Realizability Toposes and Language Semantics

John R. Longley

Doctor of Philosophy
University of Edinburgh
1994

Abstract

Realizability toposes are "models of constructive set theory" based on abstract notions of computability. They arose originally in the study of mathematical logic, but since then various realizability toposes (particularly the *effective topos*) have found their way into several areas of computer science. This thesis investigates the general theory of realizability toposes, and their application to the semantics and logic of some simple programming languages.

In the earlier chapters we study the "pure theory" of realizability toposes. Each realizability topos is constructed from a *partial combinatory algebra* (PCA), which may be regarded as providing a notion of untyped computation. We introduce a new notion of morphism between PCAs, and show that these exactly correspond to certain functors between the toposes. Using this we are able to establish some previously unknown inequivalences between realizability toposes.

Next we develop some "domain theory" in realizability toposes. The search for a theory that works well for a wide class of models leads us to identify a new category of predomains, the *well-complete objects*, whose properties we study.

Finally we consider several versions of the programming language PCF and their semantics. We show how these languages may be adequately interpreted in realizability toposes, and prove a variety of universality and full abstraction results for particular languages with respect to particular models. We also obtain some more model-independent results asserting the "equivalence" between the call-by-name, call-by-value and lazy variants of PCF. We end with a discussion of how our models give rise to simple and intuitive "program logics" for these languages.

Acknowledