persisting in the background of these developments is a deeper, more philosophical question of whether there is such a thing as a *canonical* notion of sequential computation at higher type. Clearly, the kind of computation *defined* by PCF is at least a contender for such a standard. But it seems to us that there is no compelling evidence (yet) that PCF-style computation is the only acceptable notion of higher-type sequentiality. The problem of characterizing higher-type sequentiality should be thought of in connection with a problem which Kleene posed in [Kle78] (see also [Kle80]).

> **Kleene's problem**. Find "a class of functions which shall coincide with all the partial functions which are 'computable' or 'effectively decidable', so that Church's 1936 Thesis will apply with the higher types included."

In fact in this paper, Kleene initiated what is in effect an attack in the full abstraction problem for PCF. The series of four papers by Kleene are all concerned with an attempt to give meaning to PCF (or rather to Kleene's own preferred version of Platek's recursion in terms of schemes) in terms of rules for a dialogue. Kleene's idea of a dialogue developed in parallel with and independently of the work on CDSs. While Kleene was not able to obtain a definitive characterization at higher types, the general game-theoretic perspective, a version of which we present in this paper, is already present in his work.

Kleene's initiative was followed up by Robin Gandy and his student Giovanni Pani. Unlike Kleene and Platek, who considered only monotonic (*i.e.* order-preserving) functions, Gandy and Pani have been working in the continuous framework usual in computer science. Their work is not published, but they have investigated a number of possible approaches and have accumulated numerous (counter) examples. One of us (Hyland) has talked informally with Gandy and Pani about their ideas on a number of occasions. In particular, Gandy first pointed out the importance of his "no dangling question mark" condition for an explanation of PCF-style computability. (The account of approach currently favoured by Gandy which we have seen leads us to believe that it differs from that which we present.) A more detailed comparison of our approach with Gandy's will be given in §9. We also discuss there the little known work of Sazonov who in the mid 1970s produced a machine oriented characterization of the PCF-definable functionals. (The algorithmic work of the "Siberian school" was roughly contemporaneous with but independent of the early work of Milner and Plotkin.)

The question of higher-type sequentiality and Kleene's seemingly more general problem are of fundamental importance to Computer Science. They certainly deserve further investigation. For a survey of the full abstraction problem of PCF, see *e.g.* [BCL86, Cur92b, Ong95]. Curien's book (second edition) [Cur93] provides an excellent account of the main body of research inspired by the full abstraction problem of PCF.

1.5 Outline of the paper

In the next section, we study the model theory of PCF in the light of standard ideas from both categorical logic and categorical type theory. We take a (concrete) model of PCF to be a Platek category (cartesian closed with conditionals and fixed points) equipped with what we call a simple object of numerals. In the same categorical spirit the notion of observational equivalence is analysed. Given a notion of observables on a symmetric monoidal closed category $\mathbb{C}$ (*e.g.* a model of PCF), we give precise definition for the induced observational preorder (over homsets of $\mathbb{C}$) and study the associated quotient construction $\mathbb{C} \mapsto \hat{\mathbb{C}}$. These analyses yield a general categorical setting within which to articulate and reason about the standard (though hitherto concretely understood) properties of adequacy, order-extensionality (equivalently the context lemma) and full abstraction.

In Part II we formalize the class of dialogue games in which the two players involved are required to observe the disciplines of justification and priority mentioned above. We make a category $\mathbb{CA}$ out of such games: objects are computational arenas which are a kind of environment for such dialogue games, and maps are innocent strategies. The main result of Part II is that this category $\mathbb{CA}$ is cartesian closed and enriched over dI-domains. With respect to an intrinsic notion of observables, $\mathbb{CA}$ satisfies the context lemma; equivalently $\mathbb{CA}$ is order-extensional.

The category $\mathbb{CA}$ is considered as a model of PCF and **P** – an extension of PCF by definition-by-cases constructs – in Part III. We prove a strong definability theorem: there is an order-isomorphism between compact elements of the model and a class of finite canonical forms of **P** (ordered by Ω-matching). As a corollary the observational quotient $\hat{\mathbb{CA}}$ of $\mathbb{CA}$ is an order-extensional, order fully abstract model for **P**. The strong definability result extends to a universality theorem for $\mathbb{CA}$: modulo observational equivalence, all recursive innocent strategies are PCF-definable. We conclude this paper with a discussion of and comparison with related work. Directions for further research are identified.

Chronology The results presented here were first announced in a message entitled "Dialogue games and innocent strategies: an approach to (intensional) full abstraction for PCF" in the `Types` and `Linear` email lists in July '93 in conjunction with a preliminary announcement of Abramsky, Jagadeesan and Malacaria entitled "Games and full abstraction of PCF".

2 Models of PCF

2.1 Categorical semantics

The categorical perspective Aspects of the models of PCF which we present in this paper do not fit quite naturally into the context of denotational semantics as traditionally conceived. Hence we think it worth describing in outline one notion of model for PCF from the point of view of categorical logic and categorical type theory. (These are distinct traditions and we borrow from each.)

Standard references for models of simple type theories are [LS86, Cro93]. Usually we have a category (or better, a 2-category) of categories equipped with certain structure. These categories are equivalent (in a sense which needs to be made precise) to type theories of a certain kind, and so can be identified with type theories.

Typically we are interested in one particular type theory T and so in the corresponding category $\mathbb{T}$ constructed from its syntax. The perspective of categorical logic is that models of T in an arbitrary category $\mathbb{C}$ of the sort in question are given by structure preserving functors $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ from the *classifying category* $\mathbb{T}$ to $\mathbb{C}$. (These matters are explained carefully in [Cro93] where $\mathbb{T}$ is called *generic.*)

Notations and conventions We shall write the composition of maps $f : A \longrightarrow B$ and $g : B \longrightarrow C$ as $f; g : A \longrightarrow C$. We stress at once that we shall take a relaxed attitude towards notation. In principle we can distinguish between

(i) the syntax of some type theory T,

(ii) the interpretation of the syntax in the (syntactic) classifying category $\mathbb{T}$, and

(iii) the interpretation of the syntax in some arbitrary model $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$.

In categorical type theory, (i) and (ii) may harmlessly be identified; but the interpretation of syntax in some specific model is usually indicated by semantic brackets (see [Cro93]). However we prefer to overload notation by dropping the semantics brackets and allow the context to disambiguate what we write. Thus we shall systematically describe maps in our semantic categories using a mixed syntax consisting of the syntax of our type theory (PCF) augmented by names for individual objects and maps in the model. (Or we can think of the syntax of $\mathbb{C}$ in the sense of categorical type theory.) Our convention will be to let a term t of type B with free variables (amongst the) $x_1, \cdots, x_n$ of types $A_1, \cdots, A_n$ respectively denote a map

$$t \quad : \quad A_1 \times \cdots \times A_n \quad \longrightarrow \quad B.$$

We develop a theory of models for PCF along the general lines of categorical type theory. Some of the material is quite routine, but there are a number of points of interest; and we take the opportunity to recast the standard notions of denotational semantics in the more general framework.

Extensionality and order-extensionality There are a couple of items which we may as well make precise now. We assume for the purpose of this discussion that our categories are equipped with a terminal object $\mathbf{1}$, and that the global sections functor is appropriately thought of as giving the elements of types. (So we set aside models of linear logic.) In general such categories will not be concrete in the natural way; that is, the global sections functor will not be faithful. When it is, that is when

$$f = g : A \longrightarrow B \qquad \Longleftrightarrow \qquad \forall a : \mathbf{1} \longrightarrow A . a; f = a; g : \mathbf{1} \longrightarrow B,$$

we say that the model is *extensional.* Similarly in the common order-enriched situation we may ask whether the global sections functor regarded as an **Poset**-enriched functor to the enriching category **Poset** (which is enriched over itself) is faithful. When it is, that is when

$$f \leqslant g : A \longrightarrow B \qquad \Longleftrightarrow \qquad \forall a : \mathbf{1} \longrightarrow A . a; f \leqslant a; g : \mathbf{1} \longrightarrow B,$$

we say that the model is *order-extensional.* Category theorists often talk of the category (enriched category) having enough points.

Computational soundness and adequacy Like the standard domain theoretic models, categorical models of a functional programming language are static: they are essentially models of equational theories. In particular we shall model the programming language of PCF given in §1.1 by modelling the equational theory from §1.2. Thus the question arises of what should be the equational theory associated with a programming language.

An operational semantics for a (typed) functional programming language typically provides:

- a distinguished collection of program types P;
- for each program type P a distinguished collection $\mathcal{V}_P$ of (closed) terms $v : P$ called values;
- for each program type P a relation of convergence to value $s \Downarrow v$ between arbitrary (closed) terms $s : P$ and values $v : P$.

As usual we write $s\Downarrow$ for $\exists v. s \Downarrow v$. (Note that this outline encompasses untyped languages which can be regarded as having a single (program) type.)

Consider for the moment a model of T (of some unspecified kind); we write $[\![-]\!]$ for the interpretation function. In the model we should be able to distinguish a collection of values as the "elements" of the interpretation $[\![P]\!]$ of each program type P. Then the model is said to be

- *computationally sound* just when for any $s : P$, if $s\Downarrow$ then the interpretation $[\![s]\!]$ is a value in $[\![P]\!]$;
- *computationally adequate* just when for any $s : P$, if the interpretation $[\![s]\!]$ is a value in $[\![P]\!]$ then $s\Downarrow$.

These notions clearly admit stronger versions. We say that the model is

- *strongly computationally sound* just when $s \Downarrow v : P$ implies $[\![s]\!]$ is a value in $[\![P]\!]$ and that $[\![s]\!] = [\![v]\!] \in [\![P]\!]$;
- *strongly computationally adequate* just when $[\![s]\!]$ is a value in $[\![P]\!]$ implies $s \Downarrow v$ for some v with $[\![s]\!] = [\![v]\!] \in [\![P]\!]$.

Remark 2.1.1 (i) In practice models which are computationally sound and adequate are strongly so. (In the presence of suitable equality tests this will be automatic, but it holds in their absence.) Of course strong computational soundness and computational adequacy imply strong computational adequacy.

(ii) One might be tempted by a condition of the form

$$[\![s]\!] = [\![v]\!] \in [\![P]\!] \quad \Longrightarrow \quad s \Downarrow v : P.$$

But this fails for traditional models of lazy languages where abstractions are regarded as values.

Since the point of our models is that the process of computation should be seen as the replacement of equals by equals, we clearly want models to be strongly computationally sound. This can be ensured by insisting that the (initial, syntactic) classifying model is so, in other words by insisting that

$$s \Downarrow v : P \quad \Longrightarrow \quad s = v$$

in a corresponding equational theory. Also we should at least be able to consider computationally adequate models where computation to value is faithfully reflected. This requires that the classifying model be computationally adequate; in other words that $s = v$ in the equational theory implies $s \Downarrow v : P$. Thus the natural requirement on an equational theory associated with a programming language is that

$$(\dagger) \qquad s \Downarrow v : P \quad \Longleftrightarrow \quad s = v \quad \text{in the theory}$$

(In general there will be many theories satisfying this requirement.)

Now consider the specific case of PCF. It seems natural in view of §1.1 to regard PCF as having two program types ι, o. The values of type o are the booleans t, f and those of type ι are the numerals. So we see that Corollary1.2.2 says that the requirement (†) is satisfied in the case of PCF by the equational theory which we presented in §1.2. As a result the (initial, syntactic) classifying model $\mathbb{T}$ for PCF will be (strongly) computationally sound and adequate.

2.2 Platek categories

We start by establishing a very general categorical context for recursion theory. We adopt what we take to be Platek's original conceptions [Pla66] and make higher types, the conditional (or definition by cases) at all types and fixed points at all types the basis for our discussion.

Note that in contrast with the usual formulation of PCF, we take a conditional at all types as a basic rather than defined construction; this seems more natural from a semantic point of view and does not[9] entail a substantial change of the programming language or type theory.

Definition 2.2.1 A *Platek category* is a cartesian closed category $\mathbb{C}$ equipped with the following additional structure:

(i) *The conditional.* An object B, two maps $\mathsf{t} : \mathbf{1} \longrightarrow B$, $\mathsf{f} : \mathbf{1} \longrightarrow B$ and a family of maps

$$\gamma_A : B \times A \times A \quad \longrightarrow \quad A$$

for each object A of $\mathbb{C}$ with the property that

$$\begin{array}{ccccc} A \times A \cong \mathbf{1} \times A \times A & \xrightarrow{\mathsf{t} \times \mathbf{1} \times \mathbf{1}} & B \times A \times A & \xleftarrow{\mathsf{f} \times \mathbf{1} \times \mathbf{1}} & \mathbf{1} \times A \times A \cong A \times A \\ & {\scriptstyle \mathsf{fst}} \searrow & \downarrow {\scriptstyle \gamma_A} & \swarrow {\scriptstyle \mathsf{snd}} & \\ & & A & & \end{array}$$

commutes.

(ii) *Fixed points.* A family of maps for each object A of $\mathbb{C}$

$$\mathsf{Y}_A : A \Rightarrow A \quad \longrightarrow \quad A$$

with the property that the diagram

$$\begin{array}{ccc} (A \Rightarrow A) \times (A \Rightarrow A) & \xrightarrow{\mathbf{1} \times \mathsf{Y}_A} & (A \Rightarrow A) \times A \\ \uparrow {\scriptstyle \Delta} & & \downarrow {\scriptstyle \mathsf{ev}} \\ A \Rightarrow A & \xrightarrow[\mathsf{Y}_A]{} & A \end{array}$$

[9] To see this, consider PCF extended by a conditional cond^A at every type A. For terms s and t of type $A = (A_1, \cdots, A_n, \iota)$ and b of boolean type, we note that $\lambda x_1 : A_1 \cdots x_n : A_n.\mathsf{cond}^\iota b(s\vec{x})(t\vec{x})$ (where $\vec{x}$ are not free in s, t and b) is extensionally (and hence also observationally) equivalent to $\mathsf{cond}^A bst$.

commutes.

Remark 2.2.2 (i) γ_A interprets the conditional at type A and the commutative diagram gives the two usual equations:

$$\begin{aligned}\mathsf{cond}(\mathsf{t}, x, y) &= x \\ \mathsf{cond}(\mathsf{f}, x, y) &= y.\end{aligned}$$

Category theorists might expect to see the requirement that γ be a natural transformation which would give the naturality equation

$$h(\mathsf{cond}(b, x, y)) \quad = \quad \mathsf{cond}(b, h(x), h(y)).$$

However we do not need to insist on this as part of the general theory.

(ii) Y_A interprets the fixed-point operator at type A and the commutative diagram gives the standard fixed-point equation:

$$f(\mathsf{Y}_A(f)) \quad = \quad \mathsf{Y}_A(f).$$

In examples, we shall often have familiar properties of Y (dinaturality, Bekic-Scott property for products), but again we do not need to insist on them as part of the general theory. (Indeed we do not know a complete list of equational properties of the fixed-point operator in categories of domains.)

(iii) Note that we do not say that t and f are distinct. However if they are the same, then for every A in $\mathbb{C}$, the two projections $\mathsf{fst}, \mathsf{snd} : A \times A \longrightarrow A$ are identical. It follows at once that A is subterminal (the unique map $A \longrightarrow \mathbf{1}$ is monic). But the fixed-point operator provides at least one element (global section) for any A. So in case t and f are equal, the Platek category is equivalent to the (one-object-one-map) category $\mathbf{1}$.

Many of our Platek categories will be order-enriched and some will be enriched in some category of structured domains. The standard reference for enriched category theory is [Kel82]. We need to make clear what we mean by an enriched Platek category.

Definition 2.2.3 Suppose that $\mathbb{V}$ is a symmetric monoidal category. By a *Platek category enriched over* $\mathbb{V}$ we mean the following: a category $\mathbb{C}$ enriched over $\mathbb{V}$,

(i) which is cartesian closed in the enriched sense, so that the natural isomorphisms characterizing the products and function spaces in $\mathbb{C}$ are maps between the appropriate hom-objects in $\mathbb{V}$, and

(ii) whose underlying category is an ordinary Platek category.

Note that as things stand there is no interaction between the enrichment and the conditional or the fixed points.

Maps of Platek categories As maps between Platek categories we should take suitable structure preserving functors. We spell this out in the following definition.

Definition 2.2.4 Suppose $\mathbb{C}$ and $\mathbb{D}$ are Platek categories. A functor $F : \mathbb{C} \longrightarrow \mathbb{D}$ is a *map of Platek categories* (or just a *map* when the context is obvious) under the following conditions:

(i) F preserves products and function spaces in the usual up-to-isomorphism sense: the canonical maps

$$\begin{array}{rcl} F(\mathbf{1}) & \longrightarrow & \mathbf{1} \\ F(A \times B) & \longrightarrow & F(A) \times F(B) \end{array}$$

are isomorphisms, and the resulting canonical map

$$F(A \Rightarrow B) \quad \longrightarrow \quad F(A) \Rightarrow F(B)$$

is also an isomorphism.

(ii) F preserves the conditional in the sense that the canonical map

$$B \quad \longrightarrow \quad F(B)$$

is an isomorphism. (It follows that modulo isomorphism, $\gamma_{F(A)}$ is $F(\gamma_A)$ and so on.)

(iii) F preserves fixed points in the sense that

$$\begin{tikzcd} F(A) \Rightarrow F(A) \arrow[r, "\mathsf{Y}_{F(A)}"] & F(A) \arrow[d, equal] \\ F(A \Rightarrow A) \arrow[u, "\cong"] \arrow[r, "F(\mathsf{Y}_A)"'] & F(A) \end{tikzcd}$$

commutes.

The basic setting for our categorical semantics is the category of Platek categories and maps (of Platek categories). Of course the usual notion of natural transformation make this naturally a 2-category, but for the most part we are able to suppress this.

For completeness, we make clear what we mean by a map of Platek categories in the enriched setting.

Definition 2.2.5 Suppose that $\mathbb{C}$ and $\mathbb{D}$ are Platek categories enriched over the symmetric monoidal category $\mathbb{V}$. A *map of enriched Platek categories* $F : \mathbb{C} \longrightarrow \mathbb{D}$ (or just map when the context is obvious) is an enriched functor $F : \mathbb{C} \longrightarrow \mathbb{D}$ whose underlying ordinary functor is a map of ordinary Platek categories.

2.3 Models of PCF

In the previous subsection, we defined structure on a category which models the basic processes of definition (typed λ-calculus, conditionals and fixed points) in PCF, but we have yet to consider how to model the arithmetical structure on the basic data type of individuals. We choose to regard this as a question of a different kind: we treat ι separately from o. Note that o has a dual role: it is a data type but it is first introduced to give a basic recursion-theoretic construction, the conditional. One equational theory for arithmetic was presented in [Sco93], but here we concentrate on the categorical interpretation of the weak equational theory of §1.2 which reflects Plotkin's operational semantics.

As a preliminary consider a category $\mathbb{C}$ with terminal object $\mathbf{1}$. Suppose that we have an object N of $\mathbb{C}$ equipped with maps

$$\mathbf{1} \xrightarrow{0} N \qquad\qquad N \xrightarrow{s} N$$

Then (overloading notation) we can define maps $n : \mathbf{1} \longrightarrow N$ for n a natural number inductively: the map $0 : \mathbf{1} \longrightarrow N$ is already given and we set $n+1 : \mathbf{1} \longrightarrow N$ equal to the composite

$$\mathbf{1} \xrightarrow{n} N \xrightarrow{s} N$$

We refer to the maps $n : \mathbf{1} \longrightarrow N$ so defined as *numerals*. Note that we do not assume that numerals $n : \mathbf{1} \longrightarrow N$ and $m : \mathbf{1} \longrightarrow N$ with $m \neq n$ are distinct maps in $\mathbb{C}$.

We can of course now regard any $f : N \longrightarrow N$ in $\mathbb{C}$ as "defining" a numerical function $F : \mathbb{N} \longrightarrow \mathbb{N}$; but the usual context is a category with products.

Definition 2.3.1 Suppose that $\mathbb{C}$ is a category with finite products and that N is an object of $\mathbb{C}$ equipped with maps

$$\mathbf{1} \xrightarrow{0} N \qquad\qquad N \xrightarrow{s} N.$$

Take numerals $n : \mathbf{1} \longrightarrow N$ as just defined. A map $f : N^k \longrightarrow N$ in $\mathbb{C}$ *numeralwise represents* (or numeralwise *expresses*) the numerical function $F : \mathbb{N}^k \longrightarrow \mathbb{N}$ just when the composite

$$\mathbf{1} \xrightarrow{(n_1, \cdots, n_k)} N^k \xrightarrow{f} N$$

is equal to $F(n_1, \cdots, n_k) : \mathbf{1} \longrightarrow N$ for all natural numbers $n_1, \cdots, n_k$.

Suppose also that B is an object of $\mathbb{C}$ equipped with maps

$$\mathbf{1} \xrightarrow{\mathsf{t}} B \qquad\qquad \mathbf{1} \xrightarrow{\mathsf{f}} B.$$

A map $r : N^k \longrightarrow B$ *numeralwise represents* the k-ary relation R on $\mathbb{N}$ just when the composite

$$\mathbf{1} \xrightarrow{(n_1, \cdots, n_k)} N^k \xrightarrow{r} B$$

is $\mathsf{t} : \mathbf{1} \longrightarrow B$ for all $\langle n_1, \cdots, n_k \rangle \in R$ and $\mathsf{f} : \mathbf{1} \longrightarrow B$ for all $\langle n_1, \cdots, n_k \rangle \notin R$.

With these standard ideas as background we give the categorical counterpart to the arithmetical equations of §1.2.

Definition 2.3.2 Suppose that $\mathbb{C}$ is a category with finite products (a terminal object suffices) equipped with a diagram

$$\begin{tikzcd} \mathbf{1} \arrow[r, shift left, "\mathsf{t}"] \arrow[r, shift right, "\mathsf{f}"'] & B. \end{tikzcd}$$

Consider an object N of $\mathbb{C}$ equipped with maps

$$\mathbf{1} \xrightarrow{0} N \qquad N \xrightarrow{s} N \qquad N \xrightarrow{p} N \qquad N \xrightarrow{z} B.$$

We say that N so equipped is a *simple object of numerals* (relative to $\mathbf{1} \overset{\mathsf{t}}{\underset{\mathsf{f}}{\rightrightarrows}} B$) just when the diagrams

$$\begin{tikzcd} \mathbf{1} \arrow[r, "0"] \arrow[dr, "0"'] & N \arrow[d, "p"] \\ & N \end{tikzcd} \qquad \begin{tikzcd} \mathbf{1} \arrow[r, "n+1"] \arrow[dr, "n"'] & N \arrow[d, "p"] \\ & N \end{tikzcd}$$

$$\begin{tikzcd} \mathbf{1} \arrow[r, "0"] \arrow[dr, "\mathsf{t}"'] & N \arrow[d, "z"] \\ & B \end{tikzcd} \qquad \begin{tikzcd} \mathbf{1} \arrow[r, "n+1"] \arrow[dr, "\mathsf{f}"'] & N \arrow[d, "z"] \\ & B \end{tikzcd}$$

commute. (Here of course the $n : \mathbf{1} \longrightarrow N$ are the numerals derived from $0 : \mathbf{1} \longrightarrow N$ and $s : N \longrightarrow N$.)

This definition is weak in two respects. First it only provides information about the behaviour of the numerals $n : \mathbf{1} \longrightarrow N$. Two of the diagrams say that the standard predecessor function is numeralwise represented by p; while the other two say that z numeralwise represents a test for zero. We hope to suggest this focus on numeralwise representability by speaking of an object of *numerals* rather than *natural numbers*. Of course the arithmetical equations of §1.2 are concerned only with numerals and this reflects the fact that the only values in the operational semantics are numerals. The second way in which the definition is weak is that it is purely equational and makes no references to the possibility of any recursion. We intend to suggest this feature by the qualification "simple". This simplicity is part of the charm of Plotkin's reduction rules [Plo77] giving the operational semantics.

In general the force of the definition of simple object of numerals (as of the corresponding equations) will be extremely weak. Very few functions need be numeralwise representable. (Think for example about what can be explicitly defined in the case $B = N$, $\mathsf{t} = 0$ and $\mathsf{f} = 1$.) However in the context of Platek categories the definition is strong as we shall explain in the next subsection.

We can now describe our notion of a categorical model for PCF. We give two definitions the first of which is very simple.

Definition 2.3.3 (Concrete version) A categorical model for PCF is a Platek category equipped with a simple object of numerals.

We hope that it is clear how the equational theory of PCF can be modelled in such a structure. Note in passing that a given Platek category may contain many non-isomorphic simple objects of numerals; so the choice of such is definitely part of the structure.

Remark 2.3.4 For some purposes it is useful to look at the definition from the point of view of categorical logic. Let $\mathbb{T}$ be a cartesian closed category generated by the equational theory of PCF. (There are a number of equivalent ways to construct $\mathbb{T}$. For example one might augment PCF with a terminal type and products and take equivalence classes of terms in the manner described by Lambek and Scott [LS86] and by Crole [Cro93]; or else consider formal products of types and equivalence classes of tuples of terms.) $\mathbb{T}$ is a Platek category and it contains by construction a simple object of numerals which we may as well continue to call ι. So $\mathbb{T}$ is as one would hope a model for PCF in the sense just given.

Definition 2.3.5 (Abstract version) A categorical model for PCF is a map $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ of Platek categories.

The point is that since the notion of a simple object of numerals is purely equational, $\mathcal{M}(\iota)$ is a simple object of numerals in $\mathbb{C}$. All that the map $\mathcal{M}$ does effectively is to pick out this structure and so the two definitions are essentially equivalent. The second definition has the virtue of making clear the sense in which $\mathbb{T}$ (or the identity $\mathbb{T} \longrightarrow \mathbb{T}$) is initial amongst models for PCF. Usually we shall simply write $\mathbb{C}$ for a model of PCF, letting the structure in the sense of Definition 2.3.3 or the structure and interpreting functor $\mathcal{M}$ in the sense of Definition 2.3.5 be understood.

We can justify this definition in terms of the discussion in §2.1. The values in PCF are the booleans of type o and numerals of type ι. Thus Corollary 1.2.2 says in effect that the initial model $\mathbb{T}$ of PCF is computationally sound and adequate (in the strong sense). Hence all our models of PCF are (strongly) computationally sound. (This is not a serious restriction.) Computational adequacy on the other hand has a number of characterizations.

Proposition 2.3.6 *Let $\mathbb{C}$ ($\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$) a be a model of* PCF. *Then the following are equivalent:*

(i) *$\mathbb{C}$ is computationally adequate.*

(ii) *$\mathbb{C}$ is strongly computationally adequate.*

(iii) *If $\mathcal{M}(s : \mathbf{1} \longrightarrow \iota)$ (respectively $\mathcal{M}(s : \mathbf{1} \longrightarrow o)$) is a numeral (respectively boolean) in $\mathbb{C}$ then s is a numeral (respectively boolean) in $\mathbb{T}$.*

(iv) *If $\mathcal{M}(s) = \mathcal{M}(n)$ (respectively $\mathcal{M}(s) = \mathcal{M}(\mathsf{t}), \mathcal{M}(s) = \mathcal{M}(\mathsf{f})$) in $\mathbb{C}$ then $s = n$ (respectively $s = \mathsf{t}, s = \mathsf{f}$) in $\mathbb{T}$.* $\square$

Since conditions (iii) and (iv) make no reference to the operational semantics, they suggest the following general definition. Suppose that a type theory $\mathbb{T}$ is equipped with program types P and sets of values $\mathcal{U}_P \subseteq \mathbb{T}(\mathbf{1}, P)$. We assume that any model $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ is equipped with values $\mathcal{V}_{\mathcal{M}(P)} \subseteq \mathbb{C}(\mathbf{1}, \mathcal{M}(P))$ so that $\mathcal{M}(\mathcal{U}_P) \subseteq \mathcal{V}_{\mathcal{M}(P)}$. Thus $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ is a strongly sound model of $\mathbb{T}$.

Definition 2.3.7 Let $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ be a strongly sound model of the type theory $\mathbb{T}$. We shall say that $\mathbb{C}$ (strictly speaking $\mathcal{M}$) is *adequate* just when $\mathcal{M}$ reflect values: if $\mathcal{M}(s)$ is in $\mathcal{V}_P$ then s is in $\mathcal{U}_P$.

In the case of PCF we take as values in any model the booleans and numerals. Then the above definition applies to give a generalization of the usual notion of adequacy. Not all models of PCF are adequate. In particular the *trivial model* of PCF, the unique model $\mathcal{M} : \mathbb{T} \longrightarrow \mathbf{1}$ of PCF in the (terminal) one-object-one-map category $\mathbf{1}$ is certainly not adequate. (But adequacy can fail in more subtle ways.)

In an adequate model of PCF the elements (*i.e.* global sections) of B and N may be quite bizarre. Usually however we are interested in models in which the inidividual values are distinct and in which there is just one additional non-value at each program type. This is the familiar notion of a standard model of PCF. We present this in our general setting constructively and in both a set-based and an order-enriched version.

Definition 2.3.8 Let $\mathbb{C}$ be a model for PCF and write $\mathcal{V}_P$ for the values in $\mathbb{C}$ of the program type P (either booleans or numerals).

Set-based case. We say that $\mathbb{C}$ is *standard* just when

(i) the individual values in $\mathcal{V}_P$ are distinct, and

(ii) for all $a, b : \mathbf{1} \longrightarrow P$ in $\mathbb{C}$, $a = b$ in $\mathbb{C}$ if and only if $a \in \mathcal{V}_P \iff b \in \mathcal{V}_P$.

Order-enriched case. We say that the order-enriched model $\mathbb{C}$ is *standard* just when

(i) the individual values in $\mathcal{V}_P$ are distinct, and

(ii) for all $a, b : \mathbf{1} \longrightarrow P$ in $\mathbb{C}$, $a \leqslant b$ in $\mathbb{C}$ if and only if $a \in \mathcal{V}_P \implies b \in \mathcal{V}_P$.

There is one further desirable property of models of PCF which we need to consider. First we make a general definition.

Definition 2.3.9 Suppose that $\mathbb{C}$ is a category with products and that we have a collection of program types P in $\mathbb{C}$ and sets of values $\mathcal{V}_P \subseteq \mathbb{C}(\mathbf{1}, P)$. We call a map $f : P_1 \times \cdots \times P_k \longrightarrow P$ from a product of program types $P_1, \cdots, P_k$ to a program type P *a first order map.* We say that such a map f is *(elementwise) strict in its i-th argument* just when for any $a_1 : \mathbf{1} \longrightarrow P_1, \cdots, a_k : \mathbf{1} \longrightarrow P_k$, if the composite

$$\mathbf{1} \xrightarrow{(a_1, \cdots, a_k)} P_1 \times \cdots \times P_k \xrightarrow{\ f\ } P$$

is a value, then so is $a_i : \mathbf{1} \longrightarrow P_i$. And we say that f is *strict* (without qualification) just when it is strict in all its arguments. (Note that this definition depends on the product representation $P_1 \times \cdots \times P_k$.)

We shall be interested in models for PCF in which there are appropriate strict maps.

Definition 2.3.10 A model $\mathbb{C}$ of PCF is *strict* just when the structural maps satisfy the following natural strictness conditions:

- $\gamma_N : B \times N \times N \longrightarrow N$ and $\gamma_B : B \times B \times B \longrightarrow B$ are both strict in their first argument;
- $s : N \longrightarrow N, p : N \longrightarrow N$ and $z : N \longrightarrow B$ are all strict.

Example 2.3.11 (i) The initial model $\mathbb{T}$ of PCF is strict. (This follows from Corollary 1.2.2 and the nature of the evaluation relations. For example if $\mathsf{cond}ust = n : \iota$ then $\mathsf{cond}ust \Downarrow n$; but $\Downarrow$ corresponds to a deterministic evaluation strategy so we deduce that $u\Downarrow$ as required.)

(ii) The standard Scott domain model for PCF is strict. (This is clear from its definition but see Proposition 2.4.11 below.)

(iii) The trivial model $\mathbf{1}$ is strict.

We leave the reader to ruminate on non-strict models (the obvious ones are rather boring) and simply give some sufficient conditions for strictness.

Proposition 2.3.12 *Suppose that a model $\mathbb{C}$ of* PCF *is such that the functor $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ maps $\mathbb{T}(\mathbf{1}, \iota)$ surjectively onto $\mathbb{C}(\mathbf{1}, N)$ and $\mathbb{C}(\mathbf{1}, o)$ surjectively onto $\mathbb{C}(\mathbf{1}, B)$. If $\mathbb{C}$ is adequate then $\mathbb{C}$ is strict.*

Proof Suppose for example that $a : \mathbf{1} \longrightarrow N$ in $\mathbb{C}$ is such that

$$\mathbf{1} \xrightarrow{a} N \xrightarrow{s} N$$

is a numeral. By surjectivity $a = \mathcal{M}(u)$ for some $u : \mathbf{1} \longrightarrow \iota$ in $\mathbb{T}$. Now we have

$$\mathcal{M}(u); \mathcal{M}(\mathsf{succ}) \quad = \quad \mathcal{M}(u; \mathsf{succ})$$

a numeral. As $\mathbb{C}$ is adequate $\mathcal{M}$ reflects numerals and so $u; \mathsf{succ}$ is a numeral in $\mathbb{T}$. But succ is strict in $\mathbb{T}$ (see above) so u is a numeral. Thus $a = \mathcal{M}(u)$ is a numeral as required. The other cases are similar. $\square$

Corollary 2.3.13 *Suppose that $\mathbb{C}$ is a standard model of* PCF *Then $\mathbb{C}$ adequate implies $\mathbb{C}$ strict.*

Proof We show that the functor $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ satisfies the surjectivity hypotheses of the preceding Proposition. It suffices to show that some element of $\mathbb{T}(\mathbf{1}, \iota)$ (respectively $\mathbb{T}(\mathbf{1}, o)$) maps to a non-value in $\mathbb{C}(\mathbf{1}, N)$ (respectively $\mathbb{C}(\mathbf{1}, B)$). In $\mathbb{T}(\mathbf{1}, \iota)$ consider $\mathbf{Y}(\mathsf{succ})$. If $\mathbf{Y}(s) = \mathcal{M}(\mathbf{Y}(\mathsf{succ})) = n$ in $\mathbb{C}$, then $n = n + 1$ in $\mathbb{C}$ and $\mathbb{C}$ is equivalent to the one-object-one-map category $\mathbf{1}$ and so not standard; thus $\mathbf{Y}(s)$ is the "undefined" element of $\mathbb{C}(\mathbf{1}, N)$. A similar argument using the fixed point of a map $\mathsf{swap} : o \longrightarrow o$ deals with the other case. The result now follows from Proposition 2.3.12. $\square$

2.4 Recursion theory

We start by considering the following definition which (like that of simple objects of numerals) deals only with numerals.

Definition 2.4.1 Suppose $\mathbb{C}$ is a category with finite products (for the moment a terminal object suffices). An object N of $\mathbb{C}$ equipped with maps

$$\mathbf{1} \xrightarrow{0} N \qquad\qquad N \xrightarrow{s} N$$

is an *iterative object of numerals* just when it comes equipped for any object X and maps $a : \mathbf{1} \longrightarrow X$, $f : X \longrightarrow X$ with a choice of maps $r = \mathsf{it}(a, f) : N \longrightarrow X$ such that the diagrams

$$\begin{tikzcd} \mathbf{1} \arrow[r, "0"] \arrow[dr, "a"'] & N \arrow[d, "r"] \\ & X \end{tikzcd} \qquad\qquad \begin{tikzcd} \mathbf{1} \arrow[r, "n+1 = n; s"] \arrow[d, "n; r"'] & N \arrow[d, "r"] \\ X \arrow[r, "f"] & X \end{tikzcd}$$

commute.

In other words an iterative object of numerals is one which enables us to represent iterations numeralwise. Now a standard argument shows that in a category with products we can also numeralwise represent recursion: that is, we can give for any object X and maps $a : \mathbf{1} \longrightarrow X$, $g : N \times X \longrightarrow X$ a choice of maps $r = \mathsf{rec}(a, g) : N \longrightarrow X$ such that the diagrams

$$\begin{tikzcd} \mathbf{1} \arrow[r, "0"] \arrow[dr, "a"'] & N \arrow[d, "r"] \\ & X \end{tikzcd} \qquad \begin{tikzcd} \mathbf{1} \arrow[r, "n+1=n;s"] \arrow[d, "(n,n;r)"'] & N \arrow[d, "r"] \\ N \times X \arrow[r, "g"] & X \end{tikzcd}$$

commute.

We remark in passing on the strength of this definition in a cartesian closed category. Suppose that $\mathbf{1} \xrightarrow{0} N \xrightarrow{s} N$ is an iterative object of numerals in a cartesian closed category $\mathbb{C}$. The closure enables us to parametrize recursive definitions. So a sequence of primitive recursive definitions based on 0 and s gives rise to a sequence of maps in $\mathbb{C}$ which numeralwise satisfy the corresponding recursion equations. Rather than spell this out in detail we give a formulation as in [LS86] where the result is stated for the (stronger) notion of weak natural number object.

Proposition 2.4.2 *If $\mathbf{1} \xrightarrow{0} N \xrightarrow{s} N$ is an iterative object of numerals in a cartesian closed category $\mathbb{C}$ then all primitive recursive functions are numeralwise representable in $\mathbb{C}$.* □

Remark 2.4.3 (i) Lambek and Scott presented the corresponding result for a weak natural number object in [LS86]. The approach is essentially to observe that a weak natural number object is an iterative object of numerals and then to follow the standard approach outlined above.

(ii) Recall that we did not assume that our numerals $n : \mathbf{1} \longrightarrow N$ were all distinct. However, in the context of an iterative object of numerals in a cartesian closed category $\mathbb{C}$, it is easy to show the following analogue of Remark 2.2.2(iii): if $n : \mathbf{1} \longrightarrow N$ and $m : \mathbf{1} \longrightarrow N$ are equal for $n \neq m$, then the category $\mathbb{C}$ is equivalent to the one-object-one-map category $\mathbf{1}$.

Let us return to the observation that in the presence of products, iteration entails recursion (at the level of numeralwise representation). The standard recursion equations for predecessor involve no parameters, so predecessor can be represented numeralwise; also a test for zero can be defined by iteration. Hence we have the following trivial result.

Proposition 2.4.4 *Suppose that the category $\mathbb{C}$ with products is equipped with a diagram* $\begin{tikzcd} \mathbf{1} \arrow[r, shift left, "\mathsf{t}"] \arrow[r, shift right, "\mathsf{f}"'] & B \end{tikzcd}$. *Then an iterative object of numerals $\mathbf{1} \xrightarrow{0} N \xrightarrow{s} N$ can be further equipped to give a simple object of numerals (relative to B).* □

Recursion in a Platek category is provided in a powerful way by fixed points. Hence it is not surprising that in a model of PCF we have a converse to the above.

Proposition 2.4.5 *In a model of* PCF *the structure $\mathbf{1} \xrightarrow{0} N \xrightarrow{s} N$ is (or can be equipped with the structure of) an iterative object of numerals.*

Proof Given $a : \mathbf{1} \longrightarrow X$ and $f : X \longrightarrow X$ we define $r : N \longrightarrow X$ implicitly by the informal equation

$$r(n) \quad = \quad \mathsf{if}\ (n = 0)\ \mathsf{then}\ a\ \mathsf{else}\ f(r(p(n))),$$

using the fixed-point operator and check that it works. □

Remark 2.4.6 It follows that the requirements on a standard model that the values be distinct is essentially the requirement that there be a non-value.

We have seen that in a model of PCF the notion of a simple object of numerals provides an algebraic way of describing an iterative object of numerals. So by Proposition 2.4.2, it provides numeralwise representations for all primitive recursive functions. Note that this representation is quite uniform: there is a representation in $\mathbb{T}$ whose interpretation in a model is a representation there. In fact all partial recursive functions can similarly be represented in a sense which we now make clear.

Definition 2.4.7 Suppose that $\mathbb{C}$ is a category with finite products and that N is an object of $\mathbb{C}$ equipped with maps $0 : \mathbf{1} \longrightarrow N$ and $s : N \longrightarrow N$. Take numerals $n : \mathbf{1} \longrightarrow N$ as usual. A map $f : N^k \longrightarrow N$ in $\mathbb{C}$ *tracks* a partial numerical function $\phi : \mathbb{N}^k \longrightarrow \mathbb{N}$ just when for any natural numbers $n_1, \cdots, n_k$ with $\phi(n_1, \cdots, n_k)$ defined, the composite $\mathbf{1} \xrightarrow{(n_1, \cdots, n_k)} N^k \xrightarrow{\quad f \quad} N$ is equal to $\mathbf{1} \xrightarrow{\phi(n_1, \cdots, n_k)} N$; and *numeralwise represents* ϕ just if, in addition, whenever the composite $\mathbf{1} \xrightarrow{(n_1, \cdots, n_k)} N^k \xrightarrow{\quad f \quad} N$ is a numeral, then $\phi(n_1, \cdots, n_k)$ is indeed defined.

Since PCF can be regarded as a programming language its terms should represent effective functions.

Proposition 2.4.8 *If $f : \iota^k \Rightarrow \iota$ is a term of* PCF *then (the interpretation of) f numeralwise represents a partial recursive function in the initial model $\mathbb{T}$.*

Proof The relation $t \Downarrow v$ is defined inductively and so is semi-recursive. The result follows as by Corollary 1.2.2, $f(n_1, \cdots, n_k) = m$ if and only if $f(n_1, \cdots, n_k) \Downarrow m$. $\Box$

A standard piece of programming gives a partial converse.

Proposition 2.4.9 *For every partial recursive function $\phi : \mathbb{N}^k \longrightarrow \mathbb{N}$, there is a term $f : \iota^k \Rightarrow \iota$ whose interpretation tracks ϕ in the initial model $\mathbb{T}$.*

Proof The argument is standard. The collection of partial functions tracked in a model is clearly closed under substitution (composition). Hence it suffices (in view of Kleene's representation of the partial recursive functions) to show how the result of applying the least number operator may be tracked. We give the simplest case. Suppose that $h : \iota \times \iota \Rightarrow \iota$ tracks (and so numeralwise represents) the total function $H : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$. Define $g : \iota \times \iota \Rightarrow \iota$ implicitly by the informal equation

$$g(n, k) \quad = \quad \mathsf{if}\ h(n, k) = 0\ \mathsf{then}\ k\ \mathsf{else}\ g(n, k+1)$$

using the least fixed-point operator. Then $f : \iota \Rightarrow \iota$ defined by $f(n) = g(n, 0)$ tracks the possibly partial function $\phi(n) = \mu k.(H(n, k) = 0)$. $\Box$

Corollary 2.4.10 *If $\mathbb{C}$ is a model of* PCF *then for every partial recursive $\phi : \mathbb{N}^k \longrightarrow \mathbb{N}$ there is a term f of* PCF *whose interpretation in $\mathbb{C}$ tracks ϕ.* $\Box$

To get the full converse we need additional arguments. We first note the following general property of strict models.

Proposition 2.4.11 *Suppose $\mathbb{C}$ is a strict model of* PCF*. Then for any $f : N^k \longrightarrow N$ in $\mathbb{C}$ there is a strict $g : N^k \longrightarrow N$ which represents the same partial function (and similarly for arbitrary first order maps).*

Proof Define $t : N \longrightarrow N$ by

$$t(x) \quad \stackrel{\text{def}}{=} \quad \mathsf{cond}(\mathsf{zero?}x)00.$$

We see that t is strict and carries all numerals to 0. Now define g by

$$g(x_1, \cdots, x_k) \stackrel{\text{def}}{=} \mathsf{cond}(\mathsf{zero?}t(x_1))(\mathsf{cond}(\mathsf{zero?}t(x_2))(\mathsf{cond} \cdots (\mathsf{cond}(\mathsf{zero?}t(x_k))f(x_1, \cdots, x_k)\Omega) \cdots) \cdots)\Omega.$$

$\square$

Corollary 2.4.12 *Suppose that $\mathbb{C}$ is a strict model of* PCF. *Then the collection of partial functions numeralwise represented in $\mathbb{C}$ is closed under substitution (composition).*

Proof The essential point is the following. If $f_1, \cdots, f_k : N^l \longrightarrow N$ numeralwise represent $\phi_1, \cdots, \phi_k : \mathbb{N}^l \longrightarrow \mathbb{N}$ and if $g : N^k \longrightarrow N$ is strict and numeralwise represents $\psi : \mathbb{N}^k \longrightarrow \mathbb{N}$, then $g(f_1, \cdots, f_k) : N^l \longrightarrow N$ numeralwise represents $\psi(\phi_1, \cdots, \phi_k) : \mathbb{N}^l \longrightarrow \mathbb{N}$. $\square$

The full converse to Proposition 2.4.8 gives the following characterization.

Theorem 2.4.13 *The partial functions numeralwise representable in the initial model $\mathbb{T}$ for* PCF *are exactly the partial recursive functions.*

Proof It simply remains to refine the proof of Proposition 2.4.9. As $\mathbb{T}$ is strict, Corollary 2.4.12 means that it suffices to show that the least number operator preserves numerlwise representability. But this is a straightforward consequence of the fact that $\mathbf{Y}$ behaves syntactically like a least fixed-point operator:

$$\mathsf{if}\ (\mathbf{Y}t)(s_1, \cdots, s_k) \Downarrow v \text{ then } t^r\Omega(s_1, \cdots, s_k) \Downarrow v \text{ for some } r.$$

In the notation of Proposition 2.4.9 we deduce by a straightforward induction that $g(n, k)$ numeralwise represents the function

$$\psi(n, k) \quad = \quad \mu l.(l \geqslant k \ \ \& \ \ H(n, l) = 0),$$

and so f numeralwise represents ϕ as required. $\square$

Remark 2.4.14 (i) This result was of course known to Platek [Pla66] and Scott [Sco93]. We sketch it here to show how the proof appears in our general perspective.

(ii) One can extract further information from the PCF definability of the least number operator. Suppose we vary PCF by omitting the predecessor but including an equality test $N \times N \longrightarrow B$. Then we can define a function numeralwise representing the predecessor function

$$p(x) \quad = \quad \mathsf{if}\ x = 0\ \mathsf{then}\ 0\ \mathsf{else}\ \mu y.\mathsf{succ} y = x.$$

However the predecessor *cannot* be recovered from (sucessor and) test for zero alone.

We note some elementary observations:

Lemma 2.4.15 *(i) If $f : (\underbrace{\iota, \cdots, \iota}_{k}, \iota)$ is a term of* PCF *then in the initial model $\mathbb{T}$ (the interpretation of) f numeralwise represents a partial recursive function.*

(ii) If f tracks ϕ in $\mathbb{T}$ then f tracks ϕ in any model of PCF. *If f numeralwise represents ϕ in $\mathbb{T}$ then f numeralwise represents ϕ in any adequate model of* PCF.

Proof (i) is Obvious in view of the effective nature of the reduction relation $\Downarrow$. We omit proof of (ii). $\square$

Finally we can say something about the representability of partial recursive functions in adequate models of PCF.

Proposition 2.4.16 *If $\mathbb{C}$ is an adequate model of* PCF *then for every partial recursive function $\phi : \mathbb{N}^k \longrightarrow \mathbb{N}$ there is a term f of* PCF *(in the sense of Remark 2.3.4) whose interpretation in $\mathbb{C}$ numeralwise represents ϕ.*

Proof If f numeralwise represents ϕ in $\mathbb{T}$ then it does so in $\mathbb{C}$ as the functor $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ reflects numerals. □

The converse of the preceding Proposition is essentially obvious:

Proposition 2.4.17 *Suppose that $\mathbb{C}$ is an adequate model of* PCF*. Then any term $f : \iota^k \longrightarrow \iota$ of* PCF *represents a partial recursive function in $\mathbb{C}$.* □

3 Observables, adequacy, observational and full abstraction

3.1 Observables, observational preorder and quotient

We start by considering a notion of observational equivalence in a general categorical setting. Throughout this section we take $\mathbb{C}$ to be a symmetric monoidal closed category, which we think of as some category of types and terms. (We suppress the structure of associativities and so on.)

Definition 3.1.1 A *notion of observables* $\mathcal{O}$ on $\mathbb{C}$ associates to each object A of $\mathbb{C}$ a set $\mathcal{O}_A$ of subsets of $\mathbb{C}(I, A)$ called *observables* at A, with the property that if $f : A \longrightarrow B$ in $\mathbb{C}$ and $S \in \mathcal{O}_B$ then

$$f^*S \quad \stackrel{\text{def}}{=} \quad \{\, a : I \longrightarrow A \mid a; f \in S \,\} \quad \in \quad \mathcal{O}_A.$$

We say that such an association $A \mapsto \mathcal{O}_A$ *equips* $\mathbb{C}$ *with observables* and that $\mathbb{C}$ so equipped is a *category with observables*.

Example 3.1.2 (i) Suppose that T is a programming language with an operational semantics as considered at the end of the last section; and suppose that $\mathbb{T}$ is the category of types and (equivalence classes of) terms for a corresponding type theory. We assume that $\mathbb{T}$ is computationally sound and adequate (in the strong sense) so that convergence to value is preserved and reflected by equalities in the type theory. Hence we do not bother to distinguish between types and terms in T and the objects and maps in $\mathbb{T}$ which are their respective denotations. Then we have relations

- of convergence $a\Downarrow$ for $a : I \longrightarrow P$ (P a program type)
- of convergence to value $a \Downarrow v$ for $a : I \longrightarrow P$, $v : I \longrightarrow P$ (P a program type and v the interpretation of a value)

on maps in $\mathbb{T}$. (Of course, $a \Downarrow v$ is just a suggestive way of writing "$a = v : I \longrightarrow P$ is the interpretation of a value"!)

Take for simplicity a simple program type P. (The generalization to more than one program type is straightforward.) For $f : A \longrightarrow P$ we define an observation

$$O_f \quad \stackrel{\text{def}}{=} \quad \{\, a : I \longrightarrow A \mid a; f\Downarrow \,\}$$

and for $f : A \longrightarrow P$ and $v : I \longrightarrow P$ a value we define

$$O_{f,v} = \{\, a : I \longrightarrow A \mid a; f \Downarrow v \,\}.$$

Now we give some notions of observables.

(i) *Termination.* The association $A \mapsto \mathcal{O}_A = \{\, O_f \mid f : A \longrightarrow P \,\}$ equips $\mathbb{T}$ with observables.

(ii) *Termination to value.* The association $A \mapsto \mathcal{O}_A = \{\, O_{f,v} \mid f : A \longrightarrow P, v : I \longrightarrow P \text{ a value} \,\}$ equips $\mathbb{T}$ with observables.

(iii) *Termination to specified value.* Choose a value $u : P$, so that we have a distinguished map $u : I \longrightarrow P$ in $\mathbb{T}$. The association $A \mapsto \mathcal{O}_A = \{\, O_{f,u} \mid f : A \longrightarrow P \,\}$ equips $\mathbb{T}$ with observables.

(ii) More generally suppose given a monoidal closed category $\mathbb{C}$, an object P (a program type) of $\mathbb{C}$ and a collection $\mathcal{V}$ of "elements" $v : I \longrightarrow P$ of P (a set of values). Then for $f : A \longrightarrow P$ we define an observation:

$$O_f \quad \stackrel{\text{def}}{=} \quad \{\, a : I \longrightarrow A \mid a; f \in \mathcal{V} \,\},$$

and for $f : A \longrightarrow P$ and $v : I \longrightarrow P \in \mathcal{V}$ we define an observation

$$O_{f,v} \quad \stackrel{\text{def}}{=} \quad \{\, a : I \longrightarrow A \mid a; f = v \,\}.$$

We generalise the notion of observables above.

(i) *Termination.* The association $A \mapsto \mathcal{O}_A = \{\, O_f \mid f : A \longrightarrow P \,\}$ equips $\mathbb{C}$ with observables.

(ii) *Termination to value.* The association $A \mapsto \mathcal{O}_A = \{\, O_{f,v} \mid f : A \longrightarrow P, v : I \longrightarrow P \in \mathcal{V} \,\}$ equips $\mathbb{C}$ with observables.

(iii) *Termination to specified value.* Choose a distinguished map $u : I \longrightarrow P \in \mathcal{V}$. The association $A \mapsto \mathcal{O}_A = \{\, O_{f,u} \mid f : A \longrightarrow P \,\}$ equips $\mathbb{C}$ with observables.

(iii) Suppose $\mathcal{O}$ equips $\mathbb{D}$ with observables and $F : \mathbb{C} \longrightarrow \mathbb{D}$ is a functor. Then it is easy to see that the association

$$A \quad \mapsto \quad (F^{-1}\mathcal{O})_A \quad = \quad \{\, F^{-1}(u) \mid u \in \mathcal{O}_A \,\}$$

equips $\mathbb{C}$ with observables.

Remark 3.1.3 (i) The reader may like to consider how very different are the notions of observables *termination*, *termination to value* and *termination to specified value* in the case of (models of) lazy languages where abstractions are values: only termination seems to correspond to a clear computational intuition in this case. However for PCF as we shall shortly see the notions will coincide in reasonable circumstances.

(ii) The construction in **(iii)** is particularly revealing in the case of a model $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ of a programming language where $\mathbb{C}$ is equipped with a notion $\mathcal{O}$ of observables as in **(ii)**. (We naturally assume that the interpretation of values in $\mathbb{T}$ is identical with those maps in $\mathbb{T}$ which become values in $\mathbb{C}$.) In this case $\mathcal{M}^{-1}\mathcal{O}$ is a notion of observables in $\mathbb{T}$ which generally will not coincide with any notion defined as in **(i)**. The structure in $\mathbb{C}$ allows us to make additional observations in $\mathbb{T}$.

It seems just worth introducing some suggestive terminology to describe special properties of notions of observables $\mathcal{O}$.

Definition 3.1.4 Suppose that $\mathbb{C}$ is a (symmetric monoidal closed) category with observables $\mathcal{O}$.

(i) We say that $U \in \mathcal{O}_G$ is a *universal observation* just when

$$\mathcal{O}_A \quad = \quad \{\, f^*U \mid f : A \longrightarrow G \,\}$$

for all A.

(ii) We say that a set l of observations is a *generating set of observations* for $\mathcal{O}$ just when

$$\mathcal{O}_A \quad = \quad \{\, f^*R \mid f : A \longrightarrow C, \quad R \in \mathcal{O}_C \text{ is in } l \,\}$$

for all A.

(iii) We say that an "element" $d : I \longrightarrow D$ is a *detector for observations* just when $\{\, d \,\} \in \mathcal{O}_D$ is a universal observation. (We also say d can be used to detect observation.)

Consider the examples in **(i)** and **(ii)** above. In the case of *termination* there is a universal observation. In the case of *termination to specified value*, the value can be used to detect observations. In the case of *termination to value* the collection $\{\, O_{\mathsf{id},v} \mid v : I \longrightarrow P \,\}$ is a generating set of observations where $\mathsf{id} : P \longrightarrow P$ is the identity.

Observational preorder Suppose that $\mathbb{C}$ is a (symmetric monoidal closed) category with observables $A \mapsto \mathcal{O}_A$. Then there is a natural notion of *observational preorder* between maps from the same hom-set. For $f, g : A \longrightarrow B$ in $\mathbb{C}$ we define

$$f \lesssim g \quad \stackrel{\text{def}}{=} \quad \overline{f} \in R \implies \overline{g} \in R \quad \text{for all } R \in \mathcal{O}_{A \multimap B}$$

where $\overline{f}, \overline{g} : I \longrightarrow (A \multimap B)$ are obtained from f, g respectively by transposing. We write the associated equivalence as $\simeq$. (The reader will see that at last the symmetric monoidal closed assumption is beginning to be used.)

Composition on either side preserves this preorder. Note that maps $h : B \longrightarrow C$ and $k : D \longrightarrow A$ give rise to obvious map $(A \multimap h) : (A \multimap B) \longrightarrow (A \multimap C)$ and $(k \multimap B) : (A \multimap B) \longrightarrow (D \multimap B)$. Suppose that $f \lesssim g : A \longrightarrow B$. Take $R \in \mathcal{O}_{A \multimap C}$;

$$\overline{f;h} \in R \quad \iff \quad \overline{f} \in (A \multimap h)^*R \quad \implies \quad \overline{g} \in (A \multimap h)^*R \quad \iff \quad \overline{g;h} \in R.$$

Thus $f;h \lesssim g;h : A \longrightarrow C$. Similarly take $S \in \mathcal{O}_{D \multimap B}$; suppose that $f \lesssim g : A \longrightarrow B$. Take $R \in \mathcal{O}_{A \multimap C}$;

$$\overline{k;f} \in S \quad \iff \quad \overline{f} \in (k \multimap B)^*S \quad \implies \quad \overline{g} \in (k \multimap B)^*S \quad \iff \quad \overline{k;g} \in S.$$

Thus $k;f \lesssim k;g : D \longrightarrow A$.

Definition 3.1.5 If we let $\hat{\mathbb{C}}(A,B) \stackrel{\text{def}}{=} \mathbb{C}(A,B)/\lesssim$ be the poset induced by the preorder $\lesssim$ on $\mathbb{C}(A,B)$, we get a new order-enriched category $\hat{\mathbb{C}}$, which we refer to as the *observational quotient* of $\mathbb{C}$. (Of course $\hat{\mathbb{C}}$ depends on the choice of observables $\mathcal{O}$.)

The category $\hat{\mathbb{C}}$ inherits a symmetric monoidal closed structure (now as an order-enriched category) from $\mathbb{C}$. Note that a map $g : C \longrightarrow D$ induces an obvious map

$$(A \multimap B) \xrightarrow{\lambda} (A \otimes C \multimap B \otimes D),$$

the transpose of

$$(A \multimap B) \otimes A \otimes C \xrightarrow{\mathrm{ev} \otimes g} B \otimes D$$

Suppose that $f \lesssim f' : A \longrightarrow B$. Take $R \in \mathcal{O}_{A \otimes C \multimap B \otimes D}$.

$$\overline{f \otimes g} \in R \quad \Longleftrightarrow \quad \overline{f} \in \lambda^* R \quad \Longrightarrow \quad \overline{f'} \in \lambda^* R \quad \Longleftrightarrow \quad \overline{f' \otimes g} \in R.$$

Thus $f \otimes g \lesssim f' \otimes g : A \otimes C \longrightarrow B \otimes D$. It follows at once that $f \lesssim f' : A \longrightarrow B$ and $g \lesssim g' : C \longrightarrow D$ entail $f \otimes g \lesssim f' \otimes g' : A \otimes C \longrightarrow B \otimes D$. Thus $\otimes$ becomes an order-enriched functor on $\hat{\mathbb{C}}$ and the symmetric monoidal structure carries over. This closed structure does likewise as $\hat{\mathbb{C}}(A \otimes B, C)$ and $\hat{\mathbb{C}}(A, B \multimap C)$ are isomorphic posets trivially by the definition.

Now suppose that $\mathbb{C}$ is a cartesian closed category so that we are dealing with a categorical product $\times$ and corresponding function space $\Rightarrow$. Then $\hat{\mathbb{C}}$ is also cartesian closed. It is enough to show that $\times$ is a categorical product in $\hat{\mathbb{C}}$. Note that a map $g : C \longrightarrow B$ gives rise to a map

$$\gamma \quad : \quad (C \Rightarrow A) \cong (C \Rightarrow A) \times \mathbf{1} \xrightarrow{\mathbf{1} \times \overline{g}} (C \Rightarrow A) \times (C \Rightarrow B) \cong (C \Rightarrow A \times B).$$

Suppose that $f \lesssim f' : C \longrightarrow A$. Take $R \in \mathcal{O}_{C \Rightarrow A \times B}$;

$$\overline{(f, g)} \in R \quad \Longleftrightarrow \quad \overline{f} \in \gamma^* R \quad \Longrightarrow \quad \overline{f'} \in \gamma^* R \quad \Longleftrightarrow \quad \overline{(f', g)} \in R.$$

Thus $(f, g) \lesssim (f', g) : C \longrightarrow A \times B$. It follows at once that $f \lesssim f' : C \longrightarrow A$ and $g \lesssim g' : C \longrightarrow B$ entail $(f, g) \lesssim (f', g') : C \longrightarrow A \times B$. The converse implication is easy and so $\times$ is a product in $\hat{\mathbb{C}}$ in the order-enriched sense.

To summarise the discussion so far, we have shown:

Proposition 3.1.6 *For any symmetric monoidal closed category $\mathbb{C}$ with observables, the observational quotient $\hat{\mathbb{C}}$ is an order-enriched category which inherits the symmetric monoidal closed structure from $\mathbb{C}$. (That is, the quotient functor preserves the structure.) If $\mathbb{C}$ is in fact cartesian closed, then the same structure is likewise inherited by $\hat{\mathbb{C}}$.* □

Remark 3.1.7 (i) Naturally different notions of observables may give rise to the same notion of observational quotient. Indeed suppose $\mathcal{O}$ and $\mathcal{O}'$ are two notions of observables on the same (symmetric monoidal closed) category $\mathbb{C}$, giving rise to preorders $\lesssim_1$ and $\lesssim_2$ respectively. Then one easily sees that

$$f \lesssim_1 g \quad \Longrightarrow \quad f \lesssim_2 g$$

holds generally just when for every $\overline{f}$,

$$\bigcap \{ R \in \mathcal{O}_1 \mid \overline{f} \in R \} \quad \subseteq \quad \bigcap \{ R \in \mathcal{O}_2 \mid \overline{f} \in R \}.$$

(ii) Generally suppose $\mathbb{C}$ is a cartesian closed category enriched over CPOs, and that it is equipped with a notion of observables. (For example the category $\mathbb{CA}$ of computational arenas and innocent strategies which is the subject matter of Part III.) We call the enriching partial order the *given ordering* of $\mathbb{C}$. It is easy to see that if for each A, every observable $R \in \mathcal{O}_A$ is upper-closed with respect to the given ordering of $\mathbb{C}(\mathbf{1}, A)$, then the given ordering is contained in the associated observational preorder. (This is the case for $\mathbb{CA}$.)

3.2 Observables: the case of PCF

Recall from §2.4 that we regard PCF as having two program types o, ι. The values of type o are the booleans and those of type ι are the numerals. Thus in $\mathbb{T}$ we have values $\mathtt{t}, \mathtt{f} : \mathbf{1} \longrightarrow o$ and $n : \mathbf{1} \longrightarrow \iota$ for each natural number n. In a model $\mathbb{C}$ of PCF it remains natural to take the booleans and numerals in $\mathbb{C}$ as values. (They are just the images of values in $\mathbb{T}$.) Thus in any model $\mathbb{C}$ of PCF we have notions of observables along the lines of Example 3.1.2(ii). In principle we can distinguish *nine separate notions.* We have

- termination
- termination to value
- termination to specified value

and we may take these as

- at N only
- at B only
- at both N and B.

We now show that in good circumstances these distinct notions coincide.

We start by considering the general situation. Recall from Example 3.1.2**(ii)** that our different notions of observables are defined in terms of observations:

$$\begin{array}{rcl} O_f & \stackrel{\text{def}}{=} & \{\, a : I \longrightarrow A \mid a; f \in \mathcal{V}_P \,\} \\ O_{f,v} & \stackrel{\text{def}}{=} & \{\, a : I \longrightarrow A \mid a; f = v \,\} \end{array}$$

for $f : I \longrightarrow P$ where P is a program type and $v \in \mathcal{V}_P$ a value. Now suppose P and Q are program types. If $v \in \mathcal{V}_Q$ we write $k_v : P \longrightarrow Q$ for a strict map carrying all values to the constant value v. if $u \in \mathcal{V}_P$ and $v \in \mathcal{V}_Q$ we write $l_{u,v}$ for a strict map which carries just the value u to a value, that value being v. When such maps exist we get connexions between observations:

$$\begin{array}{rcl} O_{f,u} & = & O_{(f;l_{u,v}),v} \\ O_{f,u} & = & O_{f;l_{u,u}} \\ O_f & = & O_{(f;k_v),v}. \end{array}$$

In other words the different kinds of observations are interchangeable.

By Proposition 2.4.11 we are in this good position in strict models of PCF.

Proposition 3.2.1 *If $\mathbb{C}$ is a strict model for* PCF *then the nine separate notions of observables (introduced at the start of the subsection) coincide.* □

In the non-strict situation it seems best to make a choice. We shall take as the *standard notion of observables* for PCF that of termination at both ground types. In general unless we say otherwise this is the one we shall mean; and for an arbitrary model $\mathbb{C}$ of PCF we shall write $\hat{\mathbb{C}}$ for the quotient category with respect to this notion. But in case $\mathbb{C}$ is strict all reasonable choices give the same result.

We note in passing the following simple fact about the observational quotient of models of PCF.

Proposition 3.2.2 *Suppose that $\mathbb{C}$ is a model of* PCF*; then $n \leqslant s$ in $\hat{\mathbb{C}}$ entails $n = s$ in $\mathbb{C}$. In particular*

(i) $\mathbb{C}$ *adequate* $\iff$ $\hat{\mathbb{C}}$ *adequate*

(ii) $\mathbb{C}$ *standard* $\Longrightarrow$ $\hat{\mathbb{C}}$ *standard*

(iii) $\mathbb{C}$ *strict* $\iff$ $\hat{\mathbb{C}}$ *strict.* □

(Of course the second implication is not reversible; $\mathbb{T}$ is not standard but $\hat{\mathbb{T}}$ is.) A consequence is the following simple property of the observational preorder.

Corollary 3.2.3 *Suppose that in a model of* PCF *the maps $f, g : N^k \longrightarrow N$ numeralwise represent the partial functions $\phi, \psi : \mathbb{N}^k \longrightarrow \mathbb{N}$ respectively. Then $f \lesssim g$ entails $\phi \subseteq \psi$ (i.e. ϕ extends ψ).*

Proof Suppose $\phi(n_1, \cdots, n_k) = m$. Then in $\mathbb{C}$ we have

$$m = (n_1, \cdots, n_k); f \quad \lesssim \quad (n_1, \cdots, n_k); g.$$

We deduce $m = (n_1, \cdots, n_k); g$ in $\mathbb{C}$ and so $\psi(n_1, \cdots, n_k) = m$. □

3.3 Behavioural preorders, order-extensionality and context lemma

We now turn to the standard notion(s) of observational preorders defined concretely over terms of a programming language. We restrict attention to PCF though much of the discussion has wider application. Let s and t be closed terms of type A. Recall that s is said to approximate t observationally if $C[s]\Downarrow$ implies $C[t]\Downarrow$ for every type-compatible context $C[X]$ such that $C[s]$ and $C[t]$ are programs. Suppose $x_1 : A_1, \cdots, x_n : A_n \vdash s, t : B$, and let θ range over *closing substitutions* i.e. type-preserving functions from variables to closed PCF-terms. There are several ways by which this notion of behavioural preorder may be extended to a preorder on open terms.

- closure by context: $s \lesssim t \stackrel{\text{def}}{=}$ if $C[s]\Downarrow$ then $C[t]\Downarrow$ for all type-compatible contexts $C[X]$ such that both $C[s]$ and $C[t]$ are programs
- closure by abstraction: $s \lesssim^o t \stackrel{\text{def}}{=}$ if $C[\lambda \vec{x} : \vec{A}.s]\Downarrow$ then $C[\lambda \vec{x} : \vec{A}.t]\Downarrow$ for all contexts $X : A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B \vdash C[X]$ of program type
- closure by substitution: $s \lesssim^s t \stackrel{\text{def}}{=}$ if $C[s_\theta]\Downarrow$ then $C[t_\theta]\Downarrow$ for all closing substitutions θ and contexts $X : B \vdash C[X]$ of program type.

Remark 3.3.1 Each of the preorders may be regarded as an appropriate definition (for different reasons).

(i) The first preorder $\lesssim$ is the observational preorder with respect to which inequationally full abstraction is standardly defined. Note that free variables in s (and similarly t) are bound as a result of the context substitution $C[s]$.

(ii) The second preorder $\lesssim^o$ (the superscript "o" is for closure) corresponds precisely to the observational preorder of the PCF type theory $\mathbb{T}$ with respect to the notion of observables as defined in Example 3.1.2.

(iii) The third preorder $\lesssim^s$ (the superscript "s" is for substitution) was studied in [AO93] in the context of the lazy λ-calculus.

We could have defined $\lesssim$ (and similarly for the other two preorders) as:

$$C[s] \Downarrow v \implies C[t] \Downarrow v, \quad \text{for all values } v;$$

but this is equivalent to the simpler formulation given earlier. For if for some $C[X]$, we have $C[s] \Downarrow v$ and $C[t] \Downarrow v'$ where v and v' are distinct values, take $D[X]$ to be $\mathsf{cond}(\mathsf{eq}C[X]v)0\Omega$. Then $D[s]\Downarrow$ and $D[t]\Uparrow$. (This is a concrete proof of Proposition 3.2.1 for the strict initial model $\mathbb{T}$ of PCF.)

What is the relationship between the three preorders? Restricted to closed terms, it is clear that they are equivalent. For open terms, it is easy to see that $\lesssim$ implies $\lesssim^o$ and $\lesssim^s$ since the effects of closure and closing substitution respectively can be simulated (by an appeal to Proposition 1.2.1) by appropriate contexts. But in fact the three preorders coincide even for open terms.

Lemma 3.3.2 *The preorder $\lesssim^o$ is contained in the preorder $\lesssim$.*

Proof Consider PCF-terms $y_1 : B_1, \cdots, y_m : B_m \vdash s, t : A$. Take a context $X : A \vdash C[X]$ such that $C[s]$ and $C[t]$ are both programs. For any fresh variable $z : E$ where E is $B_1 \Rightarrow \cdots \Rightarrow B_m \Rightarrow A$, $(\lambda z : E.C[z\vec{y}])(\lambda \vec{y} : \vec{B}.s) = C[s]$ is an equation in the type theory $\mathbb{T}$. Therefore, by Proposition 1.2.1, if $C[s]\Downarrow$ then $(\lambda z : E.C[z\vec{y}])(\lambda \vec{y} : \vec{B}.s)\Downarrow$. Suppose $s \lesssim t$; take $D[X] \equiv (\lambda z : E.C[z\vec{y}])(\lambda \vec{y} : \vec{B}.X)$, then $(\lambda z : E.C[z\vec{y}])(\lambda \vec{y} : \vec{B}.t)\Downarrow$, and so, $C[t]\Downarrow$ by Proposition 1.2.1. $\square$

Remark 3.3.3 The above simple syntactic argument is really quite general.

Lemma 3.3.4 *The preorder $\lesssim^s$ is contained in the preorder $\lesssim^o$.*

Proof This is a simple application of Milner's context lemma. Suppose $s \lesssim^s t$. Then for all closing substitutions θ we have $s_\theta \lesssim t_\theta$, and hence for all closed terms $\vec{a}$ of types $\vec{A}$ we have

$$(\lambda \vec{x}.s)\vec{a} \quad \lesssim \quad (\lambda \vec{x}.t)\vec{a}.$$

But now by the context lemma it follows that $\lambda \vec{x}.s \lesssim \lambda \vec{x}.t$, and so, $s \lesssim^o t$. $\square$

Hence we can conclude:

Proposition 3.3.5 *As preorders over open terms, $\lesssim$, $\lesssim^o$ and $\lesssim^s$ are equivalent.* $\square$

Context lemma and order-extensionality

To our knowledge, the first context lemma (or Operational Extensionality Theorem as Meyer calls it in [MC88]) was proved by Milner in [Mil77]. Since then, several results of a similar kind but for different languages have been proved; see e.g. the work of Berry [Ber81], Curien [Cur93], Stoughton [Sto88], Howe [How89] and Abramsky and Ong [AO93, Ong92] *etc.*

Adapting Meyer's terminology, it seems reasonable to say that in a (symmetric monoidal closed) category $\mathbb{C}$ equipped with a notion of observables, the *observational extensionality theorem is valid* just in case the induced observational preorder $\lesssim$ satisfies the following:

$$f \lesssim g : A \longrightarrow B \quad \iff \quad \forall a : \mathbf{1} \longrightarrow A. a; f \lesssim a; g : \mathbf{1} \longrightarrow B.$$

This is equivalent to the condition that the global sections functor from $\hat{\mathbb{C}}$ to the enriching category of posets is faithful, that is, to the condition that the order-enriched category is order-extensional.

Suppose now that the notion of observables $\mathcal{O}$ on $\mathbb{C}$ is that based on termination at program type. Then the observational preorder $\lesssim$ is just the (analogue of the) preorder $\sqsubsetsim^o$ described above for the case of PCF. But quite generally $\sqsubsetsim^o$ coincides with the contextual preorder $\sqsubsetsim$. Hence in these circumstances we refer to the observational extensionality theorem as the context lemma. This is consistent with the usual definitions. Curien in [Cur93, p. 324] defines the context lemma (in the case of PCF) as the following property: for any *closed* terms s and t of the same type $A = (A_1, \cdots, A_n, \iota)$ say,

$$s \sqsubsetsim t \qquad \Longleftrightarrow \qquad su_1 \cdots u_n \Downarrow v \implies tu_1 \cdots u_n \Downarrow v \quad \text{for any value } v \text{ and any } u_i : A_i.$$

By an easy inductive argument, the context lemma is equivalent to:

$$s \sqsubsetsim^\xi t \qquad \Longleftrightarrow \qquad su_1 \sqsubsetsim^\xi tu_1 \quad \text{for any } u_1 : A_1,$$

where $\sqsubsetsim^\xi$ may be any of the three behavioural preorders we have just considered.

If the context lemma (or observational extensionality theorem) is valid in a model of PCF, we can exploit it to good effect. As a simple example we give a converse to Corollary 3.2.3.

Proposition 3.3.6 *Suppose that the context lemma is valid in a model $\mathbb{C}$ for* PCF *and that the observational quotient $\hat{\mathbb{C}}$ is standard. Take $f, g : N^k \longrightarrow N$ in $\mathbb{C}$ numeralwise representing the partial functions $\phi, \psi : \mathbb{N}^k \longrightarrow \mathbb{N}$ respectively, and suppose that f is strict. Then we have*

$$f \lesssim g \qquad \Longleftrightarrow \qquad \phi \subseteq \psi.$$

Proof By Corollary 3.2.3 we only need the converse implication, so we assume $\phi \subseteq \psi$. By the context lemma, it suffices to show

$$\begin{aligned} f(a_1, \cdots, a_k) &\lesssim g(a_1, \cdots, a_k) \\ (a_1, \cdots, a_k); f &\lesssim (a_1, \cdots, a_k); g \end{aligned}$$

for all $a : \mathbf{1} \longrightarrow N^k$ in $\mathbb{C}$. As $\hat{\mathbb{C}}$ is standard it suffices to show that $(a_1, \cdots, a_k); f$ a numeral implies $(a_1, \cdots, a_k); g$ a numeral. As f is strict we know that $(a_1, \cdots, a_k); f$ a numeral, b say, implies $a_1, \cdots, a_k$ are numerals. We deduce that $\phi(a_1, \cdots, a_k) = b$ and so as $\phi \subseteq \psi$, $\psi(a_1, \cdots, a_k) = b$. But then $(a_1, \cdots, a_k); g = b$ as required. □

3.4 Adequacy, observational and full abstraction

We give an account of the notions of adequacy and full abstraction in the general framework we have introduced.

Definition 3.4.1 Suppose that $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ is a model of the type theory $\mathbb{T}$ and that $\mathbb{T}$ is equipped with a notion of observables $\mathcal{O}$.

(i) $\mathbb{C}$ is *adequate* just when for any $R \in \mathcal{O}_A$ we have

(†) $s \in R$ and $\mathcal{M}(s) = \mathcal{M}(t)$ implies $t \in R$.

(ii) Suppose further $\mathbb{C}$ is order-enriched. $\mathbb{C}$ is *order-adequate* just when for any $R \in \mathcal{O}_A$ we have

($\dagger$) $s \in R$ and $\mathcal{M}(s) \leqslant \mathcal{M}(t)$ implies $t \in R$.

The connexions with standard notions of adequacy is quite straightforward. Note that condition ($\dagger$) for a generating set of observations is sufficient to ensure adequacy. Consider the three notions of observables discussed in Example 3.1.2.

We assume that values in $\mathbb{C}$ are such that u is a value in $\mathbb{T}$ if and only if $\mathcal{M}(u)$ is a value in $\mathbb{C}$.

- In case $\mathcal{O}$ is termination, $\mathbb{C}$ is adequate if and only if $\mathcal{M}(s)$ a value in $\mathbb{C}$ implies s a value in $\mathbb{T}$.
- In case $\mathcal{O}$ is termination to value, $\mathbb{C}$ is adequate if and only if $\mathcal{M}(s) = \mathcal{M}(u)$ a value in $\mathbb{C}$ implies $s = u$ in $\mathbb{T}$.

The first of these is the notion generally taken as standard notion of adequacy. Note that the idea of order-adequacy is neglected for the good reason that one never seems to consider model $\mathbb{C}$ where $\mathcal{M}(s)$ is greater than but not equal to a value.

We can give an easy alternative characterization of our notion of adequacy which is familiar in the case of the usual notion.

Proposition 3.4.2 *Suppose that $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ and that $\mathcal{O}$ is a notion of observables on $\mathbb{T}$.*

(i) $\mathbb{C}$ is adequate if and only if for all $s, t : A \longrightarrow B$

$$\mathcal{M}(s) = \mathcal{M}(t) \quad \Longrightarrow \quad s \simeq t.$$

(ii) Suppose that $\mathbb{C}$ is order-enriched. $\mathbb{C}$ is order-adequate if and only if for all $s, t : A \longrightarrow B$

$$\mathcal{M}(s) \leqslant \mathcal{M}(t) \quad \Longrightarrow \quad s \lesssim t.$$

The notion of full abstraction also makes sense at this level of generality.

Definition 3.4.3 Suppose that $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ is a model of the type theory $\mathbb{T}$ and that $\mathbb{T}$ is equipped with a notion of observables $\mathcal{O}$.

(i) $\mathbb{C}$ is *equationally fully abstract* just when for all $s, t : A \longrightarrow B$

$$\mathcal{M}(s) = \mathcal{M}(t) \quad \Longleftrightarrow \quad s \simeq t.$$

(ii) Suppose further that $\mathbb{C}$ is order-enriched. $\mathbb{C}$ is *(order) fully abstract* just when

$$\mathcal{M}(s) \leqslant \mathcal{M}(t) \quad \Longleftrightarrow \quad s \lesssim t$$

for all $s, t : A \longrightarrow B$. This notion is often called *inequational* full abstraction.

It is clear that this is a simple generalization of the standard notion.

Finally we introduce the notion which is fundamental to our treatment of PCF.

Definition 3.4.4 Suppose that $\mathcal{M} : \mathbb{T} \longrightarrow \mathbb{C}$ is a model of a type theory $\mathbb{T}$. Then $\mathbb{C}$ is *observationally abstract* just when for all $s, t : A \longrightarrow B$

$$\mathcal{M}(s) \lesssim \mathcal{M}(t) \quad \Longleftrightarrow \quad s \lesssim t.$$

Thus in the given circumstances $\mathbb{C}$ is *observationally abstract* just when the composite

$$\mathbb{T} \longrightarrow \mathbb{C} \longrightarrow \hat{\mathbb{C}}$$

is fully abstract.

One way to think of observational abstraction is as follows. If $\mathbb{C}$ is observationally abstract, then the contexts in $\mathbb{C}$ allow us to make no more distinctions between PCF-definable maps than do the contexts in $\mathbb{T}$. So $\mathbb{T} \longrightarrow \mathbb{C}$ induces induces (an order-embedding) $\hat{\mathbb{T}} \longrightarrow \hat{\mathbb{C}}$.

Part II

Dialogue games and innocent strategies

4 Dialogue games over computational arenas

Dialogue games are played by two players in a prescribed setting or environment called a *computational arena*. The dialogue game playable in a given computational arena is completely determined by the associated game tree. We specify a game tree in two stages:

- First the *computational arena* spells out the moves (which are questions and answers) of the game and the justification ordering between question-moves.
- The game tree is then systematically generated from the set of moves subject to a number of ground rules. Formally the game tree is represented as the collection of all paths in the tree. Such paths are called *legal positions*.

4.1 An approach based on dialogue games

Dialogue games are two-person games. The two players are called Player (or P) and Opponent (or O). In diagrams we represent Player's move as the hollow circle "$\circ$", and Opponent's move as the filled circle "$\bullet$". A dialogue game is played in a computational arena which sets out the moves of the game. There are four kinds of moves: Player's question which we represent generically as "(", Opponent's answer ")", Opponent's question"[" and Player's answer "]". The representation of questions and answers as left and right matching parentheses respectively reflects the following convention: Player's question can only be answered by Opponent, and *vice versa*. In addition every answer is associated with a unique question.

Not all question-moves are necessarily available at the start of the game. Some of them may become available or enabled as the play progresses. Except for the initial questions (which do not need any justification), a question-move can only be made provided its unique justifying (or enabling) move has been made. This notion of justification is formulated as a partial ordering between questions so that the resultant partially ordered set is an upside-down forest.

Definition 4.1.1 A *computational arena* A consists of the following data:

- A partially ordered set of questions $\langle\, \mathsf{Qn}(A), \leqslant_A \,\rangle$ such that the upper set of each question is a finite linear order. So the questions form an upside down forest (of trees), the root of each tree being a maximal element in the ordering.
- An association to each question of a set of possible answers. This is represented as a map $\mathsf{qn}_A : \mathsf{Ans}(A) \longrightarrow \mathsf{Qn}(A)$ where $\mathsf{Ans}(A)$ is the set of all answers of the arena A. An answer a is said to be an *appropriate answer* of the question $\mathsf{qn}_A(a)$.

Questions of depth 0, 2, 4, *etc.* are associated with Opponent (O). We refer to these questions as *O-questions*. Questions of depth 1, 3, 5, *etc.* are associated with Player (P), and we call them *P-questions*. Answers appropriate to an O-question are associated with Player, and they are called *P-answers*. Similarly answers appropriate to a P-question are associated with Opponent, and they are called *O-answers*. Questions of depth 0 (corresponding to the roots of trees) are called *initial* or *opening questions*, and they have a special status.

Let q and q' range over questions. We say that q' *justifies* q if q' is the unique question immediately above q in the ordering; that is to say, q' is the least question in $\mathsf{Qn}(A)$ such that $q \leqslant_A q'$ and $q \neq q'$. For the sake of uniformity, we shall also refer to the question $\mathsf{qn}_A(a)$ as the (unique) *justifying question* of the answer a.

Ground rules Given a computational arena, a play involving Player and Opponent observes the following rules:

- A play of a dialogue game always starts by Opponent asking an initial question.
- Thereafter the play alternates strictly between Player and Opponent. A play ends as soon as the initial question is answered.

Principles of Civil Conversation. Each play traces out a dialogue of questions and answers which obeys the following principles:

1. *Justification.* A question is asked only if the dialogue at that point "warrants" it in the sense that (an instance of) the unique justifying question is pending *i.e.* already asked but not yet answered. Likewise, an answer is proffered only if (an instance of) the unique question with which it is associated is pending.
2. *Priority.* Questions pending in a dialogue are answered on a "last-asked-first-answered" basis: the question which is last asked must be answered first. This is equivalent to Gandy's "no-dangling-question-mark condition".

Definition 4.1.2 Formally a *well-formed sequence* s of a computational arena A is a sequence of moves $m_1 \cdot m_2 \cdots m_n$ such that each move m_i is associated with a natural number μ_i called the *justification index* of m_i satisfying $\mu_i < i$ and conditions (w1) to (w4) in the following. By convention μ_1 is 0. The indices are best thought of as a way of representing justification pointers. Note that the preceding requirement $\mu_i < i$ means that the justification pointers always point backwards from m_i to m_{μ_i}. So a well-formed sequence s is by definition equipped with an auxiliary sequence of justification indices; both sequences are of the same length.

We say that a move m_j (which may be a question or an answer) is *explicitly justified* by the question m_i if m_i justifies m_j, and that the justification pointer at m_j points to m_i. We say that m_j is an *explicit answer* of the question m_i if m_j is an appropriate answer of m_i and that $\mu_j = i$.

(w1) *Initial question to start.* The first move m_1 in s is an initial question of A and there can be no occurrence of any initial question of A in the rest of s. By convention μ_1 is 0: an initial move is not justified by any move.

(w2) *Alternating play.* The sequence alternates between Player's move and Opponent's move.

(w3) *Explicit justification.* There are two cases:

- Any non-initial question may be asked if an instance of its unique justifying question has already been asked and has not been answered so far. More precisely for any non-initial question m_j in s, the move indexed by μ_j (which is m_{μ_j}) explicitly justifies m_j, and the segment $m_{\mu_j} \cdot m_{\mu_j+1} \cdots m_j$ of s does not contain any explicit answer of m_{μ_j}. Note that this means that for $\mu_j < k < j$, if m_k is an appropriate answer of the question m_{μ_j} then $\mu_k \neq \mu_j$; in fact it is a consequence of condition (w4) that $\mu_k > \mu_j$.
- Any answer a may be offered if an instance of its unique justifying question $\mathsf{qn}_A(a)$ has already been asked and has not been answered so far. More precisely any answer m_j in s explicitly answers the question m_{μ_j}, and that the segment $m_{\mu_j} \cdot m_{\mu_j+1} \cdots m_{j-1}$ of s does not contain any explicit answer of m_{μ_j}.

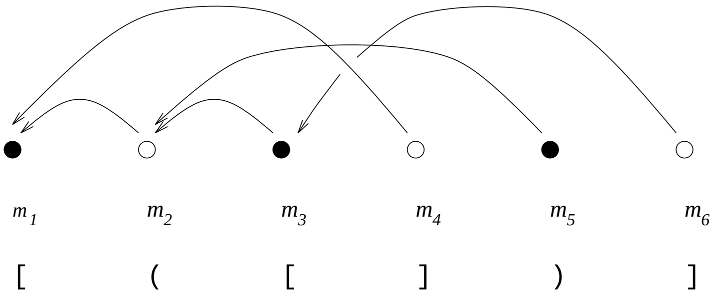

Figure 1: An example.

(w4) *"Last asked first answered"* (or "*no dangling question mark*"). Any sequence $s \equiv m_1 \cdot m_2 \cdots m_n$ satisfying the preceding three conditions is said to satisfy the "last asked first answered" condition if for any answer move m_i in s, the move m_{μ_i} which explicitly justifies m_i is the last unanswered question in $m_1 \cdot \cdots \cdot m_{i-1}$.

This condition is equivalent to Gandy's no-dangling-question-mark condition. We first introduce a definition. A question m occurring in a sequence t of moves equipped with an auxiliary sequence of justification indices is said to be *dangling in the sequence t* if t does not contain any explicit answer of m. Using the same notations as before, any sequence $s \equiv m_1 \cdot m_2 \cdots m_n$ is said to satisfy the *no-dangling-question-mark condition* if for every answer m_j occurring in the sequence s, the segment $m_{\mu_j} \cdot m_{\mu_j+1} \cdots m_j$ contains no dangling question in itself. Note that by condition (w3) the question m_{μ_j} is explicitly answered by m_j

The principle of priority is a version of the so called well-bracketing condition in formal language theory. There is a tradition in game semantics of intuitionistic logic which uses essentially the same condition, see *e.g.* Felscher's survey paper [Fel86].

For example, as shown in Figure 1, the sequence $m_1 \cdot m_2 \cdots m_6$ of shape $[\cdot(\cdot[\cdot]\cdot)\cdot]$ with the corresponding sequence of justification indices $0 \cdot 1 \cdot 2 \cdot 1 \cdot 2 \cdot 3$ violates the no dangling question mark condition (since (say) the segment $m_2 \cdots m_5$ has a dangling question m_3). It is easy to see that every initial subsequence of a well-formed sequence is well-formed.

Remark 4.1.3 (i) For any well-formed sequence $s \equiv m_1 \cdot m_2 \cdots m_n$, for each i, the justification index μ_i of m_i is a pointer from m_i to the move m_{μ_i} which explicitly justifies m_i, regardless of whether m_i is a question or an answer. The indices are a representation of pointers in terms of relative positions in the well-formed sequence s. Therefore whenever the well-formed sequence is altered or transformed in any way (say by removing some element), the auxiliary sequence of justification indices has to be systematically recalculated in order to preserve the original justification relationship. In the following we shall only be concerned with a particular kind of transformation of well-formed sequences called *projection*. We say that a sequence $s \equiv m_1 \cdot m_2 \cdots m_n$ *projects* to s' (or s' is a projection of s) if s' is obtained from s by deleting some elements from s; equivalently s' is a subsequence of s.

(ii) A projection of a well-formed sequence s to a subsequence s' *respects justification* if whenever a move m in s occurs in the subsequence s', so does the question explicitly justifying m. If this condition is satisfied, we can be sure that the image s' of the projection (after the indices have been systematically reset) satisfies condition (w3) of explicit justification.

(ii) Note that so long as m_{μ_j} is a non-initial question of a well-formed sequence, m_{μ_j} is in turn justified by the move m_α where α is the justification index μ_{μ_j} of m_{μ_j}. We can iterate this process thereby tracing out the history of explicitly justifying questions or simply the *history of justification* of m_j which must end with the only unjustified question of the well-formed sequence — the initial question m_1. Clearly for any non-initial move m_i, its history of justification is a unique subsequence of s. For any move m occurring in the history of justification of m', we say that m' is *hereditarily justified* by m. We state two elementary properties of well-formed sequences. The proof is straightforward and we omit it.

Lemma 4.1.4 *(i) In any initial subsequence of a well-formed sequence, the number of answers occurring in it is less than or equal to the number of questions.*

(ii) Any well-formed sequence whose last element is an explicit answer to the initial O-question is maximal. □

4.2 Views and legal positions

Definition 4.2.1 *Player's view*, or *P-view*, $\ulcorner p \urcorner$ of a well-formed sequence p of moves is defined recursively. Let q range over well-formed sequences of moves, and r over segments of well-formed sequences.

$$
\begin{array}{lcll}
\ulcorner [\urcorner & \stackrel{\mathrm{def}}{=} & [& \text{if “[” is initial,} \\
\ulcorner q \cdot (\cdot r \cdot [\urcorner & \stackrel{\mathrm{def}}{=} & \ulcorner q \urcorner \cdot (\cdot [& \text{if “(” explicitly justifies “[”,} \\
\ulcorner q \cdot) \urcorner & \stackrel{\mathrm{def}}{=} & \ulcorner q \urcorner \cdot) & \\
\ulcorner q \cdot [\cdot r \cdot] \urcorner & \stackrel{\mathrm{def}}{=} & \ulcorner q \urcorner & \text{if “]” explicitly answers “[”,} \\
\ulcorner q \cdot (\urcorner & \stackrel{\mathrm{def}}{=} & \ulcorner q \urcorner \cdot (. &
\end{array}
$$

Note that this definition is by recursion over the initial subsequences of a well-formed sequence. There is no ambiguity in the second clause: given an O-question “[” at the end of the sequence, there may well be several occurrences of the unique justifying question “(” of “[” in the sequence to the left of “[”; but by condition (w3) there is a pointer emanating from “[” indicating a specific instance of “(” which justifies it *explicitly*. For example the P-view of a well-formed sequence of moves may have the shape

$$[\cdot(\cdot)\cdot(\cdot)\cdot(\cdot[\cdot(\cdot)\cdot(\cdot)\cdot(\cdot[\cdot(\cdot)\cdots.$$

By construction whenever there is a pattern “$(\cdot[$” in a P-view, the O-question “[” is explicitly justified by the P-question “(”. Also there can be no segments of the form “$[\cdots]$” in a P-view. This may be read as the following: Player ignores answers to questions posed by Opponent.

There is a dual definition of *Opponent's view*, or *O-view*, $\llcorner p \lrcorner$ of a well-formed sequence p of moves:

$$
\begin{array}{lcll}
\llcorner q \cdot [\cdot r \cdot (\lrcorner & \stackrel{\mathrm{def}}{=} & \llcorner q \lrcorner \cdot [\cdot (& \text{if “[” explicitly justifies “(”,} \\
\llcorner q \cdot] \lrcorner & \stackrel{\mathrm{def}}{=} & \llcorner q \lrcorner \cdot] & \\
\llcorner q \cdot (\cdot r \cdot) \lrcorner & \stackrel{\mathrm{def}}{=} & \llcorner q \lrcorner & \text{if “)” explicitly answers “(”,} \\
\llcorner q \cdot [\lrcorner & \stackrel{\mathrm{def}}{=} & \llcorner q \lrcorner \cdot [. &
\end{array}
$$

The O-view of an empty sequence is the empty sequence. Since a well-formed sequence never begins with a P-question, we omit the case of $\llcorner (\lrcorner$. An O-view can never have a segment of the form $(\cdots)$:

$$[_1 \qquad (_2 \qquad [_3 \qquad (_4 \qquad [_5 \qquad (_6 \qquad)_7$$

Figure 2: Picture of a legal position.

Opponent ignores answers to questions posed by Player. An O-view may, for example, have the shape

$$[\cdot(\cdot[\cdot]\cdot[\cdot]\cdot[\cdot(\cdot[\cdot]\cdot[\cdot(\cdot\cdots.$$

The following properties of P-view and O-view are easy to verify:

- By repeated application of condition (w3), if $q\cdot(\cdot r\cdot)$ is a well-formed sequence and if "(" explicitly justifies ")", then
$$\ulcorner q\cdot(\cdot r\cdot)\urcorner \quad = \quad \ulcorner q\urcorner\cdot(\cdot).$$
Dually if $q\cdot[\cdot r\cdot]$ is a well-formed sequence and if "[" explicitly justifies "]", then
$$\llcorner q\cdot[\cdot r\cdot]\lrcorner \quad = \quad \llcorner q\lrcorner\cdot[\cdot].$$

- If p is a well-formed sequence ending with an O-move (respectively P-move), then the last move of p is preserved by P-view (respectively O-view); that is to say, the last move of $\ulcorner p\urcorner$ (respectively $\llcorner p\lrcorner$) comes from the same last move of p.

What kind of a sequence is the P-view (or O-view) of a well-formed sequence? Is it necessarily a well-formed sequence? For s ranging over well-formed sequences, the operation of P-view $s\mapsto\ulcorner s\urcorner$ is a projection. So $\ulcorner s\urcorner$ inherits the justification pointers from s in the natural way mentioned in the Remark 4.1.3. Unfortunately the projection does not always respect justification. For example the following well-formed sequence with its auxiliary sequence of justification indices

$$[_1\cdot(_2\cdot[_3\cdot(_4\cdot[_5\cdot(_6\cdot)_7 \qquad\qquad 0\cdot1\cdot2\cdot3\cdot2\cdot5\cdot6$$

as shown in Figure 2 has P-view $[_1\cdot(_2\cdot[_5\cdot(_6\cdot)_7$. With respect to the inherited justification pointers, this sequence does not satisfy condition (w3) of well-formed sequence: the last P-question "$(_6$" inherits a justification pointer to "$[_3$" which does not appear in the P-view. As it stands, as unary operations over well-formed sequences, P-view and O-view are only properly defined on certain well-formed sequences. What additional condition characterizes these well-formed sequences? This motivates the visibility condition.

Visibility condition Recall that we choose to specify the game tree of a dialogue game in terms of the collection of all paths in the tree. Such paths are called legal positions which we define as follows.

Definition 4.2.2 A *legal position* of a computational arena A is a well-formed sequence t which satisfies the following *visibility condition*:

> For any initial subsequence $s\cdot($ of the sequence t, the O-question "[" explicitly justifying the P-question "(" occurs in the P-view of s. Similarly for any initial subsequence $s\cdot[$ of the sequence t, the P-question explicitly justifying "[" occurs in the O-view of s.

It is easy to see that every initial subsequence of a legal position of an arena is a legal position. It is also easy to check that as projections acting on legal positions, both the operations of P-view and O-view respect justification. In performing the operation $\ulcorner - \urcorner$ (respectively $\llcorner - \lrcorner$), we implicitly assume that the justification indices are systematically reset in $\ulcorner s \urcorner$ (respectively $\llcorner s \lrcorner$) in the appropriate way.

The above visibility condition applies only to questions of a legal position. However this condition is strong enough to ensure that a corresponding visibility condition automatically holds for answers. This is made precise in the following lemma whose essentially straightforward proof is omitted.

Lemma 4.2.3 *The explicitly justifying question of every P-answer (respectively O-answer) in a legal position appears in the P-view (respectively O-view) of the legal position up to that point. More precisely,*

(i) for every initial subsequence $s \cdot)$ of a legal position t, the P-question which "$)$" explicitly answers occurs in $\llcorner s \lrcorner$;

(ii) for every initial subsequence $s \cdot]$ of a legal position t, the O-question which "$]$" explicitly answers occurs in $\ulcorner s \urcorner$. □

A sequence of moves of a computational arena A, equipped with an auxiliary sequence of justification pointers, is said to be a *P-view* (respectively *O-view*) if it is the P-view (respectively O-view) of some legal position of A. The operations of P-view and O-view are well-defined on legal positions: they map legal positions to legal positions.

Notation. Given a sequence $s \equiv m_1 \cdot m_2 \cdots m_n$, we write $s_{\leqslant m_i}$ for the initial subsequence of s up to and including m_i, that is to say, $m_1 \cdot m_2 \cdots m_i$. We write $s_{<m_i}$ for $m_1 \cdot m_2 \cdots m_{i-1}$.

Proposition 4.2.4 *Let s be a legal position of a computational arena. Both the O-view and the P-view of s are legal positions.*

Proof We will just consider the case of the P-view $\ulcorner s \urcorner$ of a legal position s for illustration. By the recursive definition of $\ulcorner - \urcorner$, it is clear that conditions (w1) and (w2) are satisfied by $\ulcorner s \urcorner$. For (w3), there are four cases to consider. First P-question. Consider an initial subsequence $p \cdot ($ of $\ulcorner s \urcorner$. By construction we see that p is just $\ulcorner s_{<(} \urcorner$. Since s satisfies the visibility condition, "$($" is explicitly justified by some O-question which occurs in p. Secondly O-question. Consider an initial subsequence of the shape $p \cdot [$ of $\ulcorner s \urcorner$. By definition, either p is the empty sequence, in which case, "$[$" is an initial question; or p ends with an P-question "$($" which explicitly justifies "$[$". Thirdly, P-answer: but by construction, a P-view does not contain any P-answer. Fourthly, O-answer. For any initial subsequence $p \cdot)$ of $\ulcorner s \urcorner$, suppose, for a contradiction, the explicitly justifying question "$($" of "$)$" does not occur in p. Note that p is $\ulcorner s' \urcorner$ such that $s' \cdot)$ is an initial subsequence of s. There are two possibilities:

- "$($" occurs in a segment $(_1 \cdots [_2$ of s where "$(_1$" explicitly justifies "$[_2$", and that p contains the segment $(_1 \cdot [_2$. In this case, s contains the segment $(_1 \cdots (\cdots [_2 \cdots)$. By the no dangling question condition, an appropriate answer of "$[_2$", say "$]_3$", occurs before "$)$". That is to say, s contains the segment $(_1 \cdots (\cdots [_2 \cdots]_3 \cdots)$. A P-view does not contain any P-answer: we consider each of the three cases explaining the disappearance of "$]_3$" in $\ulcorner s' \urcorner$ in turn. Case 1: s contains the segment $[_2 \cdots (_4 \cdots]_3 \cdots [_5$ with "$(_4$" explicitly justifying "$[_5$". By the no dangling question condition, "$(_4$" explicitly justifies an appropriate answer which occurs before "$]_3$". But this contradicts our assumption that "$(_4$" remains explicitly unanswered up to "$[_5$". Case 2: s contains the segment $[_2 \cdots [_4 \cdots]_3 \cdots]_5$ with "$[_4$" explicitly justifying "$]_5$" such that "$[_4$" and "$[_5$" vanish in

$\ulcorner s' \urcorner$ according to the fourth clause of the definition of $\ulcorner - \urcorner$. But this violates the no dangling question condition. Case 3: the segment $[_2 \cdots]_3$ vanishes under the P-view operation by virtue of the fourth clause of the recursive definition. But this contradicts our assumption that the segment $(_1 \cdot [_2$ appears in p.

- "(" occurs in a segment $[_1 \cdots]_2$ of s where "$[_1$" explicitly justifies "$]_2$". So s contains the segment $[_1 \cdots (\cdots]_2 \cdots)$: note that) has to occur to the right of $]_2$, for otherwise it would not occur in the P-view $\ulcorner s \urcorner$. But this violates the no dangling question condition.

Condition (w4) is vacuously satisfied since it is easy to see that whenever an answer and its explicitly justified question occur in a P-view, they are necessarily P-question and O-answer and adjacent to each other. Finally, the visibility condition. For any initial subsequence $p \cdot [$ of a P-view, by construction, the last move "(" of p explicitly justifies "[". Since any initial subsequence of a P-view is a P-view, and so, is P-view invariant, "(" occurs in $\ulcorner p \urcorner$. For any initial subsequence $p \cdot ($ of a P-view $\ulcorner s \urcorner$, note that by construction, p is $\ulcorner s' \urcorner$, for some initial subsequence s' of s such that $s' \cdot ($ is in turn a subsequence of s. Since s satisfies the visibility condition, the explicitly justifying question of "(" appears in $\ulcorner s' \urcorner \equiv p$. It then remains to observe that p is P-view invariant. $\square$

Lemma 4.2.5 *The operations of P-view and O-view are idempotent i.e.* $\ulcorner \ulcorner p \urcorner \urcorner = \ulcorner p \urcorner$, *and* $\llcorner \llcorner p \lrcorner \lrcorner = \llcorner p \lrcorner$ *for any legal position* p. $\square$

The proof is straightforward and is left to the reader. We note that since an initial subsequence of a P-view (respectively O-view) is a P-view (respectively O-view), it is therefore invariant under the P-view (respectively O-view) operation.

Recall that the history of justification of a move m in a well-formed sequence s is a well-defined subsequence of s which may be traced out by successively "chasing" the justification pointers starting from m until the initial question is reached. Our next result shows that the history of justification of a move in a legal position may be defined completely in terms of P-view and O-view.

Lemma 4.2.6 *Let t be a legal position of a computational arena.*

(i) If t ends with a question m then the history of justification of m (as a subsequence of t) is $\ulcorner \llcorner t \lrcorner \urcorner = \llcorner \ulcorner t \urcorner \lrcorner$, *and is therefore a legal position.*

(ii) If t ends with an answer m which is explicitly justified by (say) the question m' in t, then the history of justification of m is $\ulcorner \llcorner t_{\leqslant m'} \lrcorner \urcorner \cdot m = \llcorner \ulcorner t_{\leqslant m'} \urcorner \lrcorner \cdot m$, *where $t_{\leqslant m'}$ is the initial subsequence of t up to and including m'.*

Proof **(i)** We shall just consider the case of m being a P-question; the argument for the case of m being an O-question is similar. We prove by induction on the length of such sequences. The base case of length 2 is immediate. For the inductive case consider the following analysis of t where we use "(" to represent m, and that "('" and "[" are specific instances of moves in t such that "[" explicitly justifies "(" and that "('" explicitly justifies "[":

$$\overbrace{\underbrace{\underbrace{\overbrace{\underbrace{\cdots}_{s''}\ ('\ \cdots}^{s'}\ [}_{s}\ \cdots}_{t'}\ (}^{t}$$

As shown in the above diagram, we write $s \equiv s' \cdot [$ and $t \equiv t' \cdot ($. Now, we have $\llcorner t \lrcorner = \llcorner s' \lrcorner \cdot [\cdot($. Hence, we have

$$\ulcorner \llcorner t \lrcorner \urcorner \quad = \quad \ulcorner \llcorner s' \lrcorner \cdot [\cdot(\urcorner \quad = \quad \ulcorner \llcorner s' \lrcorner \cdot [\urcorner \cdot (\quad = \quad \ulcorner \llcorner s \lrcorner \urcorner \cdot (.$$

Hence, by the induction hypothesis, we have

$$\ulcorner \llcorner t \lrcorner \urcorner \quad = \quad \llcorner \ulcorner s \urcorner \lrcorner \cdot (. \tag{1}$$

We can already conclude that the history of justification of "(" is $\ulcorner \llcorner t \lrcorner \urcorner$ since, by the induction hypothesis, the history of justification of "[" is $\llcorner \ulcorner s \urcorner \lrcorner$. However we still need to show that $\ulcorner \llcorner t \lrcorner \urcorner = \llcorner \ulcorner t \urcorner \lrcorner$. Observe that $\ulcorner s \urcorner = \ulcorner s'' \urcorner \cdot (' \cdot [$, hence we get from (1),

$$\ulcorner \llcorner t \lrcorner \urcorner \quad = \quad \llcorner \ulcorner s'' \urcorner \cdot (' \cdot [\lrcorner \cdot (\quad = \quad \llcorner \ulcorner s'' \urcorner \cdot (' \lrcorner \cdot [\cdot(. \tag{2}$$

Now we have

$$\llcorner \ulcorner t \urcorner \lrcorner \quad = \quad \llcorner \ulcorner t' \urcorner \cdot (\lrcorner \quad = \quad \llcorner \ulcorner t' \urcorner_{<[} \lrcorner \cdot [\cdot(. \tag{3}$$

Recall that $p_{<m}$ denotes the initial subsequence of p up to but not including m. The last equation is justified since the O-question "[" which explicitly justifies "(" occurs in $\ulcorner t' \urcorner$: this is the visibility condition. For the same reason, and because "('" explicitly justifies "[", we infer that

$$\ulcorner t' \urcorner_{<[} \quad = \quad \ulcorner s'' \urcorner \cdot ('. \tag{4}$$

Combining equations (3) and (4), we get $\llcorner \ulcorner t \urcorner \lrcorner = \llcorner \ulcorner s'' \urcorner \cdot (' \lrcorner \cdot [\cdot($. Hence from (2), we have $\llcorner \ulcorner t \urcorner \lrcorner = \ulcorner \llcorner t \lrcorner \urcorner$. The proof of **(ii)** follows immediately. $\square$

4.3 Constructions of computational arenas

We could already have defined the product $A \times B$ and function space $A \Rightarrow B$ of computational arenas A and B. These will turn out to be actual product and function space of a cartesian closed category of computational arenas. The verification of the respective universal properties will have to wait until the category is introduced in the next section.

Product For product we simply take the obvious "disjoint sum" of the arenas A and B as directed graphs. More precisely

$$\begin{aligned}
\mathsf{Qn}(A \times B) &\stackrel{\text{def}}{=} \mathsf{Qn}(A) + \mathsf{Qn}(B), \\
\mathsf{Ans}(A \times B) &\stackrel{\text{def}}{=} \mathsf{Ans}(A) + \mathsf{Ans}(B), \\
\mathsf{qn}(A \times B) &\stackrel{\text{def}}{=} \mathsf{qn}(A) + \mathsf{qn}(B) \quad (= [\mathsf{qn}(A); \mathsf{in}_1, \mathsf{qn}(B); \mathsf{in}_2])
\end{aligned}$$

where in_i is the canonical injection map and $[f, g]$ the so-called source tupling.

Function space For $A \Rightarrow B$ it is simplest to draw a picture as in Figure 3. (In the picture there is only one initial move in B.) The initial moves of $A \Rightarrow B$ are those of B; and to the tree "below" each such initial move, we graft onto it a copy of the forest of questions of A. More formally we define

$$\begin{aligned}
\mathsf{Qn}(A \Rightarrow B) &\stackrel{\text{def}}{=} (\mathsf{Qn}(A) \times \mathsf{M}_B) + \mathsf{Qn}(B), \\
\mathsf{Ans}(A \Rightarrow B) &\stackrel{\text{def}}{=} (\mathsf{Ans}(A) \times \mathsf{M}_B) + \mathsf{Ans}(B), \\
\mathsf{qn}(A \Rightarrow B) &\stackrel{\text{def}}{=} (\mathsf{qn}(A) \times \mathsf{Id}_{\mathsf{M}_B}) + \mathsf{qn}(B),
\end{aligned}$$

where M_B is the set of initial (equivalently maximal) questions of B. The justification ordering $\leqslant_{A \Rightarrow B}$ is defined to be the least partial order which includes the partial order associated with $\mathsf{Qn}(A) \times \mathsf{M}_B +$

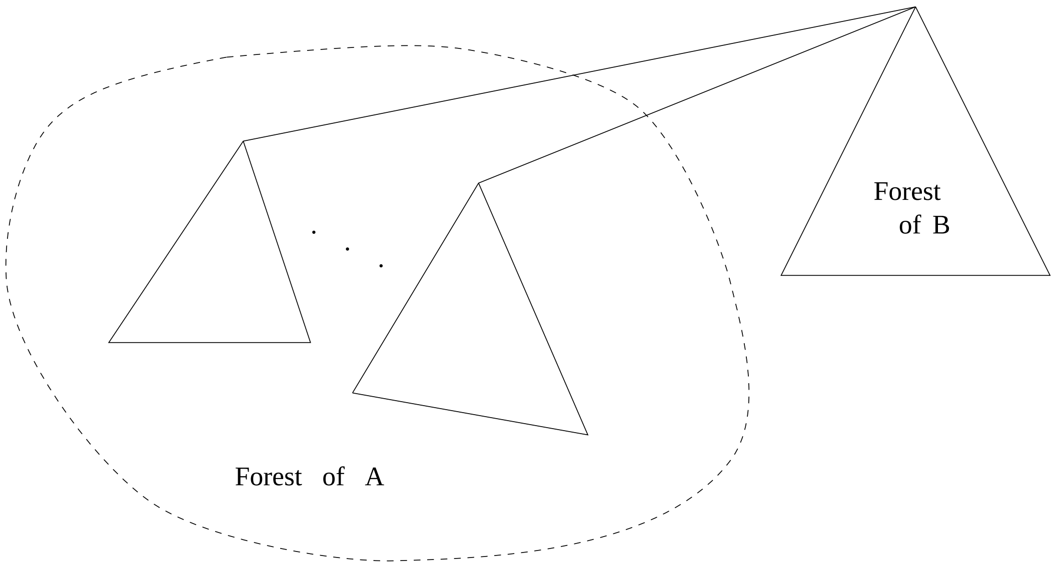

Figure 3: The forest of questions of $A \Rightarrow B$.

$\mathsf{Qn}(B)$ viewed as a construction of posets (M_B being a discrete poset), and satisfies the additional condition:

$$(q, m) \leqslant_{A \Rightarrow B} m \quad \text{for any } m \in \mathsf{M}_B \text{ and } q \in \mathsf{Qn}(A).$$

The net effect of the construction is that moves of the new computational arena $A \Rightarrow B$ are defined in terms of those of A and B in a way similar to the game semantics of the linear formula $(!A)^{\perp} \propto B$ (read "shriek A perp par B") in the style of Blass according to which a P-move (respectively O-move) in A becomes an O-move (respectively P-move) in $A \Rightarrow B$ (see [Bla92]). More accurately corresponding to each A-question at level $2n$ (respectively $2n+1$) of the forest $\mathsf{Qn}(A)$, there are m copies of the same question at level $2n+1$ (respectively $2n+2$) of the forest $\mathsf{Qn}(A \Rightarrow B)$, where m is the number of initial questions in $\mathsf{Qn}(B)$.

This is a good place to consider some examples and fix some notations.

Example 4.3.1 (i) The natural numbers computational arena $\mathbb{N}$ is specified by the following data:

- The forest of questions is a singleton tree — the initial O-question "$[^{\iota}$" (or simply "$[$" whenever its type is clear).
- The answers are all P-answers $]_0,]_1,]_2, \cdots$ which are appropriate to the only question "$[^{\iota}$".

(ii) The boolean computational arena $\mathbb{B}$ is defined similarly: the forest of questions is a singleton "$[^{o}$"; the answers (all P-answers) are "$]_{\mathsf{t}}$" and "$]_{\mathsf{f}}$".

There is no harm in writing the answers simply as $0, 1, 2, \cdots$ rather than $]_0,]_1,]_2, \cdots$, and we shall do so occasionally.

Remark 4.3.2 More generally, for any PCF-type $A = (A_1, \cdots, A_n, \iota)$, the forest of questions of the corresponding PCF-arena A is an inverted finite tree which is constructed by recursion as follows: "below" the initial question corresponding to ι, graft onto it a copy each of the tree of questions

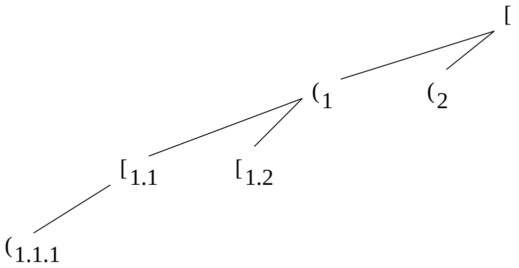

Figure 4: Forest of questions of arena $(((\iota,\iota),\iota,\iota),\iota,\iota)$.

corresponding to $A_1, \cdots, A_n$ respectively (of course, O-questions and P-questions in the A_is are interchanged as a result).

It is useful to establish a naming convention for questions of a PCF-arena A. Each question occurring in the tree is marked by an occurrence which is a finite sequence of positive integers. The occurrence is defined as follows:

- the initial question of ι (or o) has occurrence ϵ, the empty sequence
- for $1 \leqslant i \leqslant n$, if a question m of the arena A_i has occurrence l then m regarded as a question of $(A_1, \cdots, A_n, \iota)$ has occurrence $i \cdot l$.

For example the forest of questions of the arena $(((\iota,\iota),\iota,\iota),\iota,\iota)$ is shown in Figure 4 with the questions annotated with occurrences. Answers of a (PCF-)arena are just "copies" of answers at program type, namely, $]_0,]_1,]_2, \cdots$ or $]_{\mathsf{t}}$ and $]_{\mathsf{f}}$.

4.4 Properties of function space arenas

The definition of function space arena is remarkably simple. Our task now is to investigate properties of the function space arena and to express them synthetically, *i.e.* in terms of the respective properties of the sub-arenas A and B.

Components Let s be a legal position of a function space arena $A \Rightarrow B$ and let a be an (instance of an) A-initial move in s. We refer to the following subsequences of s as the *components* of s:

- $s \upharpoonright B$, the B-component of s (or the projection of s onto B), is the subsequence of s consisting of all B-moves in s,
- $s \upharpoonright (A, a)$, the (A, a)-component of s (or the projection of s onto the component (A, a)), is the subsequence of s consisting of all moves in A which are hereditarily justified by a.

In addition we write $s \upharpoonright (A, a)^+$ to mean the subsequence $m \cdot s \upharpoonright (A, a)$ where m is the initial B-move of s. Clearly every move of s belongs to precisely one component of s.

Two useful properties we shall prove shortly about function space arenas are:

- *Projection convention.* The projection of a legal position of $A \Rightarrow B$ onto B is a legal position in B, while the projection onto A can be read as an appropriate interleaving of a finite number of legal positions in A.

- *Switching convention.* Player, but not Opponent, is allowed to switch component, that is to say, either between a B-component and an (A,a)-component, or between different (A,a)-components.

Both conventions are reminiscent of similar conditions which are axioms in the construction of par games in the Blass-style game semantics of Linear Logic (see for example [Bla92, AJ94, HO93]). It is an indication of the simplicity of the arena approach that these conventions are consequent features and *not* part of the definition of the function space computational arena.

Let s be a legal position of the arena A. Let b_0 be an initial move of the arena B, and suppose s begins with an initial move a. It is easy to see that $b_0 \cdot s$ is a legal position of the function space arena $(A \Rightarrow B)$ such that

$$\{b_0 \cdot s\} \upharpoonright (A,a)^+ \quad = \quad b_0 \cdot s;$$

which is the same as saying that all moves of the legal position $b_0 \cdot s$ are in the component $(A,a)^+$. (In this paper we use curly parentheses "{" and "}" to indicate operator precedence rather than the more standard "(" and ")" which are reserved for denoting P-questions and O-answers respectively.) Conversely if t is a legal position in $(A \Rightarrow B)$ such that

$$t \upharpoonright (A,a)^+ \quad = \quad t,$$

then $t \upharpoonright (A,a)$ is a legal position in A beginning with the initial A-move a.

Lemma 4.4.1 *Let $b_0 \cdot s$ be a legal position of the arena $A \Rightarrow B$ such that all moves in s belong to the component (A,a). Then we have:*

(i) $\ulcorner b_0 \cdot s \urcorner^{A \Rightarrow B} = b_0 \cdot \llcorner s \lrcorner_A$.

(ii) $\llcorner b_0 \cdot s \lrcorner_{A \Rightarrow B} = b_0 \cdot \ulcorner s \urcorner^A$

Proof The proof is a straightforward induction on the length of the legal position in question. We sketch the argument for **(i)** for illustration. Suppose the $(A \Rightarrow B)$-legal position $b_0 \cdot s$ is of the form $b_0 \cdot p \cdot m \cdot r \cdot m'$ where the P-question m explicitly justifies the O-question m'. Then its P-view is $\ulcorner b_0 \cdot p \urcorner^{A \Rightarrow B} \cdot m \cdot m'$, which by the induction hypothesis is $b_0 \cdot \llcorner p \lrcorner_A \cdot m \cdot m'$. As a move in A, m is the O-question which explicitly justifies the P-question m'. Therefore $\llcorner p \lrcorner_A \cdot m \cdot m'$ is just $\llcorner p \cdot m \cdot r \cdot m' \lrcorner_A$. The other cases of the recursive definition of P-view are dealt with similarly. $\Box$

We are now in a position to state and prove the Switching Convention.

Proposition 4.4.2 *Let s be a legal position of an arena $A \Rightarrow B$ beginning with an initial move b, and let the last move of s be a P-move m (for (i) and (ii)).*

(i) **(O-view projection 1)**. *If m is in B then* $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright B = \llcorner s \upharpoonright B \lrcorner_B = \llcorner s \lrcorner_{A \Rightarrow B}$.

(ii) **(O-view projection 2)**. *If m is in the component (A,a) then*

$$\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A,a)^+ \quad = \quad b \cdot \ulcorner s \upharpoonright (A,a) \urcorner^A \quad = \quad \llcorner s \lrcorner_{A \Rightarrow B}.$$

(iii) **(Switching convention)**. *Whenever any two consecutive moves m and m' in $s = [\cdots m \cdot m' \cdots$ are in different components of s, then they are of the shape $\bullet \cdot \circ$, as opposed to $\circ \cdot \bullet$. In other words Player, but not Opponent, is allowed to switch component.*

Proof We prove (i), (ii) and (iii) by mutual induction on the length of s. The respective base cases are trivial. We consider the inductive cases in turn.

(i). Let m^- be the move in s which explicitly justifies m; by construction of s, m^- is an O-move in B. Since $s_{<m^-} \cdot m^-$ has length less than that of s, by the induction hypothesis of (iii), the last move of $s_{<m^-}$ is in B. Hence

$$\begin{aligned}
\llcorner s \lrcorner \restriction B &= \llcorner s_{<m^-} \lrcorner \restriction B \cdot m^- \cdot m && \text{by the induction hypo. of (i)} \\
&= \llcorner s_{<m^-} \restriction B \lrcorner \cdot m^- \cdot m \\
&= \llcorner \{ s_{<m^-} \cdot m^- \cdot m \} \restriction B \lrcorner \\
&= \llcorner s \restriction B \lrcorner .
\end{aligned}$$

Also by induction $\llcorner s \lrcorner \restriction B = \llcorner s_{<m^-} \lrcorner_{\restriction} B \cdot m^- \cdot m = \llcorner s_{<m^-} \lrcorner \cdot m^- \cdot m$, and so, $\llcorner s \lrcorner \restriction B = \llcorner s \lrcorner$.

(ii). If m is (an instance of) an initial A-move a then $\llcorner s \lrcorner = b \cdot m$. We then have

$$\llcorner s \lrcorner_{A \Rightarrow B} \restriction (A,a)^+ \quad = \quad b \cdot m \quad = \quad b \cdot \ulcorner s \restriction (A,a) \urcorner A.$$

If m is not an initial A-move then let m^- be the O-move explicitly justifying m. Note that both m and m^- belong to the same component (A,a), say. We have $\llcorner s \lrcorner = \llcorner s_{<m^-} \lrcorner \cdot m^- \cdot m$. By the induction hypothesis of (iii), the last move of $s_{<m^-}$ is in the same component (A,a) as m^-. Hence by the induction hypothesis of (ii),

$$\llcorner s \lrcorner_{A \Rightarrow B} \restriction (A,a)^+ \quad = \quad b \cdot \ulcorner s_{<m^-} \restriction (A,a) \urcorner^A \cdot m^- \cdot m \quad = \quad b \cdot \ulcorner s \restriction (A,a) \urcorner^A .$$

Also by induction $\llcorner s \lrcorner_{A \Rightarrow B} \restriction (A,a)^+ = \llcorner s \lrcorner_{A \Rightarrow B}$.

(iii). Let s be a legal position whose last move m is an O-move, and let m^- be the P-move in s immediately preceding m. There are two cases. First, m^- is in (A,a). By the induction hypothesis of (ii), $\llcorner s_{\leqslant m^-} \lrcorner$ has the form $[\cdot p$ where "[" is the initial B-move in s, and p is a sequence of moves in (A,a). By the visibility condition the move which explicitly justifies m is either "[" or some move in p. The former is to be rejected since m being an O-move can only be justified by a P-question. Hence m is explicitly justified by some move in the (A,a)-component, and so, it must be a move of the same component. The other case of m^- in B follows from the induction hypothesis of (i), and we leave the details as an easy exercise for the reader. □

Example 4.4.3 Consider the legal position s of the arena $A \Rightarrow B$ as in Figure 5. The P-view $\ulcorner s \urcorner$ is $b_1 \cdot a_1 \cdot a_4 \cdot b_4 \cdot b_5$ with justification indices $0 \cdot 1 \cdot 2 \cdot 1 \cdot 4$, and so, $\ulcorner s \urcorner \restriction B$ is $b_1 \cdot b_4 \cdot b_5$ with justification indices $0 \cdot 1 \cdot 2$. Clearly $s \restriction B = \ulcorner s \restriction B \urcorner$. Observe that $\ulcorner s \urcorner \restriction B$ is a (proper) subsequence of $\ulcorner s \restriction B \urcorner$.

Convention In the following we write $s \preccurlyeq t$ to mean that s is a subsequence (not necessarily initial) of t.

Proposition 4.4.4 (P-view Projection) *Let s be a legal position of an arena $A \Rightarrow B$. Suppose s ends with the move m.*

(i) If m is in B then $\ulcorner s \urcorner^{A \Rightarrow B} \restriction B \preccurlyeq \ulcorner s \restriction B \urcorner^B$.

(ii) If m is in the component (A,a) then $\ulcorner s \urcorner^{A \Rightarrow B} \restriction (A,a)^+ \preccurlyeq b \cdot \ulcorner s \restriction (A,a) \urcorner^A$.

Proof Since the proof is quite technical, we relegate it to Appendix A. □

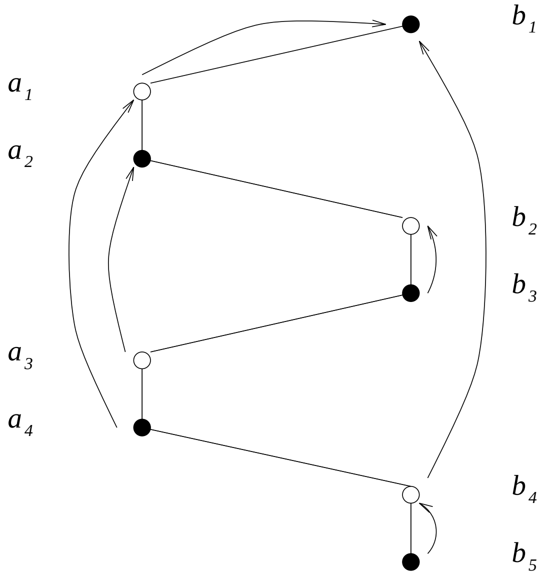

Figure 5: An example.

Proposition 4.4.5 (Projection Convention) *Let s be a legal position of the arena $A \Rightarrow B$.*

(i) The projection $s \restriction B$ of s onto B is a legal position in B.

(ii) For any instance a of an initial A-move in s, the projection $s \restriction (A, a)$ is a legal position of the arena A.

Proof **(i).** We show that $t \equiv s \restriction B$ is a legal position of B. Condition (w1) of a well-formed sequence is clearly inherited by t. An immediate consequence of the Switching Convention (Proposition 4.4.2) is that t is an alternating sequence. Given any P-question "(" in t. By regarding "(" as a move in s which is a legal position, the explicitly justifying question "[" (say) occurs before "(" in s and is pointed to by the justification index. Now "[" is a B-question, and so, it occurs in t before "(". The same argument applies to explicit justification of O-questions in t. By proceeding in an identical manner and by considering the definition of the order structure of $\mathsf{Qn}(A \Rightarrow B)$ in term of that of $\mathsf{Qn}(B)$, we see that properties (w3) as well as (w4) are inherited by t. For the visibility condition, take any P-question "(" in t. By regarding it as a P-question in s which is a legal position, we infer that the explicitly justifying question "[" of "(" occurs in $\ulcorner s_{<(} \urcorner$. Since "[" is a B-move, it occurs in $\ulcorner s_{<(} \urcorner \restriction B$. Hence, by Proposition 4.4.4, "[" occurs in $\ulcorner s_{<(} \restriction B \urcorner$.

For **(ii)** we shall just verify the visibility condition; the rest is routine. Let s be a legal position in $A \Rightarrow B$ which begins with b_0. Take any P-question (regarded as a move in A) m in $t \equiv s \restriction (A, a)$. By regarding it as an O-question (of $A \Rightarrow B$) in s, the P-question $\underline{m}$ (also in the component (A, a)) which explicitly justifies m is in $\llcorner s_{<m} \lrcorner_{A \Rightarrow B}$. Observe that $\underline{m}$ belongs to $\llcorner s_{<m} \lrcorner \restriction (A, a)^+$. By Proposition 4.4.2(ii), $\underline{m}$ is in $\llcorner s_{<m} \restriction (A, a)^+ \lrcorner$ which is just $\llcorner b_0 \cdot \{s_{<m}\} \restriction (A, a) \lrcorner$. By definition all moves of $s_{<m} \restriction (A, a)$ are in the component (A, a); and the last move is a P-move in $A \Rightarrow B$. Hence, by Lemma 4.4.1, $\underline{m}$ is in $\ulcorner t_{<m} \urcorner$ as required. $\square$

Remark 4.4.6 The payoff of the visibility condition may be seen in the above: the projection of a well-formed sequence of a function space computational arena $A \Rightarrow B$ (say) onto either of the two components A or B is not necessarily well-formed. It is easy to check that the following sequence with the auxiliary sequence of justification indices is well-formed in $A \Rightarrow B$:

$$[^B \cdot (^A \cdot [^A \cdot (^B \cdot [^A; \qquad\qquad 0 \cdot 1 \cdot 2 \cdot 1 \cdot 2$$

however the projection onto A with the auxiliary sequence of justification indices systematically reset in the natural way as follows

$$(^A \cdot [^A \cdot [^A \qquad\qquad 0 \cdot 1 \cdot 1$$

is not an alternating sequence.

5 Innocent strategies

A strategy for a player is a rule or a method that determines how a player is to respond at a position where he is expected to make a move. Abstractly a strategy for Player, say, is a partial function (of a certain kind) mapping legal positions (at which Player is to move) to P-moves. We represent a strategy as an appropriate sub-tree of the game tree associated with an arena. Since we have chosen to represent the game tree associated with an arena A formally as the collection of all paths (= legal positions) in the tree, we define a *P-strategy* σ of A to be a non-empty prefix-closed collection of legal positions of A satisfying the following conditions:

(s1) *Determinacy.* For any $s \in \sigma$ at which Player is to move, if both $s \cdot a$ and $s \cdot b$ are in σ then $a = b$.

(s2) *Contingent completeness.* For any $s \in \sigma$ at which Opponent is to move and for any O-move a, if $s \cdot a$ is a legal position then it is in σ.

5.1 Uncovering and composition of strategies

First some terminology. A sequence of moves of arenas $A_1, \cdots, A_n$ is said to be *explicitly justified* if it is equipped with justification pointers represented by an associated sequence of natural numbers called justification indices.

Suppose we are given strategies σ and τ of computational arenas $A \Rightarrow B$ and $B \Rightarrow C$ respectively. Take a legal position s of the computational arena $A \Rightarrow C$. We define the *uncovering* of s in accord with σ and τ, written $\mathbf{u}(s, \sigma, \tau)$, as the unique maximal explicitly justified sequence u of moves of A, B and C satisfying the following properties:

$$\begin{array}{llcl}
(\mathbf{u1}) & u \upharpoonright (A, C) & \leqslant & s, \\
(\mathbf{u2}) & u \upharpoonright (B, C) & \in & \tau, \\
(\mathbf{u3}) & u \upharpoonright (A, B)_b & \in & \sigma;
\end{array}$$

where $\leqslant$ in the first clause is prefix ordering between legal positions. For legal positions s and t, we say that $s \leqslant t$ holds just in case

- as sequences of moves, s is a prefix of t
- the auxiliary sequence of justification indices of s is a prefix of that of t.

The subscript b in the third clause ranges over all instances of initial B-moves in u. Note that u may be an infinite sequence of moves; in which case we read the second clause as: every finite truncation of the projected sequence $u \upharpoonright (B, C)$ belongs to τ. The same qualification applies to the third clause.

Convention. By a *component* is meant either (B, C), or $(A, B)_b$ where b is an instance of an initial B-move occurring in u. We will use X as a meta-variable ranging over components, and ρ_X a meta-variable denoting σ if X is $(A, B)_b$, or τ if X is (B, C).

Definition 5.1.1 The *uncovering* $u = \mathbf{u}(s, \sigma, \tau)$ may be generated by the following algorithm. We write $u = u_1 \cdot u_2 \cdot u_3 \cdots$ with u_i ranging over moves in arenas A, B and C. Let n be the length of u (note that n may be infinite). We show inductively that for each $i \leqslant n$, the initial subsequence $v \equiv u_1 \cdot u_2 \cdots u_i$ satisfies clauses (u1), (u2) and (u3); and if $v < u$, then the next move u_{i+1} is uniquely defined.

(1) If s is the empty sequence, then so is u; otherwise the initial move of s is the initial move of u.

(2) If $v \equiv u_1 \cdot u_2 \cdot \cdots \cdot u_i$ has been generated and u_i can be regarded as an O-move in the component X (that is to say, u_i is either an O-move in the arena $A \Rightarrow C$ or it is a B-move), then $v \upharpoonright X$ is inductively in ρ_X (which is σ or τ as appropriate).

- **(2.1)** *If* $(v \upharpoonright X) \cdot a \in \rho_X$ for (a necessarily unique) a, then there are two possibilities:
  - **(2.1.1)** either a is a B-move, then we define u_{i+1} to be a;
  - **(2.1.2)** or a is a move in $A \Rightarrow C$, in which case,
    - **(2.1.2.1)** if we still have $v \upharpoonright (A, C) \cdot a \leqslant s$, then define u_{i+1} to be a,
    - **(2.1.2.2)** otherwise we stop with $u = v$.
- **(2.2)** *If not*, we stop with $u = v$.

(3) If $v \equiv u_1 \cdot u_2 \cdot \cdots \cdot u_i$ has been generated and ends with a move u_i in component X which cannot be regarded as an O-move (that is to say, u_i is a P-move in $A \Rightarrow C$), then inductively, $v \upharpoonright (A, C) \leqslant s$.

- **(3.1)** *If* there is a move in $A \Rightarrow C$, say a, making $v \upharpoonright (A, C) \cdot a \leqslant s$, then define u_{i+1} to be a. (The move a must be an O-move in the same component (in the sense of Proposition 4.4.2) of $A \Rightarrow C$ as u_i, by the Switching Convention.)
- **(3.2)** *If not*, stop with $u = v$.

It is straightforward to check that u generated as above is the uncovering of s according to σ and τ, and is uniquely defined.

We make the following useful observation.

Lemma 5.1.2 *For any legal position t of arena $A \Rightarrow C$, and for any $s \leqslant \mathbf{u}(t, \sigma, \tau) \upharpoonright (A, C)$, we have*

$$\mathbf{u}(s, \sigma, \tau) \upharpoonright (A, C) \quad = \quad s.$$

Proof Let v be the maximal initial subsequence of $\mathbf{u}(t, \sigma, \tau)$ such that $v \upharpoonright (A, C) = s$. It suffices to check that v is in fact $\mathbf{u}(s, \sigma, \tau)$. □

Composition of strategies We can now formally define composition. Given strategies σ and τ of $A \Rightarrow B$ and $B \Rightarrow C$ respectively, we define

$$\sigma ; \tau \quad \stackrel{\text{def}}{=} \quad \{\, \mathbf{u}(s, \sigma, \tau) \upharpoonright (A, C) : s \text{ is a legal position of } A \Rightarrow C \,\}.$$

The composition of strategies is reminiscent of CSP-style parallel composition plus hiding [Hoa85].

The collection $\sigma;\tau$ is clearly non-empty: any initial move of $A \Rightarrow C$ (regarded as a singleton sequence) belongs to $\sigma;\tau$. For any legal position t of $A \Rightarrow C$, and for any $s \leqslant \mathbf{u}(t,\sigma,\tau) \upharpoonright (A,C)$, by Lemma 5.1.2, $s = \mathbf{u}(s,\sigma,\tau) \upharpoonright (A,C)$, and so, s is in $\sigma;\tau$. Therefore $\sigma;\tau$ is prefix-closed.

Now consider Definition 5.1.1 (of uncovering). Suppose $s \in \sigma;\tau$ ends with an O-move d_0. Writing $\mathbf{u}(s,\sigma,\tau)$ as u, it is clear that if u is infinite, then there can be no P-move a such that $s \cdot a \in \sigma;\tau$. So suppose u is finite. By construction of the uncovering u, for some finite $n \geqslant 0$, there are B-moves $d_1, \cdots, d_n$, and for each $0 \leqslant i \leqslant n$, writing X_i as the component in which d_i may be regarded as an O-move, such that:

- $u \equiv v \cdot d_0 \cdot d_1 \cdots d_n$ with $v \cdot d_0 \upharpoonright (A,C) = s$, and
- $(v \cdot d_0 \cdot d_1 \cdot \cdots \cdot d_i \upharpoonright X_i) \cdot d_{i+1} \in \rho_{X_i}$, for each $0 \leqslant i \leqslant n-1$.

Further

(A) *either* $(v \cdot d_0 \cdot d_1 \cdot \cdots \cdot d_n \upharpoonright X_n)$ is a maximal legal position in ρ_{X_n}, corresponding to case (2.2) in Definition 5.1.1; note that in this case there can be no P-move a for which $s \cdot a \in \sigma;\tau$;

(B) *or* there is some P-move a in $A \Rightarrow C$ such that $v \cdot d_0 \cdot d_1 \cdot \cdots \cdot d_n \upharpoonright X_n \cdot a \in \rho_{X_n}$ corresponding to case (3.2) in the same definition. In this case observe that $s \cdot a \in \sigma;\tau$, and a is unique.

Hence we see that condition (s1) (of strategy) is satisfied by the collection $\sigma;\tau$. Also, by reference to clause (3.1) of Definition 5.1.1, $\sigma;\tau$ inherits property (s2) from the same of σ and τ. Hence $\sigma;\tau$ is a strategy.

Composition of strategies as defined is associative. Given strategies σ,τ and ρ of arenas $A \Rightarrow B$, $B \Rightarrow C$ and $C \Rightarrow D$ respectively; take any $s \in (\sigma;\tau);\rho$, we show that $s \in \sigma;(\tau;\rho)$ by induction on the length of s. (Inclusion in the other direction is similar and we leave it as an exercise for the reader.) The base case is obvious. Let m be the last move of s. W.l.o.g. suppose m is an A-move. The case of m being an O-move is easy: by the switching convention (Proposition 4.4.2), the move m^- preceding m must also be in the same component as m. Since $s_{\leqslant m^-}$ is in $\sigma;(\tau;\rho)$ by the induction hypothesis, so must s by condition (s2) of strategy.

Now suppose m is a P-move. We consider the less straightforward case of m^- being a D-move. By the induction hypothesis $s_{\leqslant m^-}$ is in $\sigma;(\tau;\rho)$. Writing $\mathbf{u}(s_{\leqslant m^-},\sigma,\tau;\rho)$ as u, by Lemma 5.1.2, $u \upharpoonright (A,D) = s_{\leqslant m^-}$, as depicted in Figure 6. Since s is in $(\sigma;\tau);\rho$; by Lemma 5.1.2, writing $\mathbf{u}(s,\sigma;\tau,\rho)$ as v, we have $v \upharpoonright (A,D) = s$, and for some C-moves $\vec{c}$, v has shape $w \cdot \vec{c} \cdot m$ where w has m^- as the last move. Let c_n be the last move in $\vec{c}$, and suppose c_n is in the component $(A,C)_{c'}$. By definition of the uncovering v,

$$v \upharpoonright (A,C)_{c'} \quad \in \quad \sigma;\tau.$$

Observe that c_n and m are the penultimate and last move respectively in $v \upharpoonright (A,C)_{c'}$. By definition of $\sigma;\tau$ and by Lemma 5.1.2, writing $\mathbf{u}(v \upharpoonright (A,C)_{c'},\sigma,\tau)$ as l, we have

$$l \upharpoonright (A,C)_{c'} \quad = \quad v \upharpoonright (A,C)_{c'}$$

where l has shape $\cdots \vec{c} \cdot \vec{b} \cdot m$. Suppose m be in the $(A,B)_{b'}$-component of l. We have $l \upharpoonright (A,B)_{b'} \in \sigma$. It is easy to check that $u \cdot \vec{b} \cdot m$ is $\mathbf{u}(s,\sigma,\tau;\rho)$, and $u \cdot \vec{b} \cdot m \upharpoonright (A,D)$ is s. Hence $s \in \sigma;(\tau;\rho)$.

To summarize we have shown:

Proposition 5.1.3 *Composition of strategies is well-defined and associative.* □

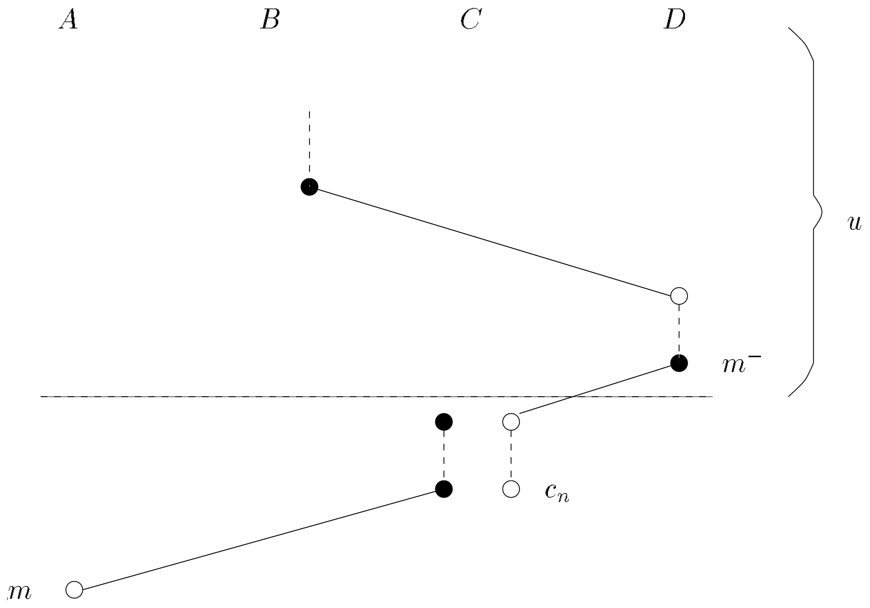

Figure 6: Composition of strategies is associative.

For strategies σ, τ and υ of arenas $A \Rightarrow B$, $B \Rightarrow C$ and $C \Rightarrow D$ respectively and for a legal position s of $A \Rightarrow D$, we define $\mathbf{u}(s, \sigma, \tau, \upsilon)$ as the unique maximal explicitly justified sequence u of moves of arenas A, B, C and D satisfying

- $u \restriction (A, D) \leqslant s$
- $u \restriction (C, D) \in \upsilon$
- $u \restriction (B, C)_{c_i} \in \tau$ for each instance c_i of initial $(B \Rightarrow C)$-move occurring in u
- $u \restriction (A, B)_{b_j} \in \sigma$ for each instance b_j of an initial $(A \Rightarrow B)$-move occurring in u.

We leave the essentially straightforward proof of the following result to the reader.

Proposition 5.1.4 *The composition $(\sigma; \tau); \upsilon$ (or equivalently $\sigma; (\tau; \upsilon)$) is*

$$\sigma; \tau; \upsilon \quad = \quad \{\, \mathbf{u}(s, \sigma, \tau, \upsilon) \restriction (A, D) : s \text{ is a legal position of } A \Rightarrow D \,\}.$$

□

5.2 Representation of innocent strategies

A strategy for Player, say, is *history-free* if Player's move at any position of the game where he is expected to play is determined by the last move of Opponent: the history of the play prior to the last move has no bearing on Player's response (see [AJ94]). If Player's move depends on the entire history of the play up to that point, then the strategy is said to be *history-sensitive*. Innocent strategies are neither history-free nor history-sensitive. Rather they determine a response to Opponent's move on the basis of a narrow *view* of the history of the play up to that point (hence the adjective "innocent").

Definition 5.2.1 (Innocent strategy) Each such P-strategy, say σ, is determined by a partial function of a certain kind mapping P-views (of legal positions at which Player is to move) to P-moves in the following way: for any legal position $s \in \sigma$ at which P is to move, and for any P-move a,

$$s \cdot a \in \sigma \qquad \Longleftrightarrow \qquad f(\ulcorner s \urcorner) \text{ is defined, and } f(\ulcorner s \urcorner) = a.$$

We call such a function f a defining partial function for the innocent strategy σ.

It is easy to see that corresponding to each innocent strategy σ, there is a least (and so unique) such partial function regarded as a graph, written f_σ. We call f_σ the *representing* partial function for the innocent strategy σ. We also call functions of the form f_σ *representing innocent functions*. The representing function f_σ may be characterised as follows: for any P-view p ending with an O-move, and for any P-move a, $f_\sigma(p)$ is defined, and equal to a if and only if there is some $s \in \sigma$ such that

(i) $\ulcorner s \urcorner = p$, and

(ii) $s \cdot a \in \sigma$.

So for any P-view p ending with an O-move, $f_\sigma(p)$ is undefined if and only if *either* there is no legal position $s \in \sigma$ such that $p = \ulcorner s \urcorner$, *or* for some (and hence for every) legal position $s \in \sigma$ such that $p = \ulcorner s \urcorner$, s is a maximal legal position in σ. (It is easy to see that a history-free strategy is automatically innocent, but the converse is not true.)

Representation of innocent strategies Clearly not every partial function from P-views to P-moves gives rise to an innocent strategy. There is, however, a necessary and sufficient condition for a partial function of the appropriate type to be the representing partial function of some innocent strategy. We first introduce a helpful notion. Let A be a computational arena. Take any P-view p of A ending with an O-move, and let f be a partial function of the following type:

$$f \quad : \quad \{\,\text{P-views of } A \text{ ending with O-moves}\,\} \rightharpoonup \{\,\text{P-moves in } A\,\}.$$

The collection $\sum_f p$ of *f-traces of p* is a subset of the legal positions of A defined as follows:

$$s \in \textstyle\sum_f p \quad \Longleftrightarrow \quad \begin{cases} s \equiv m_1 \cdot m_2 \cdots m_{2n+1}, \text{ for some } n \geqslant 0, \text{ is a legal position of } A \text{ where} \\ \bullet \quad \ulcorner s \urcorner = p, \\ \bullet \quad \text{for each } 0 \leqslant l < n,\ f(\ulcorner m_1 \cdot m_2 \cdots m_{2l+1} \urcorner) = m_{2l+2}. \end{cases}$$

Definition 5.2.2 A partial function of the above type is said to be *innocent* if for any P-view p of a legal position ending with an O-move, and for any P-move a, $f(p)$ is defined, and equal to a if and only if the following conditions are satisfied:

(if1) $\sum_f p$ is non-empty,

(if2) for any legal position s, if $s \in \sum_f p$, then $s \cdot a$ is a legal position of A.

Consider the collection σ_f of legal positions defined inductively as follows. For n ranging over ω, and writing ϵ as the empty sequence, we define

$$\begin{aligned} \sigma_f^0 &\stackrel{\text{def}}{=} \{\,\epsilon\,\}, \\ \sigma_f^{2n+1} &\stackrel{\text{def}}{=} \{\, s \cdot a \mid s \in \sigma_f^{2n}, a \in \text{O-moves, and } s \cdot a \text{ is a legal position}\,\}, \\ \sigma_f^{2n+2} &\stackrel{\text{def}}{=} \{\, s \cdot f(\ulcorner s \urcorner) \mid s \in \sigma_f^{2n+1}, f(\ulcorner s \urcorner) \text{ is defined}\,\}, \\ \sigma_f &\stackrel{\text{def}}{=} \textstyle\bigcup_{n \in \omega} \sigma_f^n. \end{aligned}$$

Lemma 5.2.3 *(i) Given any innocent (partial) function f, the collection σ_f of legal positions defined above is an innocent strategy.*

(ii) For any P-view p of a legal position ending with an O-move,

$$\sum_f p \quad = \quad \{ s \in \sigma_f : \ulcorner s \urcorner = p \}.$$

$\square$

The proof is an easy exercise which is left to the reader. In fact innocent strategies and innocent functions are the same thing, as the following proposition shows. We omit the largely straightforward proof.

Lemma 5.2.4 *For any innocent strategies σ and σ' and any innocent functions f and f', we have,*

(i) $f_{\sigma_f} = f$.

(ii) $\sigma_{f_\sigma} = \sigma$.

(iii) $f \subseteq f'$ *if and only if* $\sigma_f \subseteq \sigma_{f'}$.

(iv) $\sigma \subseteq \sigma'$ *if and only if* $f_\sigma \subseteq f_{\sigma'}$. $\square$

Consequently we are justified in regarding an innocent function f as a (unique) representation of the innocent strategy σ_f.

We use this representation in a number of ways: to explain the notion of a finite or compact strategy (see Lemma 5.5.2) and to define recursive strategies (see §5.6). In a sense innocent functions seem more fundamental than innocent strategies (as trees); but the latter are well suited to an explanation (and definition) of composition.

5.3 Composition of innocent strategies

In §5.1 we saw how to compose strategies of computational arenas. Now we consider whether this operation preserves innocence. The main result in this section is the following:

Proposition 5.3.1 *Composition of innocent strategies is well-defined.*

To prove this proposition, we need to verify the following: for any innocent strategies $\sigma = \sigma_f$ and $\tau = \tau_g$ of computational arenas $A \Rightarrow B$ and $B \Rightarrow C$ respectively, the composition $\sigma;\tau$ is an innocent strategy. By Proposition 5.1.3, we know that $\sigma;\tau$ is a strategy. It remains to prove that the strategy $\sigma;\tau$ is innocent; that is to say,

> there is a partial function h mapping P-views of legal positions (at which P is to move) of $A \Rightarrow C$ to P-moves of the same arena such that for any legal position s of $\sigma;\tau$ at which P is to move, and for any P-move a of $A \Rightarrow C$,
>
> $$s \cdot a \in \sigma_f;\tau_g \quad \Longleftrightarrow \quad h(\ulcorner s \urcorner) \text{ is defined, and is equal to } a.$$

Suppose s is a legal position of the arena $A \Rightarrow C$. Write $u \equiv \mathbf{u}(s, \sigma, \tau)$, the uncovering of s according to σ and τ. For d an O-move in the *general sense* in u (i.e. an O-move in $A \Rightarrow C$ or a B-move), let $\overline{d}$ be the preceding O-move in $A \Rightarrow C$ (so that $\overline{d}$ coincides with d when d is a move of $A \Rightarrow C$) and write

$$u(\overline{d}) \quad \equiv \quad \mathbf{u}(\ulcorner s_{\leqslant \overline{d}} \urcorner, \sigma, \tau).$$

Since the P-view of a legal position is a legal position (Lemma 4.2.4), $u(\overline{d})$ is well-defined.

Proposition 5.3.2 *For any generalised O-move d in u, we construct by recursion on d an identification of $u(\overline{d})$ as a subsequence of $u_{\leqslant d}$ of the following kind: $u_{\leqslant d}$ looks like:*

$$\bullet \boxed{} \circ \bullet \boxed{} \circ \bullet \boxed{} \circ \bullet \boxed{} \circ \quad \cdots$$

where the leftmost "$\bullet$" is an initial move in C; other occurrences of "$\bullet$" are O-moves in $A \Rightarrow C$; "$\boxed{}$" represents a block of alternating moves in B, and "$\circ$" is a P-move in $A \Rightarrow C$. The following are satisfied:

(i) *$u(\overline{d})_{\leqslant d}$ simply omits segments of the form "$\bullet \boxed{} \circ$"; where "$\bullet$", and so also "$\circ$", are missing from $\ulcorner s_{\leqslant \overline{d}} \urcorner$. We shall write $u(\overline{d})_{\leqslant d} < u_{\leqslant d}$ to mean that $u(\overline{d})_{\leqslant d}$ is a subsequence of the required kind.*

(ii) *If X is the component in which d is an O-move, then*

$$\ulcorner u(\overline{d})_{\leqslant d} \restriction X \urcorner \quad = \quad \ulcorner u_{\leqslant d} \restriction X \urcorner.$$

Before we prove the Proposition, let us unpack the equation in clause (ii) and express it in terms of the following commuting diagram: for any legal position s of $A \Rightarrow C$, and for any generalised O-move d occurring in s, and $\overline{d}$ as before,

$$\begin{tikzcd}
s, d, \overline{d} \arrow[r, mapsto, "\text{uncover } s"] \arrow[d, mapsto, "\text{truncate } s \text{ at } \overline{d}"] & \mathbf{u}(s, \sigma, \tau) \equiv u \arrow[rr, mapsto, "\text{truncate at } d"] & & u_{\leqslant d} \arrow[ddd, mapsto, "\text{P-view in } X_d"'] \\
s_{\leqslant \overline{d}} \arrow[d, mapsto, "\text{P-view in } A \Rightarrow C"] & & & \\
\ulcorner s_{\leqslant \overline{d}} \urcorner \arrow[d, mapsto, "\text{uncover}"] & & & \\
u(\overline{d}) \equiv \mathbf{u}(\ulcorner s_{\leqslant \overline{d}} \urcorner, \sigma, \tau) \arrow[r, mapsto, "\text{truncate at } d"'] & u(\overline{d})_{\leqslant d} \arrow[rr, mapsto, "\text{P-view in } X_d"'] & & \ulcorner u(\overline{d})_{\leqslant d} \restriction X_d \urcorner = \ulcorner u_{\leqslant d} \restriction X_d \urcorner
\end{tikzcd}$$

Proof We prove by a case analysis on d.

Case 1. d is an O-move in $A \Rightarrow C$.

Case 1a. d is the initial O-move in C: this is obvious.

Case 1b. d is an O-move after the initial C-move, so that $d = \overline{d}$, say in component X. Then by condition (w3) of legal position, d is explicitly justified by a P-move, call it e, in component X. Let x be the O-move immediately preceding e in u. Note that x is still in component X, though it may be a B-move.

(i). Note that $s_{\leqslant \overline{d}}$ is of the form $s_{\leqslant \overline{x}} \cdot e \cdots d$, so that taking the P-view in $A \Rightarrow C$, we have $\ulcorner s_{\leqslant \overline{d}} \urcorner = \ulcorner s_{\leqslant \overline{x}} \urcorner \cdot e \cdot d$. Hence $u(\overline{d}) \equiv \mathbf{u}(\ulcorner s_{\leqslant \overline{d}} \urcorner, \sigma, \tau) = \mathbf{u}(\ulcorner s_{\leqslant \overline{x}} \urcorner \cdot e \cdot d, \sigma, \tau)$ has $u(\overline{x}) \equiv \mathbf{u}(\ulcorner s_{\leqslant \overline{x}} \urcorner, \sigma, \tau)$ as an initial subsequence. By the recursion hypothesis, $u(\overline{x})_{\leqslant x} \lessdot u_{\leqslant x}$ "is" a subsequence of the required kind, and so, also $u(\overline{d})_{\leqslant x} = u(\overline{x})_{\leqslant x}$. By the induction hypothesis for (ii), we have $\ulcorner u(\overline{x})_{\leqslant x} \upharpoonright X \urcorner = \ulcorner u_{\leqslant x} \upharpoonright X \urcorner$. Note that we have already observed that x and d occur in the same component X. Hence,

$$h_X(\ulcorner u(\overline{d})_{\leqslant x} \upharpoonright X \urcorner) \quad = \quad h_X(\ulcorner u(\overline{x})_{\leqslant x} \upharpoonright X \urcorner) \quad = \quad h_X(\ulcorner u_{\leqslant x} \upharpoonright X \urcorner) \quad = \quad e;$$

by construction of u. And it follows by construction of $u(\overline{d})$ that $u(\overline{d})_{\leqslant d}$ is $u(\overline{d})_{\leqslant x} \cdot e \cdot d = u(\overline{x})_{\leqslant x} \cdot e \cdot d$. This provides an obvious identification of $u(\overline{d})_{\leqslant d}$ as a subsequence of $u_{\leqslant d}$ of the right kind. Note that we omit all the "$\bullet\ \boxed{}\ \circ$" segments sandwiched between e and d.

(ii). Note that $u_{\leqslant d}$ is of the form $u_{\leqslant x} \cdot e \cdots d$. So

$$\begin{aligned}
\ulcorner u_{\leqslant d} \upharpoonright X \urcorner &= \ulcorner u_{\leqslant x} \upharpoonright X \urcorner \cdot e \cdot d && \text{by the induction hypothesis of (ii)} \\
&= \ulcorner u(\overline{x})_{\leqslant x} \upharpoonright X \urcorner \cdot e \cdot d \\
&= \ulcorner u(\overline{x})_{\leqslant x} \cdot e \cdot d \upharpoonright X \urcorner \\
&= \ulcorner u(\overline{d})_{\leqslant d} \upharpoonright X \urcorner.
\end{aligned}$$

Case 2. d is an O-move in B, say in component X. Hence, d is a P-move in the paired component Y. Let y be the O-move immediately preceding d in component Y. Note that $\overline{d} = \overline{y}$.

(i). By the recursion / induction hypotheses, we have $u(\overline{y})_{\leqslant y} \lessdot u_{\leqslant y}$, a subsequence of the required kind. As $u_{\leqslant d} = u_{\leqslant y} \cdot d$, we know that $h_Y(\ulcorner u_{\leqslant y} \upharpoonright Y \urcorner) = d$. By the induction hypothesis, we know that $\ulcorner u(\overline{y})_{\leqslant y} \upharpoonright Y \urcorner = \ulcorner u_{\leqslant y} \upharpoonright Y \urcorner$, so $\ulcorner u(\overline{d})_{\leqslant y} \upharpoonright Y \urcorner = \ulcorner u_{\leqslant y} \upharpoonright Y \urcorner$. Hence, by construction, we see that $u(\overline{d})$ extends $u(\overline{d})_{\leqslant y}$ by

$$h_Y(\ulcorner u(\overline{d})_{\leqslant y} \upharpoonright Y \urcorner) \quad = \quad h_Y(\ulcorner u_{\leqslant y} \upharpoonright Y \urcorner) \quad = \quad d.$$

This gives the identification $u(\overline{d})_{\leqslant d} = u(\overline{d})_{\leqslant y} \cdot d$ as a subsequence of $u_{\leqslant d}$ of the required kind.

(ii). Now let e be the P-question in X which explicitly justifies d in u and x the O-move immediately preceding e.

We know by **(i)** that $u(\overline{d})_{\leqslant d} \lessdot u_{\leqslant d}$ is a subsequence of a certain kind. Hence, since e occurs in $u(\overline{d})_{\leqslant d}$, we infer that x and $\overline{x}$ both occur in $u(\overline{d})_{\leqslant d}$ because $\overline{x}, x, e$ and d all occur in the same segment "$\bullet\ \boxed{}\ \circ$" which is in $u(\overline{d})_{\leqslant d}$. It follows that $\overline{x}$ occurs in $s_{\leqslant \overline{d}}$ so $s_{\leqslant \overline{x}} \leqslant s_{\leqslant \overline{d}}$ and so $u(\overline{d})$ starts like $u(\overline{x})$ and in particular, $u(\overline{d})_{\leqslant x} = u(\overline{x})_{\leqslant x}$. By the induction hypothesis, we know that $\ulcorner u(\overline{x})_{\leqslant x} \upharpoonright X \urcorner = \ulcorner u_{\leqslant x} \upharpoonright X \urcorner$, and hence,

$$h_X(\ulcorner u(\overline{d})_{\leqslant x} \upharpoonright X \urcorner) \quad = \quad h_X(\ulcorner u_{\leqslant x} \upharpoonright X \urcorner) \quad = \quad e.$$

Thus, not only is $u_{\leqslant d}$ of the form $u_{\leqslant x} \cdot e \cdots d$ but also $u(\overline{d})_{\leqslant d}$ is of the form $u(\overline{d})_{\leqslant x} \cdot e \cdots d$. Hence,

$$\ulcorner u_{\leqslant d} \upharpoonright X \urcorner \quad = \quad \ulcorner u_{\leqslant x} \upharpoonright X \urcorner \cdot e \cdot d \quad = \quad \ulcorner u(\overline{d})_{\leqslant x} \upharpoonright X \urcorner \cdot e \cdot d \quad = \quad \ulcorner u(\overline{d})_{\leqslant d} \upharpoonright X \urcorner.$$

□

To prove Proposition 5.3.1, the first step is to specify the partial function

$$h \quad : \quad \{\text{P-views (of } A \Rightarrow C\text{) ending with O-moves}\} \rightharpoonup \{\text{P-moves in } A \Rightarrow C\}$$

that defines the composite innocent strategy $\sigma_f; \tau_g$.

Definition 5.3.3 For any P-view p of $A \Rightarrow C$ ending with an O-move d_0, $h(p)$ is defined and equal to a P-move a in $A \Rightarrow C$ if and only if

> for some explicitly justified sequence t of the arenas (A, B, C), and for some $n \geqslant 0$, there are B-moves $d_1, \cdots d_n$ with
>
> $$u \quad \equiv \quad \mathbf{u}(p, \sigma, \tau) \quad = \quad t \cdot d_0 \cdot d_1 \cdot d_2 \cdot \cdots \cdot d_n;$$
>
> such that $h_X(\ulcorner u(d_0) \restriction X \urcorner) = a$ where X is the component in which d_n is an O-move, and the same convention as before governs the meaning of h_X.

Proof of Proposition 5.3.1 Suppose $s \in \sigma; \tau$ is a legal position which ends with an O-move d_0. Note that d_0 appears in $\ulcorner s \urcorner = p$ as the last move. For any P-move a, by the preceding analysis, $s \cdot a \in \sigma; \tau$ is equivalent to the following:

(1) the uncovering $u \equiv \mathbf{u}(s, \sigma, \tau)$ is finite; that is to say, for some finite $n \geqslant 0$, there are B-moves $d_1, \cdots, d_n$ such that $u \equiv w \cdot d_0 \cdot d_1 \cdot d_2 \cdots d_n$, for some explicitly justified sequence w of the arenas (A, B, C); and

(2) $(u \restriction X_n) \cdot a \in \rho_{X_n}$ where, as before, we write X_n as the component in which d_n may be regarded as an O-move.

Since d_0 occurs in u and $p = \ulcorner s \urcorner$, by Proposition 5.3.2(i), statement (1) is equivalent to the assertion that $u(d_0) \equiv \mathbf{u}(p, \sigma, \tau)$ is finite, and that $u(d_0) \equiv v \cdot d_0 \cdot d_1 \cdot d_2 \cdots d_n$, for some explicitly justified sequence v of the arenas (A, B, C). Also, since σ and τ are innocent strategies by assumption, (2) is equivalent to

$$h_{X_n}(\ulcorner u \restriction X_n \urcorner) \quad = \quad a,$$

where h_{X_n} is defined as before. By Proposition 5.3.2(ii), we have $h_{X_n}(\ulcorner u(d_0) \restriction X_n \urcorner) = a$. By definition of h, we conclude that (1) and (2) amount precisely to the assertion that $h(\ulcorner s \urcorner)$ is defined, and equal to a. We have thus proved Proposition 5.3.1. $\square$

5.4 $\mathbb{CA}$: a cartesian closed category of computational arenas

At long last we have in place the necessary data for defining a category. The category $\mathbb{CA}$ of computational arenas is defined by the following data:

- Objects are computational arenas.
- Maps between computational arenas A and B are innocent P-strategies of the function space computational arena $A \Rightarrow B$.

The identity map of a computational arena A is just the "tit-for-tat strategy" or "copy-cat strategy" of the arena $A \Rightarrow A$. Consider the two components (or copies) of A in $A \Rightarrow A$. The strategy may be described informally as follows: suppose Opponent has just made the move m in one component, then Player responds by making the same move in the other component. The tit-for-tat strategy is innocent: for any $n \geqslant 0$, and for any legal position s of the form $a_1 \cdot a_1 \cdot a_2 \cdot a_2 \cdots a_n \cdot a_n \cdot a_{n+1}$, the representing partial function of the strategy maps the P-view $\ulcorner s \urcorner$ to a_{n+1}. Composition of strategies as defined in the last subsection is associative.

Product For any computational arenas A and B, the categorical product is just $A \times B$ as defined earlier. The projection map $A \times B \longrightarrow A$ is the following innocent strategy of the arena $(A \times B) \Rightarrow A$ which is another "tit-for-tat" strategy: we label the two copies (or components) of A as $(A_1 \times B) \Rightarrow A_2$. O begins by making an initial A-move in A_2. P responds by making the same initial A-move in A_1. Note that O cannot switch component. For any O-move in A_1, P responds by making the same move in A_2. Thereafter P responds to any O-move by making the same move but in the other A-component. To check the universal property we observe that the arena $C \Rightarrow (A \times B)$ is just $(C \Rightarrow A) \times (C \Rightarrow B)$. So given maps $f : C \longrightarrow A$ and $g : C \longrightarrow B$, the pairing $[f, g] : C \longrightarrow A \times B$ is defined by the disjoint union of the respective strategies.

The terminal object $\mathbf{1}$ is the empty computational arena: it has neither question nor answer. For any computational arena A, the unique innocent strategy of the arena $A \Rightarrow \mathbf{1}$ (which is just the empty arena $\mathbf{1}$) is the singleton set $\{\epsilon\}$ as defined by the empty (as a graph) innocent partial function.

Function space For computational arenas A, B, and C, it is easy to see that the arenas $(C \times A) \Rightarrow B$ and $C \Rightarrow (A \Rightarrow B)$ are isomorphic (the respective arenas are exactly the same as pictures). Hence

$$\mathbb{CA}(C \times A, B) \quad \cong \quad \mathbb{CA}(C, A \Rightarrow B)$$

is an isomorphism natural in A, B and C. We can therefore conclude that the category of computational arenas and innocent strategies is cartesian closed.

It is worth observing that many strategies that denote PCF-terms are "tit-for-tat strategies" in some appropriately general sense. In these strategies, Player's response simply consists in copying Opponent's move from one component of the arena to another. We have already seen two examples; namely, the identity and projection maps. Another is the evaluation map:

$$(A \Rightarrow B) \times A \xrightarrow{\text{ev}} B$$

for arenas A and B. The map ev is a history-free (and hence innocent) strategy in the arena $(A \Rightarrow B) \times A \Rightarrow B$ which has two components of the subarena A, one dual to the other; similarly for the subarena B. The strategy ev may be described succinctly as follows:

> "if O's move is m in some component of A (respectively B) then P's response is m in the dual component of A (respectively B)".

5.5 $\mathbb{CA}$ as an enriched category

In this subsection we show that $\mathbb{CA}$ is the underlying category of a category (which we continue to call $\mathbb{CA}$) enriched over the category of dI-domains. (For enriched category theory see [Kel82].) For computational arenas B and C, $\mathbb{CA}(B, C)$ is the set of all innocent strategies of the arena $B \Rightarrow C$. We consider the structure which this set naturally carries. Since we can readily identify $\mathbb{CA}(B, C)$ with $\mathbb{CA}(\mathbf{1}, B \Rightarrow C)$ we can restrict attention to sets of the form $\mathbb{CA}(\mathbf{1}, A)$ for any arena A. We write $\underline{A}$ for $\mathbb{CA}(\mathbf{1}, A)$ the set of all innocent strategies of a computational arena A.

For the sake of completeness we introduce some basic domain-theoretic notions. A *complete partial order* (CPO) is a poset which has a least element and joins of all directed subsets. A subset X of a poset D is said to be *consistent* if X has an upper bound in D. A CPO D is *consistently complete* if any of the following equivalent conditions are satisfied:

(1) every pairwise consistent pair of elements of D has a join in D;

(2) every consistent subset of D has a least upper bound (lub);

(3) every subset of D has a greatest lower bound (glb),

An element d of a CPO D is said to be *prime* if for every subset X of D such that $\bigsqcup X$ exists and if $d \sqsubseteq \bigsqcup X$, then there is some element $x \in X$ such that $d \sqsubseteq x$. The element d is said to be *compact* if the above condition holds for X ranging over only directed subsets of D. A CPO D is said to be *algebraic* if for every element d of D, the collection of all compact elements of D dominated by d is directed, and its lub is precisely d. Further, D is said to be *ω-algebraic* if D has countably many compact elements. By a *Scott domain*, we mean a consistently complete, ω-algebraic CPO. A consistently complete CPO D is said to be *prime algebraic* if every element d of D is the lub of all prime elements dominated by d. A *dI-domain*[10] is a consistently complete, algebraic CPO that is prime algebraic and satisfies axiom (I): every compact element dominates only finitely many elements.

We say that a legal position s of an arena A is *innocent* if there is an innocent strategy (of A) of which s is an element. Clearly not every legal position is innocent: consider the legal position

$$[\cdot(_1\cdot[_{1.1}\cdot]_1 \cdot [_{1.1}\cdot]_2$$

of the arena $(\iota \Rightarrow \iota) \Rightarrow \iota$ whose questions are annotated according to the convention introduced in Remark 4.3.2. Since innocent strategies are determined by innocent functions, it is easy to see that a legal position s (which ends in an O-move) is innocent if and only if there is an innocent function f such that $s \in \sum_f \ulcorner s \urcorner$. Given an innocent legal position s of an arena A, we define

$$\boldsymbol{\sigma}[s] \quad \stackrel{\text{def}}{=} \quad \text{the least innocent strategy containing } s.$$

Suppose $s \equiv m_1 \cdot m_2 \cdots m_{2n+1}$. It is easy to see that $\boldsymbol{\sigma}[s]$ is σ_g where the (partial) function g is defined by the following graph

$$\{\, (\ulcorner m_1 \cdot m_2 \cdots m_{2l+1} \urcorner, m_{2l+2}) : 0 \leqslant l < n \,\}.$$

It is routine to verify that f thus defined is innocent.

Example 5.5.1 Note that $\boldsymbol{\sigma}[s]$ may contain legal positions of length greater than that of s. Just take s to be the legal position (together with its auxiliary sequence of justification indices):

$$[\cdot(_1\cdot[_{1.1}\cdot]\cdot)\cdot] \qquad 0 \cdot 1 \cdot 2 \cdot 3 \cdot 2 \cdot 1$$

of the arena $(\iota \Rightarrow \iota) \Rightarrow \iota$. The least innocent strategy generated by s is defined by the following innocent function:

$$\begin{array}{lcl} [& \mapsto & (_1 \\ [\cdot(_1\cdot[_{1.1} & \mapsto &] \\ [\cdot(_1\cdot) & \mapsto &] \end{array}$$

Note that $\boldsymbol{\sigma}[s]$ contains the following family of legal positions: for each $n \geqslant 0$,

$$[\cdot(_1\cdot\underbrace{[_{1.1}\cdot]\cdot \quad \cdots \quad \cdot[_{1.1}\cdot]}_{2n}\cdot)\cdot] \qquad 0\cdot 1\cdot \underbrace{2\cdot 3\cdot 2\cdot 5\cdot \quad \cdots \quad \cdot 2\cdot 2n+1}_{2n}\cdot 2\cdot 1;$$

so the length of legal positions in $\boldsymbol{\sigma}[s]$ is unbounded.

[10] We do not include ω-algebraicity in the definition of dI-domain.

For any P-view p that ends with an O-move, and for any P-move a, it is easy to see that if $p \cdot a$ is a legal position then it is innocent. The innocent function $f_{\boldsymbol{\sigma}[p \cdot a]}$, *qua* graph, is the collection of ordered pairs (q, b) such that b is a P-move and $q \cdot b \leqslant p \cdot a$. Since P-views are P-view invariant (Lemma 4.2.5), any two pairs in the collection which agree in the first component necessarily agree in the second.

Let S be a set of innocent strategies of A bounded, say, by τ. By Lemma 5.2.4, for each $\sigma \in S$, $f_\sigma \subseteq f_\tau$; and so, $\bigcup_{\sigma \in S} f_\sigma \subseteq f_\tau$. Hence $F \stackrel{\text{def}}{=} \bigcup_{\sigma \in S} f_\sigma$ is a partial function. It is straightforward to verify that F is innocent. Clearly $\bigsqcup S$ is the strategy defined by F. (Note that in general $\bigsqcup S$ is not $\bigcup_{\sigma \in S} \sigma$, though of course $\bigcup_{\sigma \in S} \sigma \subseteq \bigsqcup S$.) We have shown that $\underline{A}$ is bounded complete.

Lemma 5.5.2 *(i) An innocent strategy $\sigma \in \underline{A}$ is compact if and only if its representing function f_σ is a finite graph.*

(ii) An innocent strategy $\sigma \in \underline{A}$ is prime if and only if it is $\boldsymbol{\sigma}[p \cdot a]$ for some P-view p (which ends with an O-move) and for some P-move a such that $p \cdot a$ is a legal position.

(iii) $\underline{A}$ is prime algebraic.

Proof We prove (ii) and (iii) for an illustration. First observe that for any innocent strategy σ of A,

$$\bigsqcup_{(p,a) \in f_\sigma} \boldsymbol{\sigma}[p \cdot a]$$

is defined and is equal to σ. If σ is prime then $\sigma \subseteq \boldsymbol{\sigma}[p \cdot a]$ for some pair (p, a) in f_σ. But in fact $\sigma = \boldsymbol{\sigma}[p \cdot a]$, since $\boldsymbol{\sigma}[p \cdot a]$ is the least strategy that contains $p \cdot a$. The other direction is now immediate; so is (iii). □

Supposing the arena A is countable (*i.e.* A has countably many questions and answers), then A has only countably many P-views. Since compact innocent strategies are finite subsets (of a certain kind) of the countable set $\{\text{P-views}\} \times \{\text{P-moves}\}$, there are only countably many such strategies. Therefore $\underline{A}$ is ω-algebraic. Finally we observe that axiom (I) is trivially satisfied.

To summarize we have shown that

Proposition 5.5.3 *(i) The collection $\underline{A}$ of innocent strategies ordered by inclusion is a dI-domain.*

(ii) In particular, if A is a PCF*-type, then $\underline{A}$ is ω-algebraic.* □

We conclude from the preceding proposition that each $\mathbb{CA}(B, C)$ carries the structure of a dI-domain. It follows easily from the definition of identities, composition, products and function spaces that the corresponding maps (natural isomorphisms in the last two cases)

$$\mathbf{1} \longrightarrow \mathbb{CA}(A, A)$$

$$\mathbb{CA}(A, B) \times \mathbb{CA}(B, C) \longrightarrow \mathbb{CA}(A, C)$$

$$\mathbb{CA}(C, A) \times \mathbb{CA}(C, B) \longrightarrow \mathbb{CA}(C, A \times B)$$

$$\mathbb{CA}(C \times A, B) \longrightarrow \mathbb{CA}(C, A \Rightarrow B)$$

are continuous maps of dI-domains (but they are not necessarily stable). Hence we have the main result of this section:

Theorem 5.5.4 *The category $\mathbb{CA}$ of computational arenas and innocent strategies is a cartesian closed category which is enriched over dI-domains.* □

5.6 Recursive strategies

Now we consider computational arenas which are in some sense recursively presented. For simplicity we suppose that the forest of questions of A is a finite tree. (We could proceed with an enumeration of a countable tree, but the definition is more cumbersome and we have no need of that generality.)

Definition 5.6.1 Let A be a computational arena with a finite tree of questions and a countable (finite or denumerable) set of answers. Then a *recursive presentation* of A consists of an enumeration of the countable set of answers corresponding to each question: for each $q \in \mathsf{Qn}(A)$ we have a bijection

$$\mathsf{qn}^{-1}(q) \quad \longrightarrow \quad \mathbb{N}$$

or

$$\mathsf{qn}^{-1}(q) \quad \longrightarrow \quad \mathbb{N}_n = \{ i : i < n \}.$$

We call a computational arena equipped with a recursive presentation a *recursively presented computational arena* (RPCA).

Example 5.6.2 (i) The computational arenas $\mathbb{N}$ and $\mathbb{B}$ of Example 4.3.1 have natural recursive presentations. ($\mathbb{N}$ also has perverse recursive presentations.)

(ii) Clearly the computational arena $\mathbf{1}$ is trivially recursively presented. And if A and B are recursively presented then so are $A \times B$ and $A \Rightarrow B$ in an obvious way.

We could consider a category with objects RPCAs and maps innocent strategies. This is hardly worth naming as it is equivalent to a subcategory of $\mathbb{CA}$: the forgetful functor is full and faithful. Rather we want to use the enumerations (codings) of answers to enable us to talk of recursive strategies. Now given an RPCA A we can concoct natural codings for P-moves and for P-views in A. (Of course the coding includes information about the justification indices.) If a is a P-move (u a P-view) we write $\#a$ ($\#u$) for its numerical code. Recall that an innocent strategy σ is completely determined by a representing innocent function f_σ which maps P-views to P-moves. We can code f_σ as a partial numerical function ϕ_σ where

$$f_\sigma(u) = a \qquad \Longleftrightarrow \qquad \phi_\sigma(\#u) = \#a.$$

(If we are sensible the set of codes for P-view will be recursive; and we may as well assume ϕ_σ not defined on numbers which do not code P-views.) The set of partial functions from $\mathbb{N}$ to $\mathbb{N}$, in which the codes ϕ_σ for strategies lie, is so important that we need a special symbol for it; we write $\mathbb{P}$ for the set of partial numerical functions.

Definition 5.6.3 An innocent strategy σ in a recursively presented arena is *recursive* just when ϕ_σ is a partial recursive function.

Clearly the identity or tit-for-tat strategy in $A \Rightarrow A$ is recursive for any RPCA A. Also the composition of two recursive strategies is recursive. This reflects the fact that the proof of Proposition 5.3.1 is constructive. (The reader need only understand this in the intuitive sense: if σ, τ are innocent strategies for $A \Rightarrow B$ and $B \Rightarrow C$ respectively, then $f_{\sigma;\tau}$ can be constructed from f_σ and f_τ.) In fact one can read off from Definition 5.3.3 the proof of the following precise result which we use later.

Proposition 5.6.4 *Given* RPCAs *A, B and C there is a recursive operator*[11] *M of two arguments $M : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$ such that if $\sigma : A \longrightarrow B$ and $\tau : B \longrightarrow C$ are innocent strategies then*

$$M(\phi_\sigma, \phi_\tau) \quad = \quad \phi_{\sigma;\tau}.$$

□

[11] In the sense of [Rog67, p. 148].

It follows in particular that we can define a category using recursive strategies. Then category $\mathbb{RA}$ of *recursive arenas* is defined by the following data:

- objects are recursively presented computational arenas,
- maps between RPCAs A and B are recursive strategies in $A \Rightarrow B$.

The identity, composition, product and function space arenas are as in $\mathbb{CA}$. So $\mathbb{RA}$ is a cartesian closed category.

Finally perhaps we should confess to a degree of overkill in our description of the category $\mathbb{RA}$. Consider the computational arenas $\mathbb{N}$ as in Example 4.3.1 and $\mathbb{N}_n$ $(n \geqslant 0)$ where $\mathbb{N}_n$ is like $\mathbb{N}$ save that only replies $i < n$ are permitted. (So $\mathbb{N}_2$ is isomorphic to $\mathbb{B}$ of Example 4.3.1.) These objects have canonical recursive presentations. Close this collection of objects under the standard terminal object, product and function space in $\mathbb{RA}$. Then any object in $\mathbb{RA}$ is isomorphic to an object which results. (This is because we insisted on a finite tree of questions.) Thus $\mathbb{RA}$ as we defined it is (up to equivalence) a very simple category indeed.

6 Context lemma for $\mathbb{CA}$

The standard notion of observables for $\mathbb{CA}$ We recall the standard notion of observables for the category $\mathbb{CA}$ of computational arenas and innocent strategies (see the discussion after Proposition 3.2.1). If an innocent strategy σ of the arena ι has an answer (n say) to O's opening question – in which case, σ is the set $\{$ "$[$", "$[\cdot n$" $\}$ – then we write $\sigma \Downarrow n$; otherwise we write $\sigma\Uparrow$. We shall write $\sigma\Downarrow$ to mean that for some n, $\sigma \Downarrow n$. Following the same pattern as Example 3.1.2, for each arena A and each innocent strategy f of the arena $A \Rightarrow \iota$, we define

$$R_f \quad \stackrel{\text{def}}{=} \quad \{\, a : \mathbf{1} \longrightarrow A \mid a; f\Downarrow \,\}.$$

The collection $\mathcal{O}_A$ of observables of type A consists of all subsets of $\mathbb{CA}(\mathbf{1}, A)$ of the form R_f. Since for any $g : B \longrightarrow \iota$ and $f : A \longrightarrow B$, $f^* R_g$ is just $R_{f;g}$, the association $A \mapsto \mathcal{O}_A$ equips $\mathbb{CA}$ with a notion of observables (which we shall refer to as the *standard* such notion). As $\mathbb{CA}$ is cartesian closed, the associated notions of observational preorder $\lesssim$ and observational quotient $\hat{\mathbb{CA}}$ (see §3.1) are well-defined.

6.1 Context lemma for $\mathbb{CA}$

It is a familiar fact that the context lemma holds for PCF. The main result of this section is that the context lemma also holds for the category $\mathbb{CA}$ of computational arenas and innocent strategies with respect to the standard observational preorder. Recall (from §3.3) that this is the same as saying that $\mathbb{CA}$ is order-extensional.

Theorem 6.1.1 (Context lemma) *The category of arenas and innocent strategies is order-extensional with respect to the standard observational preorder. That is to say, for any innocent strategies σ, τ of the arena $A \Rightarrow B$,*

$$\sigma \lesssim \tau \quad \iff \quad a; \sigma \lesssim a; \tau \quad \textit{for all innocent strategies } a \textit{ of arena } A.$$

$\square$

One direction ("$\Rightarrow$") of the proof of this theorem is almost obvious. A careful treatment of the other direction requires a somewhat complicated combinatorial analysis. We devote the rest of the section to the proof. The reader who is willing to take the context lemma on trust is advised to skip its proof on first reading and move on to the next section.

6.2 Proof of the context lemma for $\mathbb{CA}$

W.l.o.g. we shall assume that ι is the only program type. We first prove the direction "$\Rightarrow$" of the context lemma. The argument is essentially a consequence of the cartesian closed structure of the category $\mathbb{CA}$. For any innocent strategies σ and τ of the arena $A \Rightarrow B$, take any maps $x : \mathbf{1} \longrightarrow A$ and $\delta : B \longrightarrow \iota$. By the universal property of the function space construction, we have the following equation of maps:

$$
\begin{array}{lc}
 & \mathbf{1} \xrightarrow{\quad x \quad} A \xrightarrow{\qquad\quad \sigma \qquad\quad} B \\[2ex]
= & \mathbf{1} \xrightarrow{\quad \sigma \quad} (A \Rightarrow B) \xrightarrow{\langle\, \mathsf{id}, !; x \,\rangle} (A \Rightarrow B) \times A \xrightarrow{\quad \mathrm{ev} \quad} B.
\end{array}
$$

Let $\alpha : (A \Rightarrow B) \longrightarrow \iota$ be the following composition of maps:

$$
(A \Rightarrow B) \xrightarrow{\langle\, \mathsf{id}, !; x \,\rangle} (A \Rightarrow B) \times A \xrightarrow{\quad \mathrm{ev} \quad} B \xrightarrow{\qquad \delta \qquad} \iota.
$$

Clearly $x; \sigma; \delta = \sigma; \alpha$. By exactly the same reasoning, we also have $x; \tau; \delta = \tau; \alpha$. Hence if $\sigma; \alpha\Downarrow$ implies $\tau; \alpha\Downarrow$ for every $\alpha : (A \Rightarrow B) \longrightarrow \iota$, then $x; \sigma; \delta\Downarrow$ implies $x; \tau; \delta\Downarrow$ for each $x : \mathbf{1} \longrightarrow A$ and $\delta : B \longrightarrow \iota$ as required.

The other direction "$\Leftarrow$" is relegated to the Appendix.

Part III

A fully abstract and universal game model

7 A fully abstract dialogue game model of PCF

In this section we show how PCF may be interpreted in the category $\mathbb{CA}$ of computational arenas and innocent strategies. This interpretation is computationally adequate and the derived interpretation in the observational quotient $\hat{\mathbb{CA}}$ is order (or inequationally) fully abstract for PCF. Full abstraction is obtained as a consequence of a strong definability result: not only are all compact innocent strategies (of PCF-types) definable in **P**, but the valuation map actually gives an order-isomorphism between syntax (a class of finite canonical forms of a PCF-variant called **P** ordered by the standard Ω-matching) and semantics (compact innocent strategies ordered by set inclusion). The language **P** is just PCF extended by a family of definition-by-cases constructs. We conclude this section by examining in some detail two instructive examples: the innocent strategies defined by a type-2 and a type-3 functional respectively.

7.1 Semantics of PCF in $\mathbb{CA}$

PCF-types For any PCF-type A we define the interpretation $[\![A]\!]$ as a computational arena recursively as follows:

$$\begin{array}{rcl} [\![o]\!] & \stackrel{\text{def}}{=} & \mathbb{B}, \\ [\![\iota]\!] & \stackrel{\text{def}}{=} & \mathbb{N}, \\ [\![A_1 \Rightarrow A_2]\!] & \stackrel{\text{def}}{=} & [\![A_1]\!] \Rightarrow [\![A_2]\!]; \end{array}$$

where $\mathbb{N}$ and $\mathbb{B}$ are the natural numbers and boolean computational arenas defined in Example 4.3.1. Note that the forest of questions of an arena which is a PCF-type is an (inverted) finite tree *i.e.* a finite poset with a unique top element (the initial question) such that the upper set of each element is a finite linear order. Further all questions are just "copies" of the initial question at program type, and answers are "copies" of the natural numbers and / or booleans.

Convention. In the following we shall often confuse syntax (of both PCF-types and terms) with semantics and write the interpretation of a PCF-type A also as A (and similarly for PCF-terms s) provided it is safe to do so. If there is a possibility of confusion, we shall denote the dialogue game interpretation as $[\![A]\!]^{\mathbb{CA}}$ (and $[\![s]\!]^{\mathbb{CA}}$). Thus we can reserve $\mathbb{N}$ for the usual natural numbers of mathematics.

PCF-terms For each PCF-type A, the (intensional) domain of type A is the set of global sections of the arena A. As we have seen earlier, this is just the collection of innocent strategies of the arena A ordered by set inclusion. The domain of type ι is the standard flat CPO of natural numbers: the least element is the "empty strategy" — one that has no response to O's opening question. The denotation of a natural number n is the strategy that returns the answer "$]^{\iota}_{n}$" to O's opening question "$[^{\iota}$". (We shall write "$]^{\iota}_{n}$" variously as "$]_{n}$" or simply n whenever the type information is clear from the context.) We shall just state an elementary observation.

Proposition 7.1.1 *This interpretation of* PCF *is standard in the sense of Plotkin [Plo77].* □

The basic arithmetic constants are straightforwardly interpreted as innocent strategies. We consider the interpretation of the successor and conditional for illustration. Questions of the respective arenas are annotated with their respective occurrences as in Figure 7. The innocent strategy $[\![\mathsf{succ}]\!] : (\iota, \iota)$ is defined by the following innocent function: for $n \geqslant 0$,

$$\begin{cases} [& \mapsto & (_1 \\ [\cdot (_1 \cdot)_n & \mapsto &]_{n+1}. \end{cases}$$

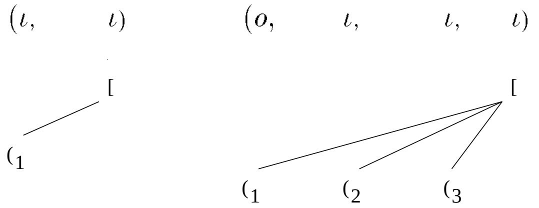

Figure 7: Trees of questions of arenas (ι, ι) and (o, ι, ι, ι).

The innocent strategy $[\![\mathsf{cond}^\iota]\!] : (o, \iota, \iota, \iota)$ is defined by the following innocent function: for n ranging over the natural numbers,

$$\begin{cases} [& \mapsto & (_1 \\ [\cdot (_1\cdot)_{\mathsf{t}} & \mapsto & (_2 \\ [\cdot (_1\cdot)_{\mathsf{f}} & \mapsto & (_3 \\ [\cdot (_1\cdot)_{\mathsf{t}} \cdot (_2\cdot)_n & \mapsto &]_n \\ [\cdot (_1\cdot)_{\mathsf{f}} \cdot (_3\cdot)_n & \mapsto &]_n. \end{cases}$$

The interpretation of λ-abstraction and application is completely determined by the cartesian closed structure of the category $\mathbb{CA}$. We regard this as standard and refer the reader to [LS86, Cro93] for a systematic treatment.

Remark 7.1.2 Any partial function can be numeralwise represented in $\mathbb{CA}$ in the way that successor can. For simplicity we restrict to the case of one variable. If $\phi : \mathbb{N} \longrightarrow \mathbb{N}$ is a partial function we have an innocent strategy $[\![\phi]\!]$ defined by the innocent function:

$$\begin{cases} [& \mapsto & (_1 \\ [\cdot (_1\cdot) & \mapsto &]_{\phi(n)} \quad \text{whenever } \phi(n) \text{ is defined.} \end{cases}$$

(ii) For any partial function (and in particular for successor) there are many different choices of innocent strategies which will numeralwise represent ϕ (in the strong sense). We call $[\![\phi]\!]$ defined in (i) the *standard representation* of ϕ.

7.2 Fixed points

We give two different presentations of the interpretation of fixed-point operators as innocent strategies. The first, a Tarski-Knaster style argument, follows more or less standard lines in denotational semantics. The second highlights the observation that the family of innocent strategies that correspond to fixed-point operators behave in a "parametric" way: they do nothing more than copying moves in a highly uniform way. Our account of the second approach is informal. While it may seem more tedious to describe than the first, the idea is actually simpler!

A standard denotational approach We say that a cartesian closed category has fixed points if it has a family of maps $\mathsf{Y}^A : (A \Rightarrow A) \longrightarrow A$ for each object A satisfying the commuting diagram in Definition 2.2.1(ii). For each type A the commuting fixed-point diagram is the following equation

$$\mathsf{Y}^A \quad = \quad \lambda f : A \Rightarrow A. f(\mathsf{Y}^A f).$$

Take a cartesian closed category $\mathbb{C}$ which is enriched over CPOs (so that not just composition but also the respective natural isomorphisms which characterize products and function spaces are continuous). We refer to the enriching ordering of the homsets as the *given ordering*. In such a category, fixed-point operators Y^A may be interpreted as the least (with respect to the given ordering) fixed point of the following simply-typed λ-term:

$$\mathcal{F} \quad = \quad \lambda F : (A \Rightarrow A) \Rightarrow A . \lambda f : A \Rightarrow A . f(Ff)$$

which has type $((A \Rightarrow A) \Rightarrow A) \Rightarrow (A \Rightarrow A) \Rightarrow A$. Since $\mathbb{C}$ is cartesian closed, $\mathcal{F}$ has interpretation as a map from $(A \Rightarrow A) \Rightarrow A$ to itself. Writing the details out in full, we define

$$\begin{aligned} \mathcal{F}^0 &\stackrel{\text{def}}{=} \perp^{(A \Rightarrow A) \Rightarrow A} \\ \mathcal{F}^{n+1} &\stackrel{\text{def}}{=} \lambda f : A \Rightarrow A . \underbrace{f(\cdots(f}_{n+1} \perp^A) \cdots), \end{aligned}$$

where $\perp^A$ is the least element of the homset $\mathbb{C}(\mathbf{1}, A)$. By assumption the lub $\bigsqcup_{n \in \omega} \mathcal{F}^n$ with respect to the given ordering (which we write as $\mathcal{Y}^A$) is a well-defined map $(A \Rightarrow A) \longrightarrow A$. For any map $g : (A \Rightarrow A) \longrightarrow A$ and $a : A \longrightarrow A$, we write application $g \cdot a$ to mean $\langle \overline{g}, \overline{a} \rangle ; \mathrm{ev}$. (We shall blur the distinction between a map g and its function space transpose $\overline{g}$.) In such a category, application is continuous

$$(\bigsqcup g_n) \cdot a \quad = \quad \bigsqcup (g_n \cdot a)$$

because pairing, transpose and composition are. Note that we do not need order-extensionality.

To see that the fixed-point diagram commutes in $\mathbb{C}$, we have

$$\begin{aligned} \lambda f : A \Rightarrow A . f \cdot (\mathcal{Y}^A \cdot f) &= \lambda f : A \Rightarrow A . \bigsqcup \underbrace{f(\cdots(f}_{n+1} \perp) \cdots) \\ &= \bigsqcup \lambda f : A \Rightarrow A . \underbrace{f(\cdots(f}_{n+1} \perp) \cdots) \\ &= \mathcal{Y}^A . \end{aligned}$$

Hence $\mathbb{C}$ has fixed points.

Suppose $\mathbb{C}$ is equipped with an observational preorder (defined with respect to a notion of observables). By Proposition 3.1.6 the observational quotient $\hat{\mathbb{C}}$ is an order-enriched category which inherits the cartesian closed structure from $\mathbb{C}$. However the enriching structure need not be a CPO. Nonetheless since the observational preorder is preserved by composition, the fixed-point diagram in $\hat{\mathbb{C}}$ commutes if and only if the following equation (of equivalence classes) holds:

$$[\mathcal{Y}_A] \quad = \quad [\Delta_{A \times A} ; \mathsf{id} \times \mathcal{Y}_A ; \mathrm{ev}].$$

This equation follows trivially from the fixed-point equation in $\mathbb{C}$.

To summarize we have shown that:

Proposition 7.2.1 *Any cartesian closed category $\mathbb{C}$ enriched over CPOs has fixed points. With respect to any observational preorder the derived quotient $\hat{\mathbb{C}}$ also has fixed points.* □

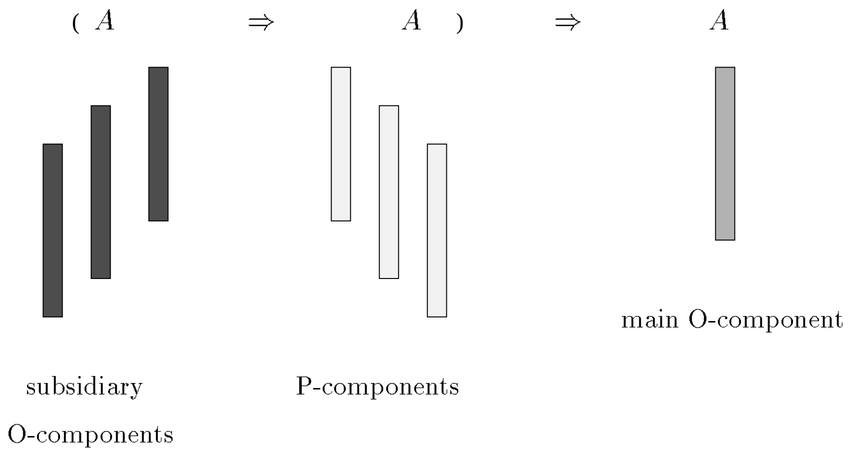

Figure 8: Components of $\mathsf{Y}^A : (A \Rightarrow A) \Rightarrow A$.

As an immediate corollary we can conclude that both $\mathbb{CA}$ and its observational quotient $\hat{\mathbb{CA}}$ have fixed points.

Fixed-point operators as uniform strategies For any arena (not just those which are PCF-types) A we describe a strategy in $(A \Rightarrow A) \Rightarrow A$. First we need to distinguish between moves in the different copies of A as in Figure 8:

- *The main O-component.* These are moves hereditarily justified by the opening O-move in A (on the right) only. That is to say, they are not hereditarily justified by P-moves which are opening moves in the copy of A in the middle.
- *The P-components.* These are moves which are hereditarily justified by an instance of an opening move in A made by P, but not by opening A-moves made by P dependent on it. There may be many such.
- *Subsidiary O-components.* Moves hereditarily justified by a sequence of three initial moves in A [·(·[.

In a game according to the strategy we describe there will be a correspondence between O- and P-components. The first P-components to occur is the dual of the main O-component. The others in order are the duals of the subsidiary O-components in order. At any stage after P has moved the duals will be copies of each other. The strategy can be succinctly described as follows: suppose O has just moved:

- *Case 1.* Opening move: we copy to create first P-component.
- *Case 2.* O opens a new subsidiary component: we copy and create new P-component.
- *Case 3.* O moves in some existing O- / P-component: we copy the move in the dual P- / O-component.

Arguing inductively it is easy to see this makes sense. The question is:

- is it an innocent strategy?
- does it satisfy the fixed-point equation / diagram?

$$
\begin{tikzcd}
 & (A \Rightarrow A) & (A \Rightarrow A) \arrow[r, "1"] & (A \Rightarrow A) & (A \Rightarrow A) & \\
(A \Rightarrow A) \arrow[r, "\Delta"] & \times & \times & \times & \times \arrow[r, "\mathrm{ev}"] & A \\
 & (A \Rightarrow A) & (A \Rightarrow A) \arrow[r, "\mathbf{Y}"] & A & A &
\end{tikzcd}
$$

Figure 9: Composition of maps $\Delta; \mathbf{1} \times \mathbf{Y}; \mathrm{ev}$.

The usual situation after an O-move is that the play has been $p \cdot (\cdot r \cdot [$ — where "(" explicitly justifies "[", or $p \cdot (\cdot r \cdot)$ — where "(" explicitly justifies ")"; and so the P-view is $\ulcorner p \urcorner \cdot (\cdot[$ or $\ulcorner p \urcorner \cdot (\cdot)$ where the moves $(\cdot[$ or $(\cdot)$ are in the same component. Then the P-move "(" displayed is a copy of an O-move "$[_1$" in the dual component, and this O-move "$[_1$" occurs at the end of $\ulcorner p \urcorner$; hence the copy reply "$(_1$" or "$]_1$" is a legitimate move *independent* of the way we reached the particular P-view, so we can safely reply $\ulcorner p \urcorner \cdot (\cdot[\cdot(_1$ or $\ulcorner p \urcorner \cdot (\cdot)\cdot]_1$ as appropriate on the basis of the view.

The composition

$$(A \Rightarrow A) \xrightarrow{\Delta} (A \Rightarrow A) \times (A \Rightarrow A) \xrightarrow{\mathbf{1} \times \mathbf{Y}} (A \Rightarrow A) \times A \xrightarrow{\mathrm{ev}} A$$

can be "represented" in parallel composition and hiding form as in Figure 9. We draw some copying paths through the picture in Figure 10.

It does not seem accidental that these three paths (in Figure 10) "partition" the whole picture. In the composed game we use the terminology (main O-component, P-components, subsidiary components, the dual of a component) already introduced. Now we can tie down the behaviour of the composed strategy by making the following observations.

1. After the opening O-move in the main O-component, P copies along path (1) (with reference to Figure 10) to give a reply in the first P-component (the dual of the main O-component). Thereafter any O-move in either of the two components is answered by copying along path (1) (in either direction) to give a P-reply (which is just a "copy") in the dual component.

2. After the first O-move (if any) in the first subsidiary O-component, P copies along path (2) to give a reply in the second P-component (its dual). Thereafter any O-move in either of these two components is answered by copying along path (2) (in either direction) to give a P-reply (just a "copy") in the dual component.

Furthermore exactly the same applies *mutatis mutandis* in the case of any subsidiary O-component which begins with an O-move justified by the opening P-move in the *first* P-component. [Perhaps it is worth noting that each of these further O-components involves P opening a fresh version of $\mathbf{Y}$ on the hidden sketch pad *i.e.* the "returning" portion of path (2) involves a fresh version of $\mathbf{Y}$ in each new case.]

3. After the first O-move (if any) in a subsidiary O-component not as in 2, P copies along path (3) to give a reply in a new P-component (its dual). Thereafter any O-move in either of these two components is answered by copying along path (3) (in either direction) to give P-reply (just a "copy") in the dual component.

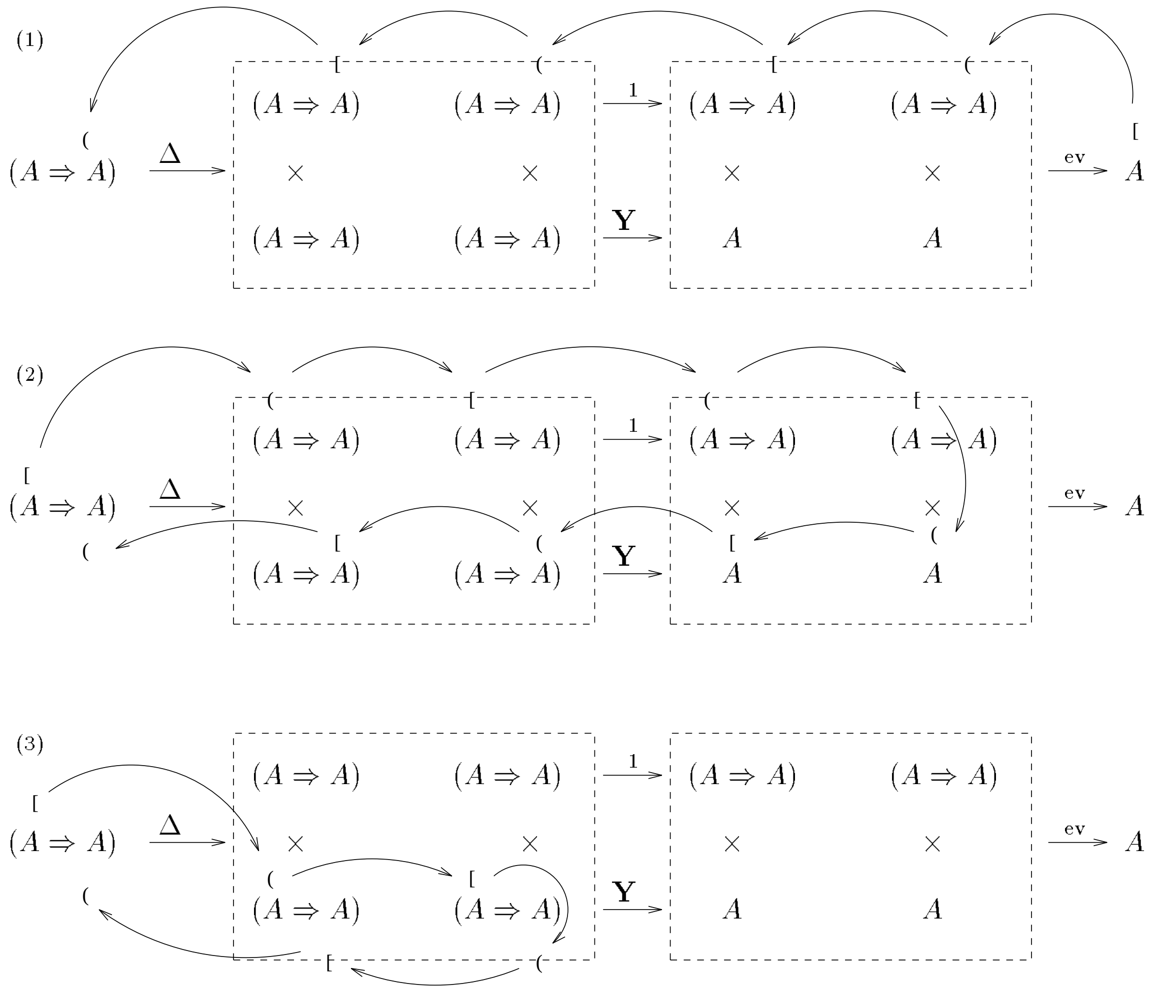

Figure 10: Copying paths.

It follows from the three observations that the composed strategy behaves exactly like $\mathbf{Y}$: that is

$$\begin{tikzcd}
(A \Rightarrow A) \times (A \Rightarrow A) \arrow[r, "\mathbf{1} \times \mathbf{Y}"] & (A \Rightarrow A) \times A \arrow[d, "\mathrm{ev}"'] \\
A \Rightarrow A \arrow[u, "\Delta"] \arrow[r, "\mathbf{Y}"'] & A
\end{tikzcd}$$

commutes.

Remark 7.2.2 We have seen two ways of representing innocent strategies formally: first as a collection of paths (legal positions) of the corresponding game tree, and secondly as an innocent function. Even for relatively simple PCF-terms, precise description of their denotations as innocent strategies in either style very quickly becomes unwieldy and opaque. Neither style is optimised for capturing the *uniform*[12] or parametric nature of (most) innocent strategies which are denotations of PCF-terms. Intuitively a uniform strategy is one whose behaviour does not depend on the type to which it is instantiated. For example the identity strategy of the arena $A \Rightarrow A$ copies moves from one component of A to its dual component regardless of the complexity of the arena A. It would be highly desirable if an expressive calculus which lent itself to a succinct description of such uniform strategies were available.

Using the preceding analysis of the interpretation of fixed-point operators as a guide, we seek a calculus for describing strategies with the following capabilities:

1. generating new names,

2. copying moves,

3. communication in the style of message passing,

3. private links (so that there is no ambiguity as to which pending question we are answering),

4. branching constructor (corresponding to definition-by-cases),

5. replication.

In addition there should be a polymorphically typed-version of such a calculus to handle uniform strategies. It is worth noting that despite being models of higher-type functions, moves of our games are just copies of program-type objects. Hence if moves are the only things we communicate, a process calculus which passes program-type signals (*e.g.* names) as opposed to more complicated objects (*e.g.* processes) would suffice as a first attempt. This naturally brings to mind the π-calculus [MPW92]. We have a way of expressing innocent strategies as terms of an appropriately sorted polyadic π-calculus (see [Mil91]). This representation reflects the behaviour of the innocent strategy exactly. At the same time this gives an apparently new encoding of PCF (and hence the simply-type λ-calculus) in the π-calculus. This and further developments are presented in [HO95].

[12] The connotation here is with parametric (as opposed to *ad hoc*) polymorphism in the sense of Strachey (see *e.g.* [Str67, Rey83]).

7.3 Characterization of compact innocent strategies

A major result in this section characterizes compact innocent strategies in terms of (a class of) finite canonical forms (FCF) of a language which is essentially PCF extended by a family of definition-by-cases constructs. This characterization is very tight: there is a one-to-one correspondence between compact innocent strategies and FCFs. More precisely the valuation map which takes FCFs to compact innocent strategies gives an isomorphism between syntax – ordered by the Ω-match ordering – and semantics. An important consequence of this correspondence is the full abstraction result for PCF.

PCF extended by definition-by-cases We introduce a language called **P** (for Platek or Plotkin) which is obtained from PCF by adding a family of definition-by-cases constructs. Each such construct is indexed by a natural number (which we shall often omit in the interest of readability) corresponding to the number of cases considered by the definition. Formally the language **P** is defined by adding the following typing rule to those that define PCF: for each program type β and $k \geqslant 0$:

$$\frac{t_0 : \beta \quad \cdots \quad t_k : \beta \qquad s : \iota}{\mathsf{case}_k^\beta \ s = [0 \Rightarrow t_0 | 1 \Rightarrow t_1 | \cdots | k \Rightarrow t_k] : \beta}$$

Notation. There is no harm in assuming that ι is the only program type and we shall do so in the rest of this section. Also we shall use two kinds of shorthand freely.

- First we write $\mathsf{case}\ s[t_0 | \cdots | t_k]$ to mean the notationally cumbersome
$$\mathsf{case}_k^\iota \ s = [0 \Rightarrow t_0 | 1 \Rightarrow t_1 | \cdots | k \Rightarrow t_k].$$
- Secondly for $r_1 < \cdots < r_k = l$, we write $\mathsf{case}\ s[r_1 \Rightarrow t_1 | \cdots | r_k \Rightarrow t_k]$ to mean $\mathsf{case}\ s[u_0 | \cdots | u_l]$, where for each $0 \leqslant i \leqslant l$
$$u_i \quad \text{is} \quad \begin{cases} t_j & \text{if } i = r_j, \text{ for some } j, \\ \Omega & \text{otherwise.} \end{cases}$$

The operational semantics of the language **P** is obtained from that of PCF by adding the following rules:

$$\frac{s \Downarrow j \qquad t_j \Downarrow v}{\mathsf{case}\ s[t_0 | \cdots | t_k] \Downarrow v} \quad 0 \leqslant j \leqslant k.$$

The case construct

$$x : \iota, y_0 : \iota, \cdots, y_n : \iota \quad \vdash \quad \mathsf{case}\ x[y_0 | \cdots | y_n] : \iota$$

has an obvious interpretation as an innocent strategy of the arena $(\underbrace{\iota, \cdots, \iota}_{n+3})$ represented by the following innocent function:

$$\begin{cases} [& \mapsto & (_1 \\ [\cdot (_1 \cdot)_i & \mapsto & (_{i+1} \\ [\cdot (_1 \cdot)_i \cdot (_{i+1} \cdot)_m & \mapsto &]_m \end{cases}$$

We define the Ω-*match ordering* $\leqslant_\Omega$ over terms of **P** as follows: for any $n \geqslant 0$, and for any **P**-context[13] $C[X_1, \cdots, X_n]$, we have

$$s \equiv C[\Omega, \cdots, \Omega] \quad \leqslant_\Omega \quad t$$

[13] The *raw* **P**-contexts are defined as follows:

$$C \quad ::= \quad X \ | \ x \ | \ c \ | \ \lambda x : A.C \ | \ C_1 \cdot C_2 \ | \ \mathbf{Y}(C) \ | \ \mathsf{case}\ C[C_0 | \cdots | C_l],$$

whenever $t \equiv C[u_1, \cdots, u_n]$ for some $\mathbf{P}$-terms $u_1, \cdots, u_n$. It is easy to see that $\leqslant_\Omega$ is a partial order over terms of the language $\mathbf{P}$.

Finite canonical form For any PCF-types $A_1, \cdots, A_n$ where $n \geqslant 0$, we define the collection

$$\mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$$

of (program-type) *finite canonical forms* (FCFs) of $\mathbf{P}$ with free variables appearing in the list $f_1, \cdots, f_n$ as follows:

- The program-type Ω and $n \geqslant 0$ are in $\mathsf{FCF}[\vec{f} : \vec{A}]$.
- For any $\vec{f} : \vec{A} \equiv f_1 : A_1, \cdots, f_n : A_n$ and for any $1 \leqslant i \leqslant n$ where

$$\begin{array}{rcll} A_i & \equiv & (C_1, \cdots, C_m, \iota) & \text{and where} \\ C_j & \equiv & (D_{j1}, \cdots, D_{jp_j}, \iota) & \text{for each } 1 \leqslant j \leqslant m; \end{array}$$

  if $r_c \in \mathsf{FCF}[\vec{f} : \vec{A}]$ for each $0 \leqslant c \leqslant k$ and if $t_j \in \mathsf{FCF}[\vec{f} : \vec{A}, \vec{y_j} : \vec{D_j}]$ for each $1 \leqslant j \leqslant m$, then

$$\mathsf{case}\ f_i(\lambda \vec{y_1} : \vec{D_1}.t_1) \cdots (\lambda \vec{y_m} : \vec{D_m}.t_m)[r_0| \cdots |r_k] \quad \in \quad \mathsf{FCF}[\vec{f} : \vec{A}].$$

Note that a FCF is by definition of program type; and it is either Ω, or a number n, or a definition-by-cases construct.

A map from finite canonical forms to compact innocent strategies For PCF-types $A = (A_1, \cdots, A_n, \iota)$ and for any $s \in \mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$, we define by recursion a partial function

$$\theta[\lambda \vec{f} : \vec{A}.s] \quad : \quad \{\text{P-views of } A\} \quad \longrightarrow \quad \{\text{P-moves of } A\}$$

and prove simultaneously that $\theta[\lambda \vec{f}.s]$ is a compact innocent function. There are three cases.

Case 1: s is Ω. The function $\theta[\lambda \vec{f}.\Omega]$ is the everywhere undefined partial function.

Case 2: s is a number n. The function $\theta[\lambda \vec{f}.n]$ is the least partial function that maps the initial question "$[^A$" of A to the answer n.

Case 3: s is the case-construct $\mathsf{case}\ f_i(\lambda \vec{y_1}.t_1) \cdots (\lambda \vec{y_m}.t_m)[r_0| \cdots |r_k]$ where i is a number between 1 and n where

$$\begin{array}{rcl} A_i & \equiv & (C_1, \cdots, C_m, \iota) \\ C_j & \equiv & (D_{j1}, \cdots, D_{jp_j}, \iota) \end{array}$$

with $r_c \in \mathsf{FCF}[\vec{f} : \vec{A}]$ for each $0 \leqslant c \leqslant k$; and $t_j \in \mathsf{FCF}[\vec{f} : \vec{A}, \vec{y_j} : \vec{D_j}]$ for each $1 \leqslant j \leqslant m$. Write $B_j \equiv (\vec{A}, \vec{D_j}, \iota)$. By the recursion hypothesis $\theta[\lambda \vec{f}.r_c]$ and $\theta[\lambda \vec{f}\vec{y_j}.t_j]$ are compact innocent functions. We then define $\theta[\lambda \vec{f}.s]$ as the least partial function satisfying the following:

- $\theta[\lambda \vec{f}.s]$ maps "$[^A$" to the initial question "$(^{A_i}$" of A_i in A
- for each $1 \leqslant j \leqslant m$, if $\theta[\lambda \vec{f}\vec{y_j}.t_j]$ maps $[^{B_j} \cdot p$ to m then $\theta[\lambda \vec{f}.s]$ maps $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p$ to m
- for each $0 \leqslant c \leqslant k$, if $\theta[\lambda \vec{f}.r_c]$ maps $[^A \cdot p$ to m then $\theta[\lambda \vec{f}.s]$ maps $[^A \cdot (^{A_i} \cdot)^{A_i}_c \cdot p$ to m;

where X ranges over meta-variables for "holes" and c over constants of PCF. A $\mathbf{P}$-context is a raw context which is type-correct.

where we write "$(^{A_i}$" and "$[^{C_j}$" to denote the initial questions of A_i and C_j respectively as they occur in A, and where "$)^{A_i}_c$" is the answer c associated with the question "$(^{A_i}$". For the definition to be sound, we need the following lemma whose proof is straightforward and we omit it.

Lemma 7.3.1 (i) If $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p$ is a P-view of A then $[^{B_j} \cdot p$ is a P-view of B_j. Further if the legal position $[^{B_j} \cdot u$ of B_j, in which the question "$[^{B_j}$" is not explicitly answered, is in $\sum_{[^{B_j} \cdot p} \theta[\lambda \vec{f} \vec{y_j}.t_j]$ then $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot u$ is a legal position of A, and is in $\sum_{[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p} \theta[\lambda \vec{f}.s]$.

(ii) If $[^A \cdot (^{A_i} \cdot)^{A_i}_c \cdot p$ is a P-view in A, then $[^A \cdot p$ is a P-view of A. □

It remains to show that $\theta[\lambda \vec{f}.s]$ thus defined is an innocent function; its finiteness is obvious. Suppose $\theta[\lambda \vec{f}.s]$ maps the P-view $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p$ (say) of A to a P-move m. By definition of $\theta[\lambda \vec{f}.s]$, $\theta[\lambda \vec{f} \vec{y_j}.t_j]$ maps the P-view $[^{B_j} \cdot p$ of B_j to m. By the recursion hypothesis ($\theta[\lambda \vec{f} \vec{y_j}.t_j]$ is innocent), $\sum_{[^{B_j} \cdot p} \theta[\lambda \vec{f} \vec{y_j}.t_j]$ is non-empty; and so, by (i) of the preceding lemma, so is $\sum_{[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p} \theta[\lambda \vec{f}.s]$. Take any legal position $[^A \cdot (^{A_i} \cdot u \in \sum_{[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p} \theta[\lambda \vec{f}.s]$. Since the innocent function $\theta[\lambda \vec{f} \vec{y_j}.t_j]$ maps $[^{B_j} \cdot p$ to m, we know that m is explicitly justified by some unanswered question which appears in the P-view $[^{B_j} \cdot p$. By the preceding Lemma, m is explicitly justified by some question which appears in $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p$ which by assumption is the P-view of $[^A \cdot (^{A_i} \cdot u$. Hence we infer that $[^A \cdot (^{A_i} \cdot u \cdot m$ is a legal position of A, and we are done.

For FCFs s and s' in $\mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$ suppose $\theta[\lambda \vec{f}.s] \subseteq \theta[\lambda \vec{f}.s']$. W.l.o.g. we may assume that s is a definition-by-cases construct defined as in the preceding. Since $\theta[\lambda \vec{f}.s]$ maps "$[^A$" to "$(^{A_i}$" and that $\theta[\lambda \vec{f}.s] \subseteq \theta[\lambda \vec{f}.s']$, by definition of $\theta[\lambda \vec{f}.s']$, we infer that s' is a definition-by-cases construct of the shape

$$\lambda \vec{f}.\mathsf{case}\ f_i(\lambda \vec{y_1}.t'_1) \cdots (\lambda \vec{y_m}.t'_m)[r'_0 | \cdots | r'_k].$$

We claim that $\theta[\lambda \vec{f} \vec{y_j}.t_j] \subseteq \theta[\lambda \vec{f} \vec{y_j}.t'_j]$. To see this, suppose $\theta[\lambda \vec{f} \vec{y_j}.t_j]$ maps the P-view $[^{B_j} \cdot p$ to some move m. By definition of $\theta[\lambda \vec{f} \vec{y_j}.t_j]$, this must mean that $\theta[\lambda \vec{f}.s]$ maps $[^A \cdot (^{A_i} \cdot [^{C_j} \cdot p$ to m. By supposition $\theta[\lambda \vec{f}.s] \subseteq \theta[\lambda \vec{f}.s']$, and by the definition of $\theta[\lambda \vec{f}.s']$, we infer that $\theta[\lambda \vec{f} \vec{y_j}.t'_j]$ also maps $[^{B_j} \cdot p$ to m. Hence, by the recursion hypothesis, we have $t_j \leqslant_\Omega t'_j$. Essentially the same reasoning justifies $r_c \leqslant_\Omega r'_c$. Hence we have $s \leqslant_\Omega s'$.

To summarize we have proved:

Proposition 7.3.2 *For any* FCF $s, s' \in \mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$,

(i) the partial function $\theta[\lambda \vec{f}.s]$ is a compact innocent function of the arena $A \equiv (A_1, \cdots, A_n, \iota)$.

(ii) $s \leqslant_\Omega s'$ if and only if $\theta[\lambda \vec{f}.s] \subseteq \theta[\lambda \vec{f}.s']$. □

A map from compact innocent strategies to finite canonical forms We show that all compact innocent strategies of PCF-types are definable in PCF by induction on the size of the defining innocent function. For any $k' \geqslant 0$, we identify the term $\mathsf{case}\ s[t_0 | \cdots | t_k]$ with

$$\mathsf{case}\ s[t_0 | \cdots | t_k | \underbrace{\Omega | \cdots | \Omega}_{k'}].$$

Proposition 7.3.3 *For any* PCF*-type $A = (A_1, \cdots, A_n, \iota)$ and any compact innocent strategies σ and σ' of A,*

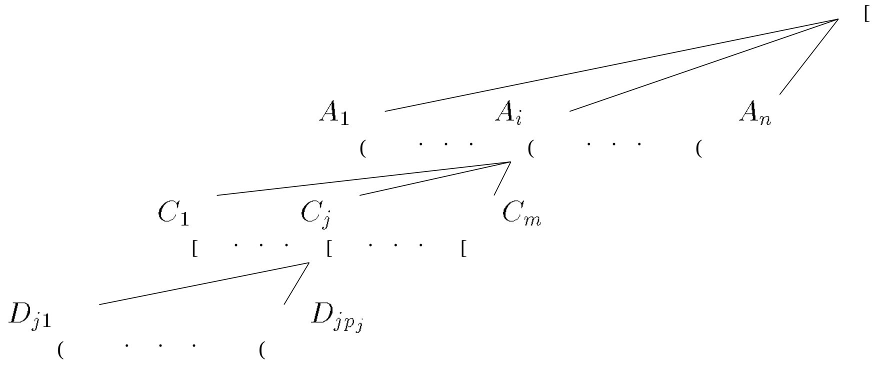

Figure 11: Shape of arena A.

(i) there is a FCF $s_\sigma \in \mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$ *of* $\mathbf{P}$ *such that* $\lambda\vec{f} : \vec{A}.s_\sigma$ *defines* σ*.*

(ii) $\sigma \subseteq \sigma'$ *if and only if* $s_\sigma \leqslant_\Omega s_{\sigma'}$*.*

Proof For $\sigma = \sigma_f$ ranging over compact innocent strategies, we define the FCF

$$s_\sigma \quad \in \quad \mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n]$$

associated with σ by recursion on the size of the domain of f. Suppose $\sigma = \sigma_f$ is a compact innocent strategy of PCF-type $A = (A_1, A_2, \cdots, A_n, \iota)$. The trivial cases are:

- the strategy σ does nothing, in which case, the term $\lambda\vec{f}.\Omega$ defines σ;
- the strategy σ immediately outputs a number l, in which case, the term $\lambda\vec{f}.l$ defines σ.

There are two cases, depending on whether A_1 is the program type. We consider the non-trivial inductive case. Suppose in response to Opponent's first move (which must be the initial A-move "$[^A$"), σ asks an A_1-question, "$(^{A_1}$" say.

$$A_1 \quad \equiv \quad (C_1, C_2, \cdots, C_m, \iota).$$

We imagine that Opponent chooses initially to compute in C_j by posing a C_j-question "$[^{C_j}$", so the P-view at this point is $[^A \cdot (^{A_1} \cdot [^{C_j}$. Of course, Opponent may subsequently switch from C_j and bring in other $C_{j'}$; but the P-strategy in question is determined independently of that for whenever Opponent raises the initial question in $C_{j'}$, Player's view immediately collapses to $[^A \cdot (^{A_1} \cdot [^{C_{j'}}$.

Suppose that $C_j \equiv (D_{j1}, D_{j2}, \cdots, D_{jp_j}, \iota)$ and consider the ensuing moves and the effect on Player's view. The P-question moves are of two kinds, see Figure 11:

- $(^{A_1}, \cdots, (^{A_n}$ and questions which are hereditarily justified by them;
- $(^{D_{j1}}, \cdots, (^{D_{jp_j}}$ and questions which are hereditarily justified by them.

So every P-view until "$[^{C_j}$" is answered can be regarded as a P-view in the arena corresponding to:

$$(D_{j1}, \cdots, D_{jp_j}, A_1, \cdots, A_n, \iota).$$

Thus we derive from our strategy σ a strategy σ_j in such an arena. Note that f_{σ_j} is smaller than f_σ — f_{σ_j} is defined on the P-view $[^A \cdot p$ if and only if f_σ is defined on $[^A \cdot (^{A_1} \cdot (^{C_j} \cdot p$. Hence, by the induction hypothesis, we have a term

$$\lambda y_{j1} : D_{j1}. \cdots .\lambda y_{jp_j} : D_{jp_j}.\lambda f_1 : A_1 \cdots .\lambda f_n : A_n.t_j(\vec{y_j}, \vec{f})$$

whose interpretation is σ_j. In the case where C_j is the program type ι, then the corresponding term is $\lambda f_1 : A_1 \cdots f_n : A_n . t_j(\vec{f})$.

Note that the σ_j's completely determine the action of σ up to the moment that Opponent answers the question "$(^{A_1}$". Now consider the position once "$(^{A_1}$" is answered by an O-answer ")": the P-view is $[^{A} \cdot (^{A_1} \cdot)$. The O-answer ranges over a finite number of possible natural numbers, say $c_1, \cdots, c_k$ in increasing order. When we continue (Opponent will have "forgotten all that has happened"), Player's view will thereafter always start with $[^{A} \cdot (^{A_1} \cdot)$; so for each value $c_1, \cdots, c_k$, we get smaller strategies $\rho_1, \cdots, \rho_k$ telling us how σ continues. We get by the induction hypothesis the corresponding terms:

$$\begin{gathered} \lambda f_1 : A_1 . \cdots . \lambda f_n : A_n . u_1(\vec{f}), \\ \vdots \\ \lambda f_1 : A_1 . \cdots . \lambda f_n : A_n . u_k(\vec{f}); \end{gathered}$$

whose interpretations are $\rho_1, \cdots, \rho_k$ respectively.

We write $\vec{y_j} \equiv y_{j1}, \cdots, y_{jp_j}$. Now consider the term:

$$\lambda \vec{f} : \vec{A} . \mathsf{case}\ f_1(\lambda \vec{y_1} . t_1) \cdots (\lambda \vec{y_m} . t_m)[c_1 \Rightarrow u_1(\vec{f}) | \cdots | c_k \Rightarrow u_k(\vec{f})].$$

It is easy to see that σ is the interpretation of the above term.

The second part of the Theorem is proved by induction on the size of the strategies. We leave the essentially straightforward details to the reader. □

Putting the two preceding results together we can say the following.

Theorem 7.3.4 (Strong definability) *There are maps in opposite directions: for any* PCF*-type* $A = (A_1, \cdots, A_n, \iota)$

$$\begin{array}{ccc} \mathsf{FCF}[f_1 : A_1, \cdots, f_n : A_n] & \rightleftarrows & \{\, \textit{compact innocent strategies of } A \,\} \\ & & \\ f_1 : A_1, \cdots, f_n : A_n \vdash s & \longmapsto & \theta[\lambda \vec{f} . s] \\ & & \\ f_1 : A_1, \cdots, f_n : A_n \vdash s_\sigma & \longleftarrow & \sigma \end{array}$$

(the choice of variables $\vec{f} : \vec{A}$ in the above is of course immaterial). The pair of maps defines a bijection and hence (by the two preceding propositions) an isomorphism between finite canonical forms and compact strategies. It is straightforward to see that for any $s \in \mathsf{FCF}[\vec{f} : \vec{A}]$, the compact innocent strategy associated with s coincides with its denotation in $\mathbb{CA}$; that is to say

$$\theta[\lambda \vec{f} . s] \quad = \quad [\![\lambda \vec{f} . s]\!]^{\mathbb{CA}}.$$

□

Remark 7.3.5 (i) In Proposition 7.3.4 we only consider the case of PCF generated from one program type ι. Nevertheless it is entirely straightforward to extend the same argument therein to deal with PCF proper *i.e.* where both ι and o are program types. The boolean conditionals would then play exactly the same role as that of definition-by-cases constructs.

(ii) The strong definability result can be straightforwardly extended to innocent strategies in general. Of course the correspondence would then be with possibly infinitary canonical forms.

We can take advantage of the Strong Definability Theorem as a representation device to explain the structure of the dialogue game model. For example it is easy to see that the arena $o \Rightarrow o$ is infinite: the following represents a family of distinct strategies:

$$\lambda x : o\, \underbrace{\mathsf{cond}\ x\ (\mathsf{cond}\ x\ (\cdots(\mathsf{cond}}_{n}\ x\ \mathsf{t}\ \Omega)\cdots)\Omega)\ \Omega$$

corresponding to the prime innocent strategy generated by

$$[\cdot\, \underbrace{(_1\cdot)_{\mathsf{t}}\cdots(_1\cdot)_{\mathsf{t}}}_{n} \quad \mapsto \quad]_{\mathsf{t}}$$

7.4 Strong adequacy and order full abstraction

Building on the definability result we can now prove that dialogue games and innocent strategies give an order-extensional, order (or inequationally) fully abstract model for PCF.

Proposition 7.4.1 (Strong adequacy) *For any $\mathbf{P}$-program s, and for any value v, $s \Downarrow v$ if and only if $[\![s]\!] \Downarrow v$ (in the category $\mathbb{CA}$).*

Proof Our proof is similar to Plotkin's proof of adequacy of the Scott function space model for PCF. Plotkin used a reducibility-style argument pioneered by Tait [Tai67] and Girard [Gir72]. Since the argument is standard and well-documented (see Plotkin's proof in [Plo77]; see also [Gun92] for an exposition), we omit it here. $\Box$

Proposition 7.4.2 *For any PCF-terms s and t of the same type, $s \lessapprox t$ in PCF if and only if $s \lessapprox t$ in the extended language $\mathbf{P}$.*

To prove the proposition, consider a translation of terms from $\mathbf{P}$ to PCF $s \mapsto \overline{s}$ defined by recursion as follows:

$$\begin{array}{rcl}
\overline{st} & \stackrel{\text{def}}{=} & \overline{s}\overline{t} \\
\overline{\lambda x : A.s} & \stackrel{\text{def}}{=} & \lambda x : A.\overline{s} \\
\overline{\mathbf{Y}(s)} & \stackrel{\text{def}}{=} & \mathbf{Y}(\overline{s}) \\
\overline{\mathsf{case}\ s[t_0|\cdots|t_k]} & \stackrel{\text{def}}{=} & \mathsf{cond}(\mathsf{eq}\overline{s}0)\overline{t_0}(\mathsf{cond}(\mathsf{eq}\overline{s}1)\overline{t_1}\cdots(\mathsf{cond}(\mathsf{eq}\overline{s}k)\overline{t_k}\Omega)\cdots);
\end{array}$$

where eq is a PCF-term of type (ι, ι, o) satisfying

- $\mathsf{eq}uv\Downarrow$ if and only if both $u\Downarrow$ and $v\Downarrow$, and further,
- $\mathsf{eq}uv \Downarrow \mathsf{t}$ if and only if $u \Downarrow n$ and $v \Downarrow n$, for some natural number n.

In addition the translation preserves variables, constants and all PCF-terms. Note that for any $\mathbf{P}$-terms s and t, $\overline{s[t/x]} \equiv \overline{s}[\overline{t}/x]$.

Lemma 7.4.3 For any closed term s of the language $\mathbf{P}$, and for any value v (which may be an abstraction), $s \Downarrow v$ (in $\mathbf{P}$) if and only if $\overline{s} \Downarrow \overline{v}$ (in PCF).

Proof We sketch the proof of the direction "$\Leftarrow$" as an illustration; the other direction may be proved in a similar way. The proof is by induction over the rules that define the relation $s \Downarrow v$. The base cases are trivial. Consider the case of the following rule:

$$\frac{s \Downarrow j \qquad t_j \Downarrow v}{\mathsf{case}\ s[t_0|\cdots|t_k] \Downarrow v} \quad 0 \leqslant j \leqslant k$$

Suppose $\overline{\mathsf{case}\ s[t_0|\cdots|t_k]} \Downarrow v$ in PCF. Since $\overline{\mathsf{case}\ s[t_0|\cdots|t_k]}$ is

$$\mathsf{cond}(\mathsf{eq}\overline{s}0)\overline{t_0}(\mathsf{cond}(\mathsf{eq}\overline{s}1)\overline{t_1}\cdots(\mathsf{cond}(\mathsf{eq}\overline{s}k)\overline{t_k}\Omega)\cdots),$$

this can only be so provided $\mathsf{eq}\overline{s}i \Downarrow \mathsf{t}$ (or equivalently $\overline{s} \Downarrow i$) and $\overline{t_i} \Downarrow v$, for some i. By the induction hypothesis, $s \Downarrow i$ and $t_i \Downarrow v$; and so, by the rule in question, we have $\mathsf{case}\ s[t_0|\cdots|t_k] \Downarrow v$. $\square$

The direction "$\Leftarrow$" of Proposition 7.4.2 is immediate since a program context of PCF is also a program context of $\mathbf{P}$. To prove the other direction, take a program context $C[X]$ of $\mathbf{P}$ such that both $C[\lambda\vec{f}.s]$ and $C[t]$ are programs, where s and t are PCF-terms. Suppose $C[\lambda\vec{f}.s] \Downarrow v$. Since $\overline{C[\lambda\vec{f}.s]}$ is $\overline{C}[\lambda\vec{f}.s]$ and $\overline{v}$ is v, by Lemma 7.4.3, $\overline{C}[\lambda\vec{f}.s] \Downarrow v$ in PCF. Assuming $s \sqsubseteq t$, we have $\overline{C}[t] \Downarrow v$. Hence, by Lemma 7.4.3 again, $C[t] \Downarrow v$ in $\mathbf{P}$. This concludes the proof of Proposition 7.4.2.

Take any (closed) PCF-terms s and t of the same type, A say. Write $[\![\, s \,]\!]$ for the denotation of s in the observational quotient $\hat{\mathbb{CA}}$ (defined with respect to the canonical notion of observable). By definition $[\![\, s \,]\!] \lesssim [\![\, t \,]\!]$ means that for any innocent strategy ρ of the arena $A \Rightarrow \iota$, for any number n,

$$[\![\, s \,]\!];\rho \Downarrow n \quad \Longrightarrow \quad [\![\, t \,]\!];\rho \Downarrow n.$$

Since the map $\rho \mapsto [\![\, s \,]\!];\rho$ from $\underline{A \Rightarrow \iota}$ (the dI-domain of all innocent strategies of $A \Rightarrow \iota$ ordered by inclusion) to $\underline{\iota}$ is a continuous function between CPOs, $[\![\, s \,]\!];\rho \Downarrow n$ if and only if $[\![\, s \,]\!];\nu \Downarrow n$ for some compact approximant ν of ρ. Hence it suffices to consider only compact innocent strategies ν of the arena $A \Rightarrow \iota$.

So suppose $s \sqsubseteq t$ in PCF, and suppose for some compact innocent strategy ν, $[\![\, s \,]\!];\nu \Downarrow n$. By the $\mathbf{P}$-definability of compact innocent strategies (Proposition 7.3.4), there is a FCF h corresponding to ν such that $[\![\, hs \,]\!] \Downarrow n$. Since $\mathbb{CA}$ is strongly adequate for $\mathbf{P}$ (Proposition 7.4.1), this is equivalent to $hs \Downarrow n$ in the extended language $\mathbf{P}$. Since $s \sqsubseteq t$ in PCF, by Proposition 7.4.2, $s \sqsubseteq t$ in the extended language $\mathbf{P}$. Hence $ht \Downarrow n$ which is equivalent to $[\![\, ht \,]\!] \Downarrow n$ in $\mathbb{CA}$, by the same adequacy result as before. To summarize we have proved:

Theorem 7.4.4 (Full abstraction) *The observational quotient $\hat{\mathbb{CA}}$ of the category $\mathbb{CA}$ gives rise to an order-extensional, order fully abstract model of* PCF. $\square$

7.5 Examples and counter-examples

A type-2 strategy Consider the type-2 PCF-term (see e.g. [BCL86, p. 129] or [Cur93])

$$F \quad = \quad \lambda f : (o,o,o).f(f\mathsf{t}\Omega)(f\Omega\mathsf{t}) \quad : \quad ((o,o,o),o)$$

For ease of explanation we label the questions of the arena $((o,o,o),o)$ as in Figure 12. We describe the innocent strategy denoted by F informally in terms (see Figure 13) of its interaction with the innocent strategy "left or" $\mathsf{l\text{-}or}$ which corresponds exactly (in the sense of Theorem 7.3.4) to the PCF-term

$$\mathsf{l\text{-}or} \quad = \quad \lambda x : o.\lambda y : o.\mathsf{cond}x\mathsf{t}(\mathsf{cond}y\mathsf{t}\mathsf{f}) \quad : \quad (o,o,o).$$

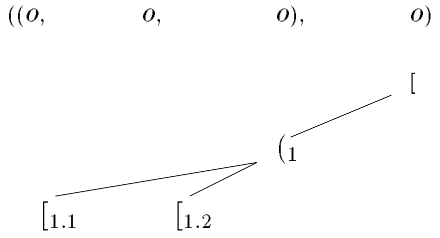

Figure 12: Tree of questions of arena $((o, o, o), o)$.

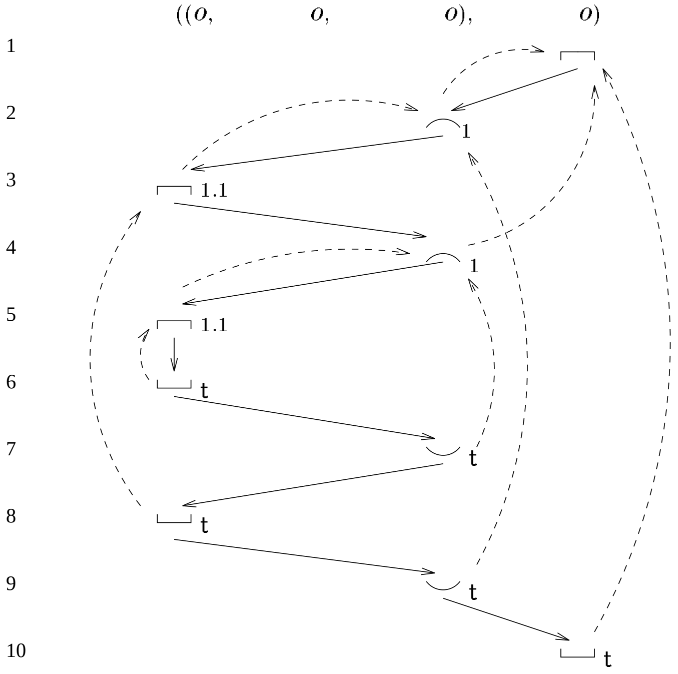

Figure 13: Trace of $F \cdot \mathsf{l\text{-}or}$.

The legal position in Figure 13 is precisely the trace of the computation $F \cdot \mathsf{l\text{-}or}$. Formally it is the uncovering of the maximal legal position “$[^{o}\cdot]_{\mathsf{t}}$” in accord with

$$\mathbf{1} \xrightarrow{\langle F, \mathsf{l\text{-}or} \rangle} ((o, o, o), o) \times (o, o, o) \xrightarrow{\mathrm{ev}} o.$$

The dotted arrows pointing “backwards” are the justification pointers. We number the moves from 1 to 10 for ease of identification. In response to the opening move “[”, Player makes the move “$(_1$” corresponding to the head variable f of F. Opponent, playing $\mathsf{l\text{-}or}$ and regarding “$(_1$” as its opening move, raises the question “$[_{1.1}$” corresponding to the head variable x (left argument) in $\mathsf{l\text{-}or}$. From this point onwards until the 3rd move is answered (in the 8th move), Player plays the substrategy $F_1 = \lambda f : (o, o, o).f\mathsf{t}\Omega$ corresponding to the subterm $f\mathsf{t}\Omega$ of F. The strategy F_1 regards the 3rd move “$[_{1.1}$” as its opening move, and responds by raising the question “$(_1$” (4th move) corresponding to the head variable f in F_1. Opponent regards the 4th move as an opening move (distinct from the 2nd move) of a new play. He responds as before by raising the question “$[_{1.1}$” (5th move) corresponding to the left argument in accord with $\mathsf{l\text{-}or}$. This corresponds to querying the first of the two arguments of the head variable in $F_1 = \lambda f.f\mathsf{t}\Omega$, so Player supplies the answer “$]_{\mathsf{t}}$” (6th move). The strategy $\mathsf{l\text{-}or}$ now has enough information to supply the answer “$)_{\mathsf{t}}$” to Player's earlier question (4-th move). In response Player concludes the substrategy F_1 by supplying the answer “$]_{\mathsf{t}}$” to Opponent's question

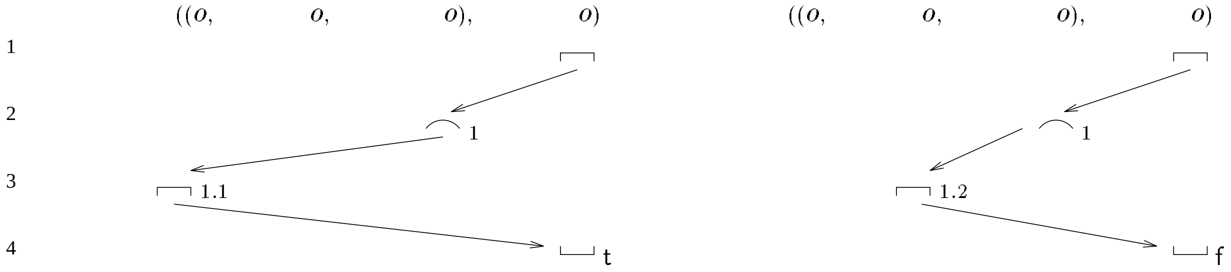

Figure 14: G_1, a (non) strategy that tells left-or and right-or apart.

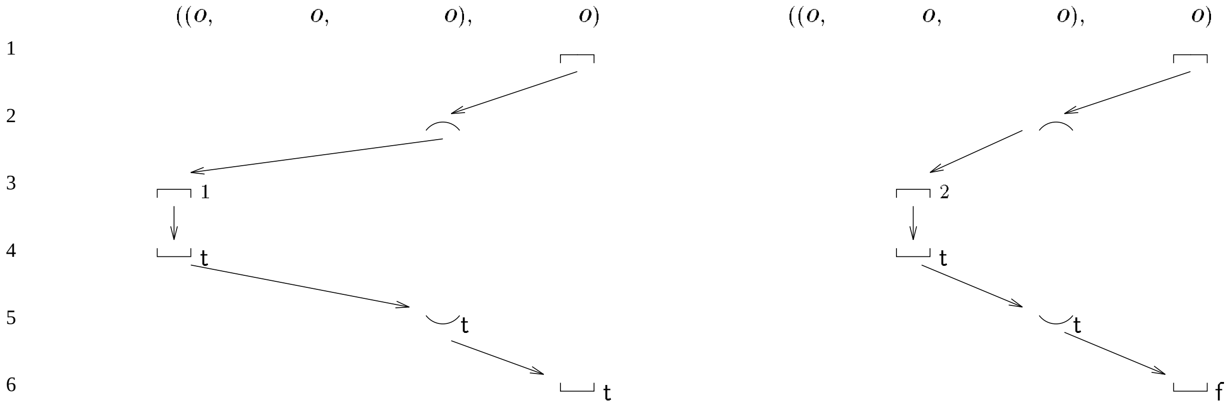

Figure 15: G_2, another (non-innocent) strategy that tells left-or and right-or apart.

in the 3rd move. Opponent can now respond to Player's question in the 2nd move by returning the answer "$)_{\mathsf{t}}$"; whereupon Player concludes the play by "$]_{\mathsf{t}}$", echoing the preceding move.

Remark 7.5.1 There can be no innocent strategy of the arena $((o, o, o), o)$ which tells $\mathsf{l\text{-}or}$ and $\mathsf{r\text{-}or}$ apart, say, by mapping the former to t and the latter to f. This is just as well in view of Curien's observation in [Cur93, p. 358]:

> The type-2 function of type $((o, o, o), o)$ which sends the left-or to t and the right-or to f is not PCF-definable.

Curien's observation highlights an important feature of PCF-style higher-type sequential composition: higher-order functionals interact extensionally with their functional arguments.

One way to see why the preceding type-2 function is not definable as an innocent strategy is to appeal to the correspondence between compact innocent strategies and finite canonical forms (FCF) of the language **P**, and then argue syntactically following Curien. It is instructive however to sketch an explanation from first principles in terms of the definition of innocent strategy. Consider the strategies G_1 and G_2 defined informally in Figures 14 and 15 respectively. The question-moves therein (all from arena $((o, o, o), o)$) are annotated with occurrences as in Figure 12. It is easy to see that both G_1 and G_2 take $\mathsf{l\text{-}or}$ to t and $\mathsf{r\text{-}or}$ to f. Fortunately neither is an innocent strategy.

- G_1's response in the 4th move (see Figure 14) violates the last-asked-first-answered condition so that the two sequences of moves are not even legal positions. This is essentially the catch (and throw) facility which has been studied by Cartwright, Curien and Felleisen in [CCF94].

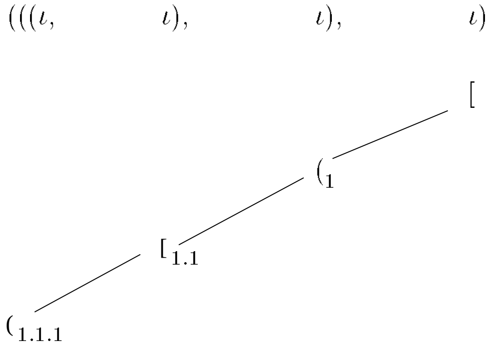

Figure 16: Tree of questions of arena $(((\iota,\iota),\iota),\iota)$.

- In the case of G_2, both sequences of moves in Figure 15 are legal positions, and we definitely have a strategy. However G_2 is not innocent because its response at the 6th move is different in the two cases despite the fact that both legal positions have the same P-view (which is "$[\cdot(_1\cdot)_{\mathsf{t}}$") when truncated at the 5th move.

The (counter-) example G_2 illustrates well the rationale for studying strategies which are invariant over P-views. This is the essence of innocence. $\square$

A type-3 strategy As another example we consider (much more briefly) the interpretation of the following type-3 terms:

$$\begin{array}{rcl} F_1 & \stackrel{\text{def}}{=} & \lambda f:((\iota,\iota),\iota).f(\lambda y:\iota.f(\lambda x:\iota.y)) \quad : \quad (((\iota,\iota),\iota),\iota) \\ F_2 & \stackrel{\text{def}}{=} & \lambda f:((\iota,\iota),\iota).f(\lambda y:\iota.f(\lambda x:\iota.x)) \quad : \quad (((\iota,\iota),\iota),\iota). \end{array}$$

We illustrate the strategies defined by F_1 and F_2 respectively in terms of their interaction with the term

$$G \stackrel{\text{def}}{=} \lambda g:(\iota,\iota).g1 \quad : \quad ((\iota,\iota),\iota)$$

The questions of the arena $(((\iota,\iota),\iota),\iota)$ are annotated with occurrences as in Figure 16. We present the play corresponding to F_1G and F_2G in Figure 17. We have omitted the justification pointers of all moves except the 6-th. Note that F_1 and F_2 are only subtly different: in their respective interaction with G, the underlying sequences of moves (as a trace of the play) are identical, they only differ in the way the 6-th move is justified. This example was communicated to us by Gandy and Pani.

8 Universality

In this section we extend the full abstraction result (Theorem 7.4.4) by showing the following:

(i) every map between PCF-types in $\hat{\mathbb{CA}}$ is PCF-definable in some (partial) function parameter;

(ii) every map between PCF-types in $\hat{\mathbb{RA}}$ is PCF-definable.

Results of this kind are often called universality theorems.

To establish the results this we show that for every PCF-type A there is a PCF-term $\mathsf{u}_A : (\iota \Rightarrow \iota) \Rightarrow A$ (we shall suppress the dependence of u on A) with the following property.

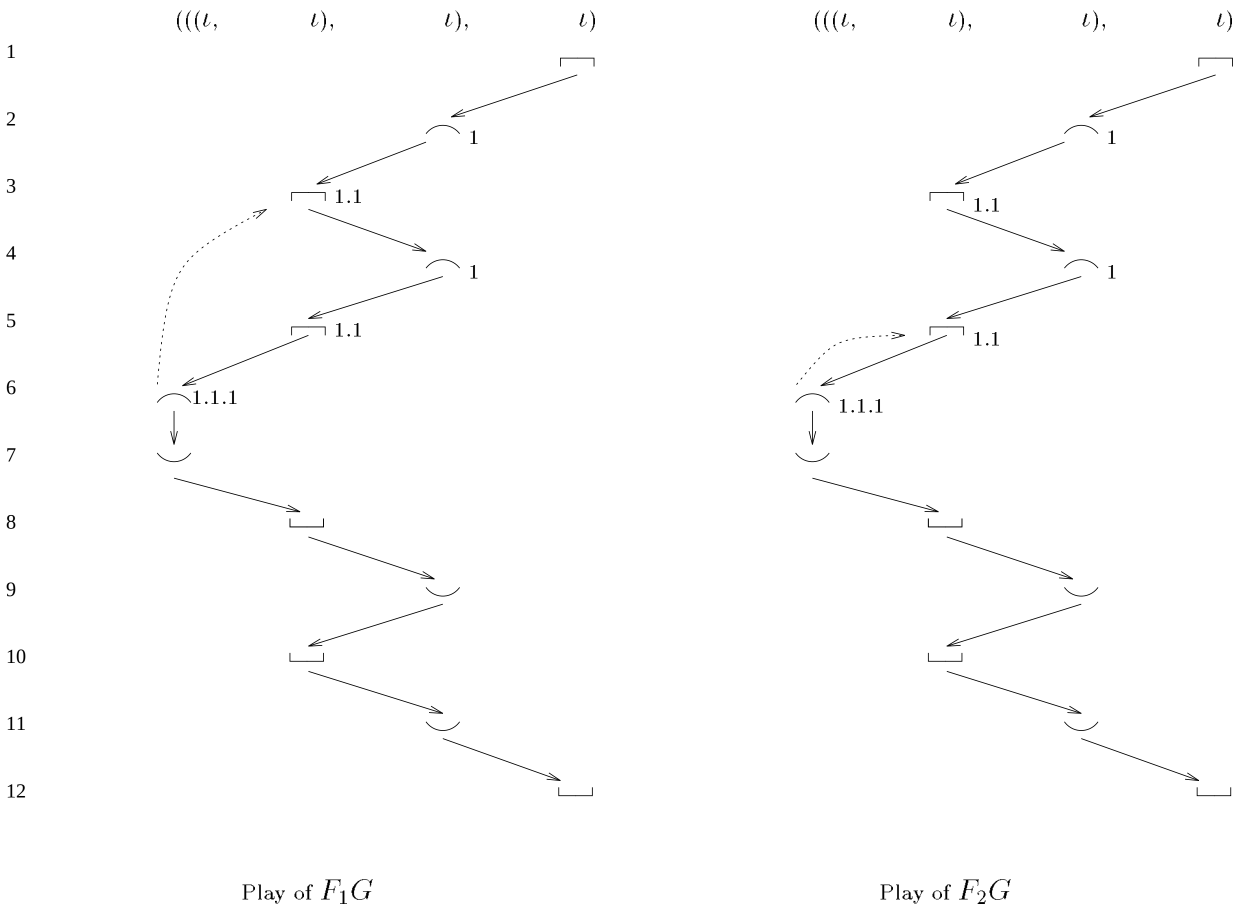

Figure 17: Two subtly different strategies.

> Suppose that σ is an innocent strategy of type A with coding function ϕ_σ as indicated in §5.6. Let $\overline{\phi_\sigma}$ be the canonical strategy in $\iota \Rightarrow \iota$ associated to $\phi_\sigma \in \mathbb{P}$. Then σ is observationally equivalent to $\overline{\phi_\sigma}; [\![\mathsf{u}]\!]$, where $[\![\mathsf{u}]\!]$ is the interpretation of u in $\mathbb{CA}$.

It should cause little confusion to drop the brackets and the bar; so henceforth we write the more natural equivalence as $\mathsf{u}(\phi_\sigma) \simeq \sigma$.

Of course the existence of a u as described immediately gives (i). (Note that definition in a partial function comes out of the proof but it is not difficult to replace it with a total function if preferred.) For (ii) we know that if σ is in $\mathbb{RA}$, then ϕ_σ is partial recursive. Now by Proposition 7.4.1 $\mathbb{CA}$ is computationally adequate. So by Proposition 2.4.16 ϕ_σ is representable in the strong sense in $\mathbb{CA}$ by the interpretation of a PCF-term s say. It follows (from the context lemma for $\mathbb{CA}$) that $\phi_\sigma \simeq s$. Hence $\mathsf{u}(s) \simeq \sigma$ and we have (ii).

8.1 Retracts of pure types

The construction of (generalized versions of) the term u just described involves a pretty standard "use of the recursion theorem" (here the fixed point operator). However it seems right to sketch a proof in some detail. We do this in the notationally simple case when A is a pure type (see below). There is no loss of generality as the full result follows in view of simple coding facts. (However some best possible estimates on the use of $\mathbf{Y}$ are lost in this approach.)

We start our simplifying scheme by introducing the notion of pure types which are conventionally denoted by natural numbers n: the pure type (denoted by) n is a particularly simple (PCF-) type of height n. For the rest of this section we shall assume that ι is the only ground type.

Definition 8.1.1 The *pure types* $0, 1, 2, \cdots$ are defined recursively by:

$$\begin{aligned} 0 &\stackrel{\text{def}}{=} \iota \\ n+1 &\stackrel{\text{def}}{=} (n \Rightarrow 0). \end{aligned}$$

Little confusion should arise from the (overloading) use of numbers both as elements of $\mathbb{N}$ (and indeed strategies in $\mathbb{CA}$) and as (names for) pure types. We shall need a simple lemma.

Lemma 8.1.2 *Any pure type n is a retract of the pure type $n+1$ by PCF-definable maps. Thus $n \lhd n+1$ in $\mathbb{T}$.*

Proof We have $0 \lhd 1$ via

$$\begin{aligned} n : 0 &\mapsto \lambda x : \iota . n : 1 \\ f : 1 &\mapsto f(0) : 0, \end{aligned}$$

and then proceed inductively. □

When studying total type theories over the natural numbers, recursion theorists standardly exploit the fact that any type A is recursively a retract of the pure type $\mathsf{ht}(A)$. However in the case of partial type structures the situation is more subtle.

We define by recursion the notion of the *rank* $\mathsf{rk}(A)$ of a PCF-type A.

$$\begin{aligned} \mathsf{rk}(\iota) &\stackrel{\text{def}}{=} 0 \\ \mathsf{rk}(A_0, \cdots, A_{k-1}, \iota) &\stackrel{\text{def}}{=} \begin{cases} \mathsf{rk}(A_0) + 1 & \text{if } k = 1 \\ \max(\mathsf{rk}(A_0), \cdots, \mathsf{rk}(A_{k-1})) + 2 & \text{if } k \geqslant 2. \end{cases} \end{aligned}$$

Proposition 8.1.3 *Any* PCF*-type of rank n is a retract by* PCF*-definable maps*[14] *of the pure type n. (So for any* PCF*-type A there is an n with $A \lhd n$ in $\mathbb{T}$.)*

Proof We start by establishing the following:

Lemma 8.1.4 *Any finite product of a pure type n is a retract of $n+1$.*

Proof By induction on n. In case $n = 0$ we have $0 \times \cdots \times 0 \lhd 1$ via

$$\begin{array}{rcl} (n_0, \cdots, n_{k-1}) & \mapsto & \lambda x : \iota.\mathsf{cond}(\mathsf{eq}\, x 0) n_0 (\mathsf{cond}(\mathsf{eq}\, x 1) n_1 \cdots) \\ f & \mapsto & (f(0), \cdots, f(k-1)). \end{array}$$

For the induction step, assume the result for r and let n be $r+1$. Then

$$\begin{array}{rcll} n \times \cdots \times n & = & (r \Rightarrow 0) \times \cdots \times (r \Rightarrow 0) & \\ & \cong & (r \Rightarrow 0 \times \cdots \times 0) & \text{CCC isomorphism} \\ & \lhd & (r \Rightarrow (0 \Rightarrow 0)) & \\ & \cong & (r \times 0 \Rightarrow 0) & \text{CCC isomorphism} \\ & \lhd & (r \times r) \Rightarrow 0 & \text{Lemma 8.1.2} \\ & \lhd & (r + 1 \Rightarrow 0) & \text{inductive hypothesis} \\ & = & n + 1. & \end{array}$$

□

Now we prove the proposition by induction on the structure of $A = (A_0, \cdots, A_{k-1}, \iota)$. Suppose $k = 1$ and $\mathsf{rk}(A) = r + 1$ then by the induction hypothesis

$$A = (A_0 \Rightarrow \iota) \quad \lhd \quad r \Rightarrow 0 \quad = \quad r + 1.$$

Suppose $k \geqslant 2$ and A has rank $r + 2$. Then by definition $\mathsf{rk}(A_i) \leqslant r$ for each i. Hence

$$\begin{array}{rcll} A_0 \times \cdots \times A_{k-1} \Rightarrow \iota & \lhd & r \times \cdots \times r \Rightarrow 0 & \text{by induction hypothesis} \\ & \lhd & (r+1) \Rightarrow 0 & \text{by preceding Lemma} \\ & = & r + 2. & \end{array}$$

□

8.2 Preamble to the main construction

Suppose that $\sigma : m_0 \times \cdots \times m_{k-1} \longrightarrow \iota$ in $\mathbb{CA}$ is an innocent strategy whose arguments are all pure types $m_i \leqslant n + 2$. Then we have the following three cases:

Undefined case. σ does not respond to the initial question "[".

Constant case. σ responds to the initial question "[" with an immediate answer "$]_c$" for some value c.

[14] Here we mean maps of the PCF-type theory $\mathbb{T}$ (see Remark 2.3.4).

Inductive case. σ responds to the initial question "[" with the initial question "(" in the i-th game m_i $(0 \leqslant i \leqslant k-1)$. Now the general circumstance is when $m_i = m+2$ is at least 2 and we analyze this first. The only interesting response for O is the initial question in $(m+1)$ which is now justified. The play until that question is answered is (effectively) a play in the game

$$m \times m_0 \times \cdots \times m_{k-1} \Rightarrow 0,$$

and we write σ' for the strategy in this game derived from σ. (See below for the simple relation between the representing innocent functions.) The special circumstances when $m_i = 1$ or $m_i = 0$ are simpler. In case $m_i = 1$, the only interesting response for O is the initial question in 0 which is now justified. The play until that question is answered is effectively a play in the game $m_0 \times \cdots \times m_{k-1} \Rightarrow 0$ and we write σ' for the strategy in this game derived from σ. Finally in case $m_i = 0$, O can only give an uninteresting immediate answer c to P's question. In all three circumstances, the analysis now continues in the same fashion. At some stage (possibly at once — the "uninteresting" response, possibly after many completed plays against σ') O may reply to P's initial question in m_i. If the answer is the value c, then the P-view will be "$[\cdot(\cdot)_c$" and P is then essentially back in the position of playing in a game $m_0 \times \cdots \times m_{k-1} \Rightarrow 0$ again. We write σ_c for the strategy obtained from σ for the succeeding play. (Again see below for the relation between the representing innocent functions.)

We consider the output behaviour of σ in these three cases. Let $\tau_0 : m_0, \cdots, \tau_{k-1} : m_{k-1}$ be k innocent strategies of pure type. Then the composite

$$(\tau_0, \cdots, \tau_{k-1}); \sigma \quad = \quad \sigma(\tau_0, \cdots, \tau_{k-1}) \quad : \quad 0$$

is a strategy in ι. So it is either the unresponsive strategy or else directly responds with answer a natural number.

In case 1, $\sigma(\tau_0, \cdots, \tau_{k-1})$ is the unresponsive strategy.

In case 2, $\sigma(\tau_0, \cdots, \tau_{k-1})$ responds with some natural number c.

In case 3, $\sigma(\tau_0, \cdots, \tau_{k-1})$ is the unresponsive strategy unless

(a) the interpretation of $\tau_i(\lambda a.\sigma'(a, \tau_0, \cdots, \tau_{k-1}))$ (in the general circumstances), or $\tau_i(\sigma'(\tau_0, \cdots, \tau_{k-1}))$ in case $m_i = 1$, or τ_i in case $m_i = 0$, is some natural number c, and

(b) $\sigma_c(\tau_0, \cdots, \tau_{k-1})$ responds with some natural number d, in which case $\sigma(\tau_0, \cdots, \tau_{k-1})$ responds with d.

Clearly the strategy σ' and sequence of strategies σ_c depend in a simple way on σ: omitting justification indices and coding details the representing innocent functions satisfy:

$$f_{\sigma'}(u) = a \iff f_\sigma(\text{"}[\cdot(\cdot u\text{"}) = a$$
$$f_{\sigma_c}(\text{"}[\cdot v''\text{)} = a \iff f_\sigma(\text{"}[\cdot(\cdot)_c \cdot v\text{"}) = a.$$

This dependence is reflected in terms of the codes ϕ_σ for σ introduced in §5.6: there are (least) recursive operators $\Phi' : \mathbb{P} \longrightarrow \mathbb{P}$ and $\Phi : \mathbb{N} \times \mathbb{P} \longrightarrow \mathbb{P}$ such that for all $\sigma : m_0 \times \cdots \times m_k \longrightarrow 0$ in $\mathbb{CA}$,

$$\Phi'(\phi_\sigma) = \phi_{\sigma'} \qquad \text{and} \qquad \Phi(c, \phi_\sigma) = \phi_{\sigma_c}.$$

We suppress the dependence of Φ and Φ' on the sequence $(m_0, \cdots, m_{k-1})$, and shall write $\Phi(c, \phi)$ in its curried form $\Phi_c(\phi)$. We need an explicit choice of Φ' and Φ. The natural choice to make is of the least such recursive operators, so we set

$$\Phi'(\phi) \quad = \quad \bigcup \{ \phi_{\tau'} : \tau \text{ is a (finite) innocent strategy with } \phi_\tau \subseteq \phi \}$$

and

$$\Phi(\phi, c) \quad = \quad \bigcup \{ \phi_{\tau_c} : \tau \text{ is a (finite) innocent strategy with } \phi_\tau \subseteq \phi \}.$$

8.3 Representability of recursive operators

It does not seem profitable to extend ideas of numeralwise representability to higher types; but in the §8.2 we decomposed codes ϕ_σ using certain recursive operators so one minor extension proves useful in the proof of universality.

We write $\mathbb{P}$ for the set of all partial functions from $\mathbb{N}$ to $\mathbb{N}$. We shall need to use the fact that certain (simple) recursive operators $\Phi : \mathbb{P} \longrightarrow \mathbb{P}$ can be represented in a suitable sense in PCF. More generally we should consider $\Phi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{N}$ and $\Phi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{P}$. The idea is to use the type $\iota \Rightarrow \iota$ as a substitute for $\mathbb{P}$, since every map $\iota \longrightarrow \iota$ in a model will numeralwise represent (in the strict sense) a unique partial function, that is, elements of $\mathbb{P}$. However we face a number of problems.

1. Not every partial function need be numeralwise representable in the model. For example in the initial model $\mathbb{T}$ the representable functions are exactly the partial recursive functions.
2. The same partial function may be numeralwise represented by distinct maps in the model. For example the constant term $\lambda x.0$ and the term defined implicitly by the recursive equation

   $$f(x) \quad \stackrel{\text{def}}{=} \quad \mathsf{if}\ x = 0\ \mathsf{then}\ 0\ \mathsf{else}\ f(\mathsf{pred0})$$

   will usually denote different maps but both will represent the constant function with value 0.
3. We can no longer dodge issues of sequentiality. Some recursive operators $\Phi : \mathbb{P} \longrightarrow \mathbb{N}$ can in no sense be represented in PCF: consider for example

   $$\Phi(\phi) = 0 \quad \iff \quad \text{either } \phi(0) = 0 \text{ or } \phi(1) = 0.$$

As regards point 3. we shall simply have to be careful to check PCF representability: in the cases where we need it, it is quite trivial. (There are in effect a number of exact characterizations of PCF – definability at this level in the literature.) As regards the first two points it seems best to cope with them as follows. First recall from Proposition 2.4.16 that for any finite partial numerical function we can find a term of PCF (in the sense of Remark 2.3.4) which numeralwise represents the function in $\mathbb{T}$. Such terms weakly represent their functions in any model but represent (without qualification) them in any adequate model. Secondly we can restrict attention to (Scott) continuous functions: any recursive operator is continuous and so determined by its values on finite functions; and a PCF-representable operator must be of this form. This motivates the following diefinition.

Definition 8.3.1 Suppose that $\mathbb{C}$ is a cartesian closed category and that N is an object of $\mathbb{C}$ equipped with $0 : \mathbf{1} \longrightarrow N$ and $s : N \longrightarrow N$. Take numerals $n : \mathbf{1} \longrightarrow N$ as usual and adopt the notion of (numeralwise) representability of partial functions from §2.4. A map $\Phi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{N}$ is represented by $F : (N \Rightarrow N)^k \times N^l \longrightarrow N$ just when for any numerals $n_1, \cdots, n_l : \mathbf{1} \longrightarrow N$ in $\mathbb{C}$ and maps $f_1, \cdots, f_k : N \longrightarrow N$ in $\mathbb{C}$ representing $\phi_1, \cdots, \phi_k \in \mathbb{P}$ we have

$$\Phi(\phi_1, \cdots, \phi_k, n_1, \cdots, n_l) = m \quad \text{iff} \quad F(f_1, \cdots, f_k, n_1, \cdots, n_l) = M : \quad \mathbf{1} \longrightarrow N.$$

A map $\Phi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{P}$ is represented by $F : (N \Rightarrow N)^k \times N^l \longrightarrow (N \Rightarrow N)$ just when the corresponding map $\Phi : \mathbb{P}^k \times \mathbb{N}^{l+1} \longrightarrow \mathbb{N}$ is represented by the exponential transpose $\overline{F} : (N \Rightarrow N)^k \times N^{l+1} \longrightarrow N$ in the sense just given.

Remark 8.3.2 We have given this definition quite generally but it has a clearer snese if one assumes that all finite partial functions are representable in $\mathbb{C}$ and that we are concerned only with the representability of continuous Φ. The reader may wish to reflect on the following easy observations about representability in the initial model $\mathbb{T}$.

Proposition 8.3.3 *(i) For any $F : (N \Rightarrow N)^k \times N^l \longrightarrow N$ in $\mathbb{T}$ there is a unique recursive operator $\Phi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{N}$ such that F represents Φ.*

(ii) If Φ is a continuous operator represented by F in $\mathbb{T}$ then Φ is the recursive operator represented by F. $\square$

We close this section by showing that a term F, which represents a continuous functional Φ in the initial model, does so also in $\mathbb{CA}$. The proof relies on a number of results which it seems best to collect together at this stage. First we need some simple facts about representability of partial functions from § 2.4.

(1) If $\mathbb{C}$ is an adequate model for PCF and $f : \iota^k \longrightarrow \iota$ represents $\phi : \mathbb{N}^k \longrightarrow \mathbb{N}$ in $\mathbb{T}$ then f represents ϕ in $\mathbb{C}$. (Proposition 2.4.16)

(2) For any model $\mathbb{C}$ of PCF if $n \lesssim s : \iota$ then $n = s : \iota$. (Proposition 3.2.2)

(3) For any model $\mathbb{C}$ of PCF, if $f \lesssim g : \iota^k \Rightarrow \iota$ represents $\phi, \psi : \mathbb{N}^k \longrightarrow \mathbb{N}$ then $\phi \subseteq \psi$.

(4) Suppose the model $\mathbb{C}$ of PCF satisfies the context lemma and htat (the observational quotient) $\hat{\mathbb{CA}}$ is standard. If $f, g : \iota^k \Rightarrow \iota$ represent $\phi, \psi : \mathbb{N}^k \longrightarrow \mathbb{N}$ with f strict in $\mathbb{CA}$, then

$$\phi \subseteq \psi \qquad \text{if and only if} \qquad f \lesssim g.$$

(Proposition 3.3.6)

We can apply these results to $\mathbb{CA}$.

(a) $\mathbb{CA}$ is standard and (hence) so is $\hat{\mathbb{CA}}$.
(b) $\mathbb{CA}$ is adequate (Proposition 7.4.1).
(c) $\mathbb{CA}$ satisfies the context lemma (Theorem 6.1.1).

Secondly the proof depends on some continuity properties of $\mathbb{CA}$.

(5) $\mathbb{CA}$ is enriched over dI-domains in such a way that a finite element $\tau : \iota \Rightarrow \iota$ is *either* a constant function and so PCF definable *or* strict and represents a finite partial function. In either case we can find $g : \iota \Rightarrow \iota$ in $\mathbb{T}$ such that if τ represents $\psi : \mathbb{N} \longrightarrow \mathbb{N}$ in $\mathbb{CA}$ then g represetns ψ in $\mathbb{T}$. It follows by (1) above that g represents ψ in $\mathbb{CA}$ and hence by (4) above that $g \simeq \tau$ in $\mathbb{CA}$.

Proposition 8.3.4 *Suppose that $F : (\iota \Rightarrow \iota)^k \times \times \iota^l \longrightarrow \iota$ represents the continuous functional $\Psi : \mathbb{P}^k \times \mathbb{N}^l \longrightarrow \mathbb{N}$ in the initial model $\mathbb{T}$ of PCF. Then F represents Ψ in $\mathbb{CA}$.*

Proof For simplicity we treat the case of $F : (\iota \Rightarrow \iota) \times \iota \longrightarrow \iota$ representing $\Psi : \mathbb{P} \times \mathbb{N} \longrightarrow \mathbb{N}$ in $\mathbb{T}$. Take $\sigma : (\iota \Rightarrow \iota)$ representing $\phi \in \mathbb{P}$ in $\mathbb{CA}$.

Suppose conversely that $F(\sigma)(n) = m$ in $\mathbb{CA}$. By (5) above we can take $\tau \leqslant_\Omega \sigma$ finite in $\iota \Rightarrow \iota$ with $F(\tau)(n) = m$; and we can find $g : \iota \Rightarrow \iota$ in $\mathbb{T}$ with $g \simeq \tau$ in $\mathbb{CA}$, and both g and τ representing $\psi : \mathbb{N} \longrightarrow \mathbb{N}$ in $\mathbb{CA}$. We deduce that $F(g)(n) = m$ in $\mathbb{CA}$ and hence as $\mathbb{CA}$ is adequate $F(g)(n) = m$ in $\mathbb{T}$. By assumption we deduce that $\Psi(\psi)(n) = m$. Now $\tau \leqslant_\Omega \sigma$ and so a *fortriori* $\tau \lesssim \sigma$ (reference?), and so by (4) above $\psi \subseteq \phi$. As Ψ is continuous we deduce $\Psi(\phi)(n) = m$. $\square$

8.4 Notational preliminaries

Before starting the main construction we establish some notation.

(i) We need a natural number code $\overline{m} = \langle\, m_0, \cdots, m_{k-1} \,\rangle$ for sequences of natural numbers; and a function to add a number at the head of a list

$$m * \langle\, m_0, \cdots, m_{k-1} \,\rangle \quad \stackrel{\text{def}}{=} \quad \langle\, m, m_0, \cdots, m_{k-1} \,\rangle.$$

(ii) We consider pure types $m \leqslant n+2$. By Lemma 8.1.2 one is a (PCF-definable) retract of the other. We write this as

$$e_m : m \longrightarrow n+2 \qquad \text{and} \qquad p_m : n+2 \longrightarrow m.$$

(iii) In analyzing $\sigma : m_0 \times \cdots \times m_{k-1} \longrightarrow 0$ we are led to consider

$$\sigma' \quad : \quad m \times m_0 \times \cdots \times m_{k-1} \quad \longrightarrow \quad 0$$

as we cannot (of course) keep fixed the number of arguments of "substrategies". Hence we are led to represent elements of $m_0 \times \cdots \times m_{k-1}$ as elements of $(\iota \Rightarrow n+2)$. We do this by means of PCF-definable retractions:

$$\begin{array}{lcll} \mathsf{fun} & : & m_0 \times \cdots \times m_{k-1} & \longrightarrow \quad (\iota \Rightarrow n+2) \\ \mathsf{tup} & : & (\iota \Rightarrow n+2) & \longrightarrow \quad m_0 \times \cdots \times m_{k-1} \end{array}$$

where

$$\begin{array}{rcl} \mathsf{fun}(a_0, \cdots, a_{k-1}) & = & \lambda x : \iota.\mathsf{cond}(\mathsf{eq}x0)(e_{m_0}a_0)(\mathsf{cond}(\mathsf{eq}x1)(e_{m_1}a_1)\cdots) \\ \mathsf{tup}(F) & = & (p_{m_0}(F(0)), \cdots, p_{m_{k-1}}(F(k-1))). \end{array}$$

(iv) We need a further PCF-definition. If $F : \iota \Rightarrow A$ and $a : A$ define $a * F : \iota \Rightarrow A$ by

$$a * F \quad \stackrel{\text{def}}{=} \quad \lambda x : \iota.\mathsf{cond}(\mathsf{eq}x0)a(F(\mathsf{pred}x)).$$

The final piece of notation that we need arises from a basic lemma which we need for the proof of universality. Recall that recursive operators Ψ' and Ψ from §8.2.

Proposition 8.4.1 *There exist* PCF*-terms H' and H that represent Ψ' and Ψ in the initial model $\mathbb{T}$.*

Proof In principle this is easy as we have simple equations:

$$\begin{array}{rcl} f_{\sigma'}(u) & = & f_\sigma(\text{“}[\cdot(\cdot u\text{”}) \\ f_{\sigma_c}(\text{“}[\cdot v\text{”}) & = & f_\sigma(\text{“}[\cdot(\cdot)_c \cdot v\text{”}) \end{array}$$

determining $f_{\sigma'}$ and f_{σ_c} in terms of f_σ; however the equivalence required for representability requires a little thought. We treat the case of H'; the case of H is similar.

First we need to observe in effect that being a finite play in accord with a strategy is semi-decidable. One can readily define using fixed points a term

$$\mathsf{accord} \quad : \quad (N \Rightarrow N) \times N \quad \longrightarrow \quad N$$

which represents the operator *accord* : $\mathbb{P} \times \mathbb{N} \longrightarrow \mathbb{N}$ defined by

$$accord(\phi, n) = 0 \quad \text{iff} \quad \begin{cases} n = \#u \text{ for some play } u \text{ in accord with a} \\ \text{finite innocent strategy } \sigma \text{ with } \phi_\sigma \subseteq \phi. \end{cases}$$

(The ideas used in the definition of *accord* were introduced in §5.6.) Then we can define H' in a proto-PCF by

$$H'(h)(n) \quad = \quad \mathsf{if}\ (n = \#u \ \mathsf{and}\ \mathsf{accord}(h, \#``[\cdot(\cdot u") = 0)\ \mathsf{then}\ h(\#``[\cdot(\cdot u").$$

To show that H' represents Ψ' take $f : N \longrightarrow N$ in $\mathbb{T}$ representing $\phi : \mathbb{N} \longrightarrow \mathbb{N}$ and $n \in \mathbb{N}$.

Suppose first that $\Psi'(\phi)(n) = m$. Then there is $\Phi_\tau \subseteq \phi$, τ a finite innocent strategy and $\Phi_{\tau'}(n) = m$. In particular it follows that $n = \#u$ where u is play in accord with τ', so that "$[\cdot(\cdot u$" is a play in accord with τ. As accord represents accord we deduce that $\mathsf{accord}(f, \#``[\cdot(\cdot u") = 0$ and so $H'(f)(n) = f(\#``[\cdot(\cdot u")$. But $\phi_\tau(\#``[\cdot(\cdot u") = \phi_{\tau'}(n) = m$ so that $\phi(\#``[\cdot(\cdot u") = m$ and so $f(\#``[\cdot(\cdot u") = m$ as f represents ϕ. Thus $H'(f)(n) = m$.

Conversely suppose that $H'(f)(n) = m$. Then $n = \#u$ where $\mathsf{accord}(f, \#``[\cdot(\cdot u") = 0$ and $f(\#``[\cdot(\cdot u") = m$. Aso accord represents *accord* and f represents ϕ we have $accord(\phi, \#``[\cdot(\cdot u") = 0$ so that $\#``[\cdot(\cdot u$" is a play in accord with some finite innocent σ with $\phi_\sigma \subseteq \phi$. But also $\phi(\#``[\cdot(\cdot u") = m$ so we can extend $\sigma$ to $\tau$ with $\phi_\tau \subseteq \phi$ and $\phi_\tau(\#``[\cdot(\cdot u") = m$. But then $\phi_{\tau'}(n) = \phi_{\tau'}(\#u) = m$ so that $\Phi'(\phi)(n) = m$. □

8.5 A universal function

The argument for universality rests on the construction of a suitable universal function. We shall show how to define, for each natural number n, a PCF-term

$$U \quad : \quad \iota \times (\iota \Rightarrow \iota) \times (\iota \Rightarrow (n+2)) \Rightarrow \iota$$

with a property which we now spell out.

Suppose we have the following data:

(i) A code $\overline{m} = \langle m_0, \cdots, m_{k-1} \rangle$ for a sequence $m_0, \cdots, m_{k-1} \leqslant n+2$ of pure types. We write $\overline{m}$ also for the corresponding strategy in the arena ι.

(ii) An innocent strategy $\sigma : m_0 \times \cdots \times m_{k-1} \longrightarrow 0$ in $\mathbb{CA}$; f_σ is the representing innocent function and ϕ_σ the partial function code (explained in §5.6). We write ϕ_σ also for the standard representation of ϕ_σ as an innocent strategy in the arena $\iota \Rightarrow \iota$ (see Remark 7.1.2).

Consider the composite in $\mathbb{CA}$

$$(\iota \Rightarrow (n+2)) \xrightarrow{\mathsf{tup}} m_0 \times \cdots \times m_{k-1} \xrightarrow{\quad\sigma\quad} \iota.$$

The universal property of U is that this is observationally equivalent to (the transpose of)

$$\lambda F : (\iota \Rightarrow (n+2)).U(\overline{m}, \phi_\sigma, F) \quad : \quad (\iota \Rightarrow (n+2)) \Rightarrow \iota.$$

If we introduce a free variable F of type $(\iota \Rightarrow (n+2))$ and unravel the definition of tup, we can write the required property as

$$U(\overline{m}, \phi_\sigma, F) \quad \simeq \quad \sigma(p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad : \quad (\iota \Rightarrow (n+2)) \longrightarrow \iota$$

Proposition 8.5.1 *For each natural number n there is a* PCF*-term*

$$U \quad : \quad \iota \times (\iota \Rightarrow \iota) \times (\iota \Rightarrow (n+2)) \Rightarrow \iota$$

so that for any $\overline{m} = \langle m_0, \cdots, m_{k-1} \rangle$ and $\sigma : m_0 \times \cdots \times m_{k-1} \longrightarrow 0$ in $\mathbb{CA}$,

$$(\dagger) \qquad U(\overline{m}, \phi_\sigma, F) \quad \simeq \quad \sigma(p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad : \quad (\iota \Rightarrow (n+2)) \longrightarrow \iota.$$

Proof Define U using the fixed-point operator to satisfy the following informal equation:

$$U(\overline{m}, \phi, F) \quad = \quad \begin{cases} \text{if } \phi(\text{“[”}) = \text{“]}_d\text{” then } d \text{ else} \\ \text{if } \phi(\text{“[”}) = \text{“(” – the initial question in the } i\text{-th game } m_i \text{ then} \end{cases}$$

- *either* (in the general case that $m_i = m+2$) if $p_{m_i}(F(i))(\lambda a : m.U(m * \overline{m}, H'(\phi), e_m(a) * F) = c$ then $U(\overline{m}, H_c(\phi), F)$
- *or* (in case $m_i = 1$) if $p_{m_i}(F(i))(U(\overline{m}, H'(\phi), F)) = c$ then $U(\overline{m}, H_c(\phi), F)$
- *or* (in case $m_i = 0$) if $p_{m_i}(F(i)) = c$ then $U(\overline{m}, H_c(\phi), F)$;

which we have written in a kind of proto-PCF using notation introduced in the previous section. It is easy to translate this (in an “up to observational equivalence” sense) into true PCF; the only unobvious point of detail is made clear in the proof below.

First we note that a simple continuity argument shows that it suffices to prove that U has the stated property (†) for finite (compact) strategies σ. Secondly note that the context lemma for $\mathbb{CA}$ turns (†) into what is essentially a point-wise claim.

To prove (†) for a finite σ we proceed by induction on the structure of σ. We recall the analysis from our preamble. In the undefined case when σ does not respond and the constant case when σ responds at once, the result is clear. (F does not come into it at all.) Hence we turn to the inductive case.

We deal with the inductive case in the general circumstances that $m_i = m + 2 \geqslant 2$, leaving the other simpler circumstances to the reader. Take a finite strategy $\sigma : m_0 \times \cdots \times m_{k-0} \longrightarrow 0$. By our induction hypothesis we have the result (†) for σ' and also for each σ_c. In particular we have

$$U(m * \overline{m}, \phi_{\sigma'}, G) \quad \simeq \quad \sigma'(p_m G(0), p_{m_0}G(1), \cdots, p_{m_{k-1}}G(k)) \quad : \quad (\iota \Rightarrow (n+2)) \longrightarrow \iota$$

It follows that

$$U(m * \overline{m}, \phi_{\sigma'}, e_m(a) * F) \quad \simeq \quad \sigma'(a, p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad : \quad m \times (\iota \Rightarrow (n+2)) \longrightarrow \iota,$$

and so

$$\lambda a.U(m * \overline{m}, \phi_{\sigma'}, e_m(a) * F) \simeq \lambda a.\sigma'(a, p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) : (\iota \Rightarrow (n+2)) \longrightarrow (m+1).$$

Applying $p_{m_i}(F(i))$ we deduce that

$$\begin{aligned} & p_{m_i}(F(i))(\lambda a.U(m * \overline{m}, \phi_{\sigma'}, e_m(a) * F)) \\ \simeq \quad & p_{m_i}(F(i))(\lambda a.\sigma'(a, p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1))) \end{aligned}$$

as maps $(\iota \Rightarrow (n+2)) \longrightarrow 0$.

Now we aim to show

$$U(\overline{m}, \phi_\sigma, F) \quad \simeq \quad \sigma(p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad : \quad (\iota \Rightarrow (n+2)) \longrightarrow 0.$$

By the context lemma for $\mathbb{CA}$ it is enough to prove this equivalence pointwise, so take $\rho : \iota \Rightarrow n+2$ an innocent strategy. Suppose then that

$$\sigma(p_{m_0}\rho(0), \cdots, p_{m_{k-1}}\rho(k-1)) \quad = \quad d$$

a value in $\mathbb{CA}$. First it follows from our analysis of σ that we must have

$$p_{m_i}\rho(i)(\lambda a.\sigma'(a, p_{m_0}\rho(0), \cdots, p_{m_{k-1}}\rho(k-1))) \quad = \quad c$$

a value in $\mathbb{CA}$. But then we have just seen that it is a consequence of the induction hypothesis that

$$p_{m_i}\rho(i)(\lambda a.U(m * \overline{m}, \phi_{\sigma'}, e_m(a) * \rho)) \quad = \quad c.$$

Now $\phi_{\sigma'} \simeq H'(\phi_\sigma)$ by Proposition 8.3.4 and Proposition 8.4.1, so we have

$$p_{m_i}\rho(i)(\lambda a.U(m * \overline{m}, H'(\phi_\sigma), e_m(a) * \rho)) \quad = \quad c.$$

Secondly it follows from our analysis of σ that

$$\sigma_c(p_{m_0}\rho(0), \cdots, p_{m_{k-1}}\rho(k-1)) \quad = \quad d.$$

But then again by the induction hypothesis

$$U(\overline{m}, \phi_{\sigma_c}, \rho) \quad = \quad d.$$

Again $\phi_{\sigma_c} \simeq H_c(\phi_\sigma)$ by Propositions 8.3.4 and 8.4.1, so we have

$$U(\overline{m}, H_c(\phi_\sigma), \rho) \quad = \quad d.$$

Putting these two facts together we see from the definition of U that

$$U(\overline{m}, \phi_\sigma, \rho) \quad = \quad d.$$

We have shown that $\sigma(p_{m_0}\rho(0), \cdots, p_{m_{k-1}}\rho(k-1)) = d$ implies $U(\overline{m}, \phi_\sigma, \rho) = d$, which is enough to show

$$\sigma(p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad \lesssim \quad U(\overline{m}, \phi_\sigma, F) \quad : \quad (\iota \Rightarrow (n+2)) \longrightarrow \iota.$$

Now to establish the opposite inequality, take $\rho : \iota \Rightarrow n+2$ an innocent strategy again and suppose that

$$U(\overline{m}, \phi_\sigma, \rho) \quad = \quad d$$

a value in $\mathbb{CA}$. Now we require that the translation of our proto-PCF in true PCF is such that this means that

$$p_{m_i}\rho(i)(\lambda a.U(m * \overline{m}, H'(\phi_\sigma), e_m(a) * \rho)) \quad = \quad c$$

is a value in $\mathbb{CA}$, and that

$$U(\overline{m}, H_c(\phi_\sigma), \rho) \quad = \quad d.$$

(This is easy to arrange.)

It follows that we can run the argument just given backwards to deduce (in view of our analysis of the output behaviour of σ) that

$$\sigma(p_{m_0}\rho(0), \cdots, p_{m_{k-1}}\rho(k-1)) \quad = \quad d.$$

This show that

$$U(\overline{m}, \phi_\sigma, F) \quad \lesssim \quad \sigma(p_{m_0}F(0), \cdots, p_{m_{k-1}}F(k-1)) \quad : \quad (\iota \Rightarrow (n+2)) \Rightarrow \iota.$$

Thus we have established (†).

Inductively we have established (†) for all finite strategies σ and hence by continuity for all σ. This completes the proof. □

We now come to the results which we discussed at the start of this section. In case $A = n+2$ is a pure type, the function u discussed there is simply a special case of the function U of the last proposition. But it follows easily from the fact that any object in $\mathbb{T}$ is a retract of (the interpretation of) a pure type, that universality at pure types implies universality at all PCF-types. Hence we have established our main results.

Theorem 8.5.2 (Universality) *(i) Every map between* PCF*-types in* $\hat{\mathbb{CA}}$ *is* PCF*-definable in some (partial) function parameter.*

(ii) Every map between PCF*-types in* $\hat{\mathbb{RA}}$ *is* PCF*-definable.* □

Remark 8.5.3 We could prove a more general version of Proposition 8.5.1 if we replaced the code

$$\overline{m} \quad = \quad \langle\, m_0, \cdots, m_{k-1} \,\rangle$$

with some system of codes for all possible sequence of arguments of height $\leqslant n+2$. This would avoid the detour through pure types at the cost of some notational complexity.

9 Conclusions and further directions

In this work (comprising Parts I, II and III) we begin by giving a survey of the so-called full abstraction problem for PCF tracing its roots to old foundational problems in recursion theory considered by Platek and also (in a related but different direction) by Kleene and others.

9.1 Comparison with related work

Gandy's approach

Robin Gandy has for some years been engaged in a project to refine the dialogue ideas considered by Kleene [Kle78] so as to provide a model for PCF satisfying the universality theorem. Latterly in collaboration with his student Giovanni Pani he has produced many examples and counterexamples; and also at least the outline of a definition. Our comments on this approach are based on discussions with Gandy and Pani and on a handwritten account by Gandy [Gan93].

The discipline of questions and answers which we use has long been a part of Gandy's framework; he calls it the "no dangling question mark" condition, and one of us (Hyland) learnt its significance from him. However further restrictions are needed to capture PCF definability and we do not fully understand the other components of Gandy's approach. In [Gan93] Gandy uses a notion of relevant record which is superficially similar to our notions of P-view and O-view. We are unsure of the exact form and the force of Gandy's notion. On the one hand Gandy raises questions about consistency and extensionality which simply do not arise for innocent strategies; and Pani's counterexample which motivates further restrictions on the notion of a good strategy is not given by an innocent strategy in our sense. On the other hand parts of [Gan93] suggest, and discussion with Gandy and Pani confirm, that they have their eyes on a greater prize. For they appear to regard a relevant record as if it coincides with purely extensional information and were this carried through they would meet the Jung-Stoughton criterion.

Sazonov's approach

In the 1970's Sazonov (see for example [Saz75], [Saz76] and [Saz76]) outlined a concrete machine-oriented approach to the problem of providing a model for PCF satisfying the universality theorem. This work is not as well known as it deserves to be and we give a brief indication of its nature.

In Sazonov's approach a (recursively sequential) function of higher type is represented by some Turing machines with oracle (TMO). A TMO F communicates with its arguments $G_1, \cdots, G_k$ (all assumed to be of simple type) by asking for the value of one such (G_1 say) on TMO's of types appropriate to be arguments (of G_1) codes for which are provided by the TMO F. Sequentiality is ensured by the requirement that a numerical answer must be provided (by G_1) before F can continue computation. What we effect by the condition of innocence is provided more directly by Sazonov via the requirements that questions essentially ask for extensional information (that this is so can be seen by a straightforward inductive argument) and that the questioning TMO only receives the answer and not how it is got. The several arguments $G_1, \cdots, G_k$ of F operate independently and when they are called again a fresh copy of the TMO is made available.

These ideas, while hard to formalize (for example the interpretation of the TMOs as extensional functions of finite type is given directly by a least fixed point), are if anything more immediately intuitive than those involved in our more abstract setting of games and innocent strategies. On the other hand Sazonov's approach has a lurking syntactic quality: his questions have a specific syntactic form involving application in PCF. There is a sense however in which the relation of Sazonov's approach with ours is very close. The order of communication of the TMOs precisely mirrors the pattern of questions and answers in our approach. This is clearly demonstrated by a translation of innocent strategies into Milner's π-calculus. (Elaborations of this will be presented elsewhere). The results can also be seen as control information for TMOs in Sazonov's sense.

The AJM approach At the same time as we were working on out treatment of PCF in terms of innocent strategies, Abramsky, Jagadeesan and Malacaria were developing a different approach also involving games and strategies [AJM94]. We make some very tentative remarks about the relation between the AJM game-theoretic approach and our own.

Both the two approaches make use of the discipline of questions and answers (which had been identified earlier by Gandy), but they differ in terms of the notion of strategy. Our approach exploits out new notion of innocent strategy, while AJM use the simple notion of history-free strategy which was already considered in Abramsky and Jagadeesan work on game semantics for multiplicative linear logic [AJ94]. On the other hand AJM rely on a rather subtle notion of move; moves and plays are considered up to equivalence under some group action. By contrast our notion of move is relatively straightforward.

It seems that the intensional models of PCF which result from the two approaches may well coincide. However the underlying linear categories appear to be different. our impression is that a function such as strict-and (defined on the obvious simple boolean game) will not be linear in the AJM setting, which it will be in ours. This suggests that the AJM analysis is in some sense deeper than ours and that our linear setting may be obtainable from theirs (for example as the Kleisli category of some comonad).

[To be included: [Win94], [PW94], [OR94].]

9.2 Further directions

Our study of the category $\mathbb{CA}$ of computational arenas and innocent strategies (in Part II) has been quite extensive, but it is certainly not complete. In Part III we show that the category $\mathbb{CA}$ gives rise to a fully abstract and universal model of PCF. A number of questions pertaining to the fully abstract game model remain open, some of which seem conceptually important. In addition there are many possibilities for extension and generalization of our results. In this section we pick out some of the more promising topics for further research and sketch some preliminary results. A fuller account will be presented elsewhere.

A calculus for describing strategies See [HO95]

Linear decomposition of $\mathbb{CA}$

In this work we have consciously chosen a simple framework of games determined by computational arenas. The style of our approach is close to that of Kleene and Gandy in one tradition, and to Berry and Curien in another. We are aware of a wider framework (larger categories of games) and of a linear decomposition of the kind pioneered by Girard but have chosen not to go into these things here.

Characterization of higher-type sequentiality

In our view one cannot claim to have understood higher-type sequentiality until a satisfactory axiomatic characterization has been obtained. This is definitely related to what we call "Kleene's problem" in §1.4 (see also the discussion in §1.3). We are inclined to believe that this is where the thrust of the full abstraction problem of PCF really is.

Solution of domain equations in $\mathbb{CA}$

Relating dialogue games to Geometry of Interaction

Is the observational quotient enriched over CPOs?

The observational quotient $\hat{\mathbb{CA}}$ is enriched over the category of posets. Is it enriched over the category of CPOs (and continuous functions)? We do not know the answer to this question. An obvious way to attack the problem is to take advantage of the Strong Definability Theorem (7.3.4) and argue syntactically.

Acknowledgements

We would like to thank Pierre-Louis Curien for his helpful suggestions.

A Proof of the projection lemma

The proof is rather complex and it requires a detailed analysis of what we call *bounded segments* in a function space arena.

Bounded segments

Let s be a legal position of the arena $A \Rightarrow B$. A segment θ of s beginning with a P-move x and ending with an O-move y is said to be *bounded* if the two end-moves x and y are an explicitly justifying pair, *i.e.* either both are questions, and x explicitly justifies y or the question x is explicitly answered by the answer y. Henceforth whenever x and y are thus related, we say that x explicitly justifies y. We call θ an (A, a)*-bounded segment* (respectively a B-bounded segment) if either, and hence both, end-moves are in the component (A, a) for some instance a of an initial A-move occurring in s (respectively B). We shall write (A, a)-bounded simply as A-bounded. The two simplest bounded segments have the shapes $\circ \cdot \bullet$ and $\circ \cdot \bullet \cdot \circ \cdot \bullet$ respectively. In both cases all moves of the bounded segment belong to the same component.

We consider two ways by which a bounded segment may be decomposed. First, **spine decomposition.**

Lemma A.0.1 *Any bounded segment θ with end-moves x and y may be decomposed in the following way:*

$$\overbrace{\overset{x}{\circ}\cdot\underbrace{\boxed{p_m\cdots q_m}}_{\xi_m}\ \cdots\ \underbrace{\boxed{p_i\cdots q_i}}_{\xi_i}\ \cdots\ \underbrace{\boxed{p_1\cdots q_1}}_{\xi_1}\cdot\overset{y}{\bullet}}^{\theta};$$

where p_i is an O-move which explicitly justifies the P-move q_i, for each $1 \leqslant i \leqslant m$ and for some $m \geqslant 0$.

Proof For suppose not, then for some $m \geqslant 1$, we have the following decomposition:

$$\underbrace{\boxed{p_m\cdots x\cdots q_m}}_{\xi_m}\ \cdots\ \underbrace{\boxed{p_i\cdots q_i}}_{\xi_i}\ \cdots\ \underbrace{\boxed{p_1\cdots q_1}}_{\xi_1}\cdot\overset{y}{\bullet}$$

where x is in ξ_m but is not q_m. By the visibility condition, x which explicitly justifies y appears in $\llcorner s_{\leqslant q_1}\lrcorner$. But

$$\llcorner s_{\leqslant q_1}\lrcorner \quad = \quad \llcorner s_{<p_m}\lrcorner\cdot p_m\cdot q_m\cdots p_1\cdot q_1;$$

which does not include x. Hence we get a contradiction. $\square$

Given a bounded segment θ as in the preceding lemma, we call the following sub-segment of θ

$$\overset{x}{\circ}\cdot\overset{p_m}{\bullet}\cdot\overset{q_m}{\circ}\cdots\overset{p_1}{\bullet}\cdot\overset{q_1}{\circ}\cdot\overset{y}{\bullet}$$

the *spine* of θ.

Projection decomposition A bounded segment θ with end-moves x and y may be decomposed in terms of bounded segments in the following way:

$$\underbrace{\boxed{\circ\cdots x\cdots\bullet}}_{\theta_{n+1}}\cdot\underbrace{\boxed{\circ\cdots\bullet}}_{\theta_n}\ \cdots\ \underbrace{\boxed{\circ\cdots\bullet}}_{\theta_1}\cdot\overset{y^-}{\circ}\cdot\overset{y}{\bullet}$$

where x and y are explicitly justifying pair; and for each $1 \leqslant i \leqslant n+1$, θ_i is a bounded segment (with end-moves x_i and y_i) which may be either A-bounded or B-bounded. Let y^- be the move which immediately precedes y in θ. Note that x may be the left end-move in θ_{n+1}. Observe that apart from y^-, every move in θ belongs to a (unique) constituent bounded segment.

Lemma A.0.2 *(i) Suppose θ is A-bounded. For any $1 \leqslant i \leqslant n+1$, let m be a P-move in the bounded segment θ_i. If m appears in $\llcorner s_{\leqslant y^-} \lrcorner$ then θ_i is an A-bounded segment. In particular since by visibility, the P-move x (which explicitly justifies y) is in $\llcorner s_{\leqslant y^-} \lrcorner$, we conclude that the segment θ_{n+1} is A-bounded.*

(ii) The statement obtained from (i) by replacing the adjective A-bounded with B-bounded is valid.

Proof To prove the lemma, we use the following claim:

Claim. The O-view $\llcorner s_{\leqslant y^-} \lrcorner$ has the following form:

$$\llcorner s_{\leqslant x_{i_l}} \lrcorner \cdot \boxed{\;i_l\;} \cdot \overset{y_{i_l}}{\bullet} \cdot \underbrace{\overset{x_{i_{l-1}}}{\circ} \cdot \boxed{\;i_{l-1}\;} \cdot \overset{y_{i_{l-1}}}{\bullet}}_{\text{spine of } \theta_{i_{l-1}}} \quad \cdots \quad \underbrace{\overset{x_{i_1}}{\circ} \cdot \boxed{\;i_1\;} \cdot \overset{y_{i_1}}{\bullet}}_{\text{spine of } \theta_{i_1}} \cdot \overset{y^-}{\circ};$$

where

- for some $l \geqslant 1$, the sequence $i_1, i_2, \cdots, i_l$ is a subsequence (not necessarily initial) of $1, 2, \cdots, n+1$;
- y_{i_1} explicitly justifies y^-; and for each $1 \leqslant j \leqslant l$, $y_{i_{j+1}}$ explicitly justifies x_{i_j};
- for each $1 \leqslant j \leqslant l$, the segment $x_{i_j} \cdot \boxed{\;i_j\;} \cdot y_{i_j}$ is the spine of the bounded segment θ_{i_j} which is A-bounded,
- m is an element of $\{ x_{i_1}, x_{i_2}, \cdots x_{i_l}, y^- \}$.

To prove the claim, first observe that by visibility the explicitly justifying move of y^- must appear in the P-view:

$$\ulcorner s_{\leqslant y_1} \urcorner \quad = \quad \ulcorner s_{< x_{n+1}} \urcorner \cdot \overset{x_{n+1}}{\circ} \cdot \overset{y_{n+1}}{\bullet} \cdot \overset{x_n}{\circ} \cdot \overset{y_n}{\bullet} \quad \cdots \quad \overset{x_1}{\circ} \cdot \overset{y_1}{\bullet};$$

where x_i explicitly justifies y_i for each $1 \leqslant i \leqslant n+1$. Hence y^- is explicitly justified by y_{i_1}, for some $1 \leqslant i_1 \leqslant n+1$ (and not by a move from $\ulcorner s_{< x_{n+1}} \urcorner$, for if so then x is excluded from $\llcorner s_{< y^-} \lrcorner$ thus violating the visibility condition applied to y). Note that y_{i_1} is an A-move; hence the segment θ_{i_1} is A-bounded. By Lemma A.0.1 $\llcorner s_{\leqslant y^-} \lrcorner$ has the form:

$$\llcorner s_{\leqslant x_{i_1}} \lrcorner \cdot \boxed{\;i_1\;} \cdot y_{i_1} \cdot y^-.$$

Inductively suppose $\llcorner s_{\leqslant y^-} \lrcorner$ has the following form:

$$\llcorner s_{\leqslant x_{i_j}} \lrcorner \cdot \boxed{\;i_j\;} \cdot y_{i_j} \cdot x_{i_{j-1}} \cdot \boxed{\;i_{j-1}\;} \cdot y_{i_{j-1}} \quad \cdots \quad x_{i_1} \cdot \boxed{\;i_1\;} \cdot y_{i_1} \cdot y^-;$$

where the segments $\theta_{i_1}, \cdots, \theta_{i_j}$ are all A-bounded. Now the last two moves of $\llcorner s_{\leqslant x_{i_j}} \lrcorner$ are w and x_{i_j} where w is the O-move which explicitly justifies x_{i_j}. By the visibility condition w appears in the P-view:

$$\ulcorner s_{\leqslant y_{i_j+1}} \urcorner \quad = \quad \ulcorner s_{< x_{n+1}} \urcorner \cdot x_{n+1} \overset{x_{n+1}}{\circ} \cdot \overset{y_{n+1}}{\bullet} \quad \cdots \quad \overset{x_{i_j+1}}{\circ} \cdot \overset{y_{i_j+1}}{\bullet}.$$

If w is in $\ulcorner s_{<x_{n+1}} \urcorner$ then set l to be j; otherwise set i_{j+1} to k such that $y_k = w$. Note that $y_{i_{j+1}}$ is an A-move; hence the segment $\theta_{i_{j+1}}$ is A-bounded. By Lemma A.0.1, we have

$$\llcorner s_{\leqslant y^-} \lrcorner \quad = \quad \llcorner s_{\leqslant x_{i_{j+1}}} \lrcorner \cdot \boxed{\quad i_{j+1} \quad} \cdot y_{i_{j+1}} \cdot \underbrace{x_{i_j} \cdot \boxed{\quad i_j \quad} \cdot y_{i_j}}_{\text{spine of } \theta_{i_j}} \quad \cdots \quad x_{i_1} \cdot \boxed{\quad i_1 \quad} \cdot y_{i_1} \cdot y^- .$$

Hence the claim is established.

Let m be a P-move in θ_i. Suppose m appears in $\llcorner s_{\leqslant y^-} \lrcorner$. Then, by the Claim, m appears in the spine of some θ_{i_j} which is an A-bounded segment. Hence the lemma is proved. $\square$

Let θ be an A-bounded segment in a legal position s of an arena $A \Rightarrow B$ with end-moves x and y. By an abuse of notation we define $\ulcorner \theta \upharpoonright B \urcorner$ as the subsequence of $\ulcorner s_{\leqslant y} \upharpoonright B \urcorner$ consisting only of moves in θ occurring immediately after (and not including) x.

Lemma A.0.3 *Let θ be an A-bounded segment in s with end-moves x and y.*

(i) The segment $\ulcorner \theta \upharpoonright B \urcorner$ has the following form:

$$\ulcorner \theta \upharpoonright B \urcorner \quad = \quad \overset{p_r}{\circ} \cdot \overset{q_r}{\bullet} \quad \cdots \quad \overset{p_1}{\circ} \cdot \overset{q_1}{\bullet}$$

for some $r \geqslant 0$ (as opposed to $\bullet \cdot \circ \cdot \bullet \quad \cdots \quad \circ \cdot \bullet$) where p_i explicitly justifies q_i for each $1 \leqslant i \leqslant r$. Note that $p_i \cdots q_i$ is a B-bounded segment in s, for each $1 \leqslant i \leqslant r$.

(ii) For any P-move m in θ which appears in $\llcorner s_{<y} \lrcorner$, m does not belong to any of the B-bounded segment $p_i \cdots q_i$ for $1 \leqslant i \leqslant r$.

The assertions obtained from the preceding by interchanging A-bounded segments with B-bounded segments remain valid.

Proof We prove both (i) and (ii) by induction on the length of $s_{\leqslant y}$. The base case of θ of the form $\circ \cdot \bullet$ for both (i) and (ii) is trivial. For the inductive case, consider the projection decomposition of the A-bounded segment θ as follows (using the same notation as in Lemma A.0.2):

$$\underbrace{\boxed{\circ \cdots x \cdots \bullet}}_{\theta_{n+1}} \cdot \underbrace{\boxed{\circ \cdots \bullet}}_{\theta_n} \quad \cdots \quad \underbrace{\boxed{\circ \cdots \bullet}}_{\theta_1} \cdot \overset{y^-}{\circ} \cdot \overset{y}{\bullet}$$

where for $1 \leqslant i \leqslant n+1$, θ_i is a bounded segment with end-moves x_i and y_i.

For each $1 \leqslant i \leqslant n+1$, if θ_i is A-bounded then by the induction hypothesis of (i), $\ulcorner \theta_i \upharpoonright B \urcorner$ is $p_{i,r_i} \cdot q_{i,r_i} \cdots p_{i,1} \cdot q_{i,1}$, for some $r_i \geqslant 0$. Note that x is by assumption a P-move in θ_{n+1}, and so, by Lemma A.0.2, θ_{n+1} is an A-bounded segment. Since x appears in $\llcorner s_{<y} \lrcorner$, by the same analysis as the Claim in the proof of Lemma A.0.2, x appears in $\llcorner s_{<y_{n+1}} \lrcorner$. Applying the induction hypothesis of (ii) to the A-bounded segment θ_{n+1} (which has end-moves x_{n+1} and y_{n+1}), we infer that x does not appear in any of the B-bounded segment $p_{n+1,i} \cdots q_{n+1,i}$, for any $1 \leqslant i \leqslant r_{n+1}$. So suppose x appears in between $q_{n+1,l+1}$ and $p_{n+1,l}$, for some $1 \leqslant l < r_{n+1}$. Then $\ulcorner \theta \upharpoonright B \urcorner$ is $\gamma_{n+1} \cdot \gamma_n \cdots \gamma_1$ where the segment γ_i's are defined as follows:

- for $1 \leqslant i \leqslant n$, we have

$$\gamma_i \quad = \quad \begin{cases} x_i \cdot y_i & \text{if } \theta_i \text{ is a } B\text{-bounded segment,} \\ \ulcorner \theta_i \upharpoonright B \urcorner & \text{if } \theta_i \text{ is an } A\text{-bounded segment;} \end{cases}$$

- $\gamma_{n+1} = p_{n+1,l} \cdot q_{n+1,l} \cdots p_{n+1,1} \cdot q_{n+1,1}$.

Hence (i) is established for the inductive case. As for (ii) take any P-move in θ which appears in $\llcorner s_{<y} \lrcorner$. Then m appears in $\llcorner s_{<y_j} \lrcorner$ for some A-bounded segment θ_j. Applying the induction hypothesis of (ii) to the A-bounded segment θ_j, we infer that m does not appear in the B-bounded segment $p_{j,k} \cdots q_{j,k}$, for each $1 \leqslant k \leqslant r_j$. Hence the result follows. $\square$

We are now ready to prove the projection lemma.

Proof of the projection lemma

(i). We prove by induction on the length of s. The base case is immediate but the inductive case requires some work. If the last move m is a P-move, then $\ulcorner s \urcorner = \ulcorner s_{<m} \urcorner \cdot m$. There are two cases. If the move preceding m in s is in B, then we have:

$$\begin{aligned} \ulcorner s \urcorner \upharpoonright B &= \ulcorner s_{<m} \urcorner \upharpoonright B \cdot m \quad \text{by the induction hypo.} \\ &\preccurlyeq \ulcorner s_{<m} \upharpoonright B \urcorner \cdot m \\ &= \ulcorner s \upharpoonright B \urcorner. \end{aligned}$$

Suppose the move y_1 preceding m is in A. Let $\underline{m}$ be the B-move preceding m in $\ulcorner s \urcorner$. We have

$$\ulcorner s \urcorner \quad = \quad \ulcorner s_{\leqslant \underline{m}} \urcorner \cdot \underbrace{\overset{x_r}{\circ} \cdot \overset{y_r}{\bullet} \cdots \overset{x_1}{\circ} \cdot \overset{y_1}{\bullet}}_{A\text{-moves}} \cdot \overset{m}{\circ}; \tag{5}$$

where x_i explicitly justifies y_i, for each $1 \leqslant i \leqslant r$, for some $r \geqslant 1$, and they are all A-moves.

Example 4.4.3 shows that for each $1 \leqslant i \leqslant r$, the A-bounded segment $x_i \cdots y_i$ in s may contain B-moves, some of which may appear in $\ulcorner s \upharpoonright B \urcorner$. To complete the argument for the inductive case, it suffices to establish the following:

Claim. $\ulcorner s \upharpoonright B \urcorner = \ulcorner s_{\leqslant \underline{m}} \upharpoonright B \urcorner \cdot \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_r} \quad \cdots \quad \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_2} \cdot \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_1} \cdot m$ where each δ_i is a segment of B-moves.

For then, from (5) we have

$$\begin{aligned} \ulcorner s \urcorner \upharpoonright B &= \ulcorner s_{\leqslant \underline{m}} \urcorner \upharpoonright B \cdot m && \text{by the induction hypo.} \\ &\preccurlyeq \ulcorner s_{\leqslant \underline{m}} \upharpoonright B \urcorner \cdot m \\ &\preccurlyeq \ulcorner s_{\leqslant \underline{m}} \upharpoonright B \urcorner \cdot \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_r} \quad \cdots \quad \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_2} \cdot \underbrace{\boxed{\circ \cdots \bullet}}_{\delta_1} \cdot m && \text{by the Claim} \\ &= \ulcorner s \upharpoonright B \urcorner. \end{aligned}$$

It remains to prove the Claim. By (5) we infer that s has the following form:

$$s_{\leqslant \underline{m}} \cdot \underbrace{\boxed{x_r \cdots y_r}}_{\theta_r} \quad \cdots \quad \underbrace{\boxed{x_1 \cdots y_1}}_{\theta_1} \cdot m;$$

where each segment $\theta_i \equiv x_i \cdots y_i$ is an A-bounded segment. It suffices to note that by Lemma A.0.3(i) each δ_i is just $\ulcorner \theta_i \upharpoonright B \urcorner$.

The case of m being an O-move reduces to the preceding case. The proof for (ii) is entirely symmetrical, and we omit it. $\square$

References

[AJ94] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59:543–574, 1994.

[AJM94] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF (extended abstract). In *Theoretical Aspects of Computer Software: TACS'94, Sendai, Japan*, pages 1–15. Springer-Verlag, 1994. LNCS Vol. 789.

[AO93] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993.

[Bar84] H. Barendregt. *The Lambda Calculus*. North-Holland, revised edition, 1984.

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

[BCL86] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1986.

[BE91] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In *Proc. 6th Annual IEEE Symp. Logic in Computer Science*, pages 138–145. IEEE Computer Society Press, 1991.

[BE93] A. Bucciarelli and T. Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–292, 1993.

[Ber78] G. Berry. Stable models of typed lambda calculus. In *Proc. 5th Colloquium on Algorithms, Languages and Programming*, pages 72–89. Springer-Verlag, 1978. LNCS Vol. 62.

[Ber81] G. Berry. Some syntactic and categorical constructions of lambda calculus models. Papport de Recherche 80, Institute National de Recherche en Informatique et en Automatique (INRIA), 1981.

[BG92] S. Brookes and S. Geva. Stable and sequential functions on scott domains. Technical Report CMU-CS-92-121, School of Computer Science, Carnegie Mellon University, 1992.

[Bla92] A. Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–220, 1992.

[Buc93] A. Bucciarelli. *Sequential Models of PCF: some contributions to the domain-theoretic approach to full abstraction.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1993.

[CCF94] R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111:297–401, 1994.

[CF92] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction (preliminary version). In *Proc. 19th ACM Symp. Principles of Programming Languages*, pages 328–342. ACM Press, 1992.

[Chu40] A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[Cro93] R. L. Crole. *Categories for Types.* Cambridge University Press, 1993.

[Cur92a] P.-L. Curien. Observable algorithms on concrete data structures. In *Proc. 7th IEEE Symp. Logic in Computer Science*, pages 432–443. IEEE Computer Society Press, 1992.

[Cur92b] P.-L. Curien. Sequentiality and full abstraction. In M. P. Fourman *et al*, editor, *Applications of Categories in Computer Science*, pages 66–94. Cambridge University Press, 1992.

[Cur93] P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming*. Birkhäuser, second edition, 1993. Progress in Theoretical Computer Science Series.

[Ehr93] T. Ehrhard. Hypercoherences: a strongly stable model of linear logic. *Mathematical Structures for Computer Science*, 3:365–385, 1993.

[Ehr94] T. Ehrhard. Projecting sequential algorithms on strongly stable functions. *Journal of Pure and Applied Logic*, 1994. To appear.

[Fel86] W. Felscher. Dialogues as a foundation for intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. III*, pages 341–372. D. Reidel Publishing Company, 1986.

[Gan67] R. O. Gandy. Computable functionals of finite type I. In J. N. Crossley, editor, *Sets, Models and Recursion Theory*. North-Holland, 1967.

[Gan93] R. O. Gandy. Dialogues, blass games, sequentiality for objects of finite type. unpublished manuscript, 1993.

[Gir72] J.-Y. Girard. Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur. Thèse de Doctorat d'Etat, Paris, 1972.

[Göd58] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, pages 280–287, 1958.

[Göd90] K. Gödel. *Kurt Gödel Collected Works, Volumes I and II*, S. Feferman, *et al* (editors). Oxford Univ. Press, 1990.

[Gun92] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. MIT Press, 1992.

[HO93] J. M. E. Hyland and C.-H. L. Ong. Fair games and full completeness for Multiplicative Linear Logic without the mix-rule. ftp-able at `theory.doc.ic.ac.uk` in directory `papers/Ong`, 1993.

[HO95] J. M. E. Hyland and C.-H. L. Ong. Pi-calculus, dialogue games and PCF. In *Proceedings of 7th* ACM *Conference on Functional Programming Languages and Computer Architecture, La Jolla, California, June 26-28, 1995*, 1995. To appear.

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

[How89] D. J. Howe. Equality in lazy computation systems. In *Proc. 4th IEEE Symp. Logic in Computer Science*, pages 198–203. Computer Society Press, 1989.

[Hyl76] J. M. E. Hyland. A syntactic characterization of the equality in some models of the lambda calculus. *J. London Math. Soc. (2)*, 12:361–370, 1976.

[JS93] A. Jung and A. Stoughton. Studying the fully abstract model of PCF within its continuous function model. In *Proc. Int. Conf. Typed Lambda Calculi and Applications, Utrecht, March, 1993*, pages 230–245. Springer, 1993. LNCS Vol. 664.

[Kel82] G. M. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge Univ. Press, 1982. L. M. S. Lecture Notes Series 64.

[Kle59] S. C. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. American Mathematical Society*, 91:1–52, 1959.

[Kle63] S. C. Kleene. Recursive functionals and quantifiers of finite types II. *Trans. American Mathematical Society*, 108:106–142, 1963.

[Kle78] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited I. In J. E. Fenstad, R. O. Gandy, and G. E. Sacks, editors, *General Recursion Theory II, Proceedings of the 1977 Oslo Symposium*, pages 185–222. North-Holland, 1978.

[Kle80] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited II. In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, pages 1–29. North-Holland, 1980.

[KP93] G. Kahn and G. D. Plotkin. Concrete domains. *Thereotical Computer Science*, 1993. in Böhm Feschrift Special Issue. Article first appeared (in French) as technical report 338 of INRIA-LABORIA, 1978.

[Lam93] F. Lamarche. Stable domains are generalized topological spaces. *Theoretical Computer Science*, 111:103–123, 1993.

[LS86] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Mathematics No. 7. Cambridge University Press, 1986.

[MC88] A. R. Meyer and S. C. Cosmadakis. Semantical paradigms: notes for an invited lecture. In *Proc. 3rd Annual IEEE Symp. Logic in Computer Science*. Computer Society Press, 1988.

[Mil75] R. Milner. Processes, a mathematical model of computing agents. In *Logic Colloquium, Bristol 1973*, pages 157–174. North Holland, Amsterdam, 1975.

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

[Mil91] R. Milner. Polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, LFCS, University of Edinburgh, 1991. Also in *Logic and Algebra of Specification*, edited by F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer-Verlag, 1993.

[Mor68] J.-H. Morris. *Lambda Calculus Models of Programming Languages.* PhD thesis, M.I.T., 1968.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100:1–77, 1992.

[Ong92] C.-H. L. Ong. The concurrent lambda calculus 1: A general pre-congruence theorem for applicative bisimulation. Unpublished Manuscript, 27pp., 1992.

[Ong95] C.-H. L. Ong. Correspondence between operational and denotational semantics. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol 4*. Oxford Univ. Press, 1995. To appear.

[OR94] P. W. O'Hearn and J. G. Riecke. Kripke logical relations and PCF. To appear, 1994.

[Pla66] R. A. Platek. *Foundations of Recursion Theory.* PhD thesis, Standford University, 1966.

[Plo75] G. D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[Plo77] G. D. Plotkin. LCF as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[PW94] G. D. Plotkin and G. Winskel. Bistructures, bidomains and linear logic. In *Proceedings of International Colloquium on Automata, Programming and Languages 1994*, 1994. LNCS Vol. ??

[Rey83] J. C. Reynolds. Types, abstraction and parametric polymorphism. In *Information Processing 1983*, pages 513–523, 1983.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill Book Company, 1967.

[Saz75] V. Yu. Sazonov. Sequentiality and parallely computable functionals. In *Proc. Symp. Lambda Calculus and Computer Science Theory.* Springer, 1975. LNCS Vol. 37.

[Saz76] V. Yu. Sazonov. Functional computable in series and in parallel (english translation). *Matematicheskii Zhurnal*, 17:648 – 672, 1976.

[Saz76] V. Yu. Sazonov. Expressibility of functions in Scott's LCF language (translated from Russian). *Algebra i Logika*, 15:308–330, 76.

[Sco93] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM and OWHY. *Theoretical Computer Science*, 121:411–440, 1993. In Böhm Festschrift, a special issue of the Journal. The article has been widely circulated as an unpublished manuscript since 1969.

[Spe62] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In *Recursive Function Theory,* Proc. Symposia in Pure Mathematics V, pages 1–27. American Mathematical Society, Providence, RI, 1962.

[Sto88] A. Stoughton. *Fully Abstract Models of Programming Languages.* Pitman, 1988. Research Notes in Theoretical Computer Science.

[Str67] C. Strachey. Fundamental concepts in programming languages. Lecture notes for the International Summer School in Computer Programming, Copenhagen, 1967.

[Tai67] W. W. Tait. Intensional interpretation of functionals of finite type i. *J. Symb. Logic*, 32:198–212, 1967.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Mathematics*, 5:285–309, 1955.

[Wad76] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the lambda calculus. *SIAM J. Computing*, 5:488–521, 1976.

[Wad78] C. P. Wadsworth. Approximate reduction and lambda calculus models. *SIAM J. Computing*, 7:337–356, 1978.

[Win80] G. Winskel. *Events in Computation.* PhD thesis, University of Edinburgh, 1980.

[Win87] G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Application and Relationships to Other Models of Concurrency*, pages 325–392. Springer Verlag, 1987. LNCS 255.

[Win94] G. Winskel. Stable bistructure models of PCF. preprint, 1994.

Index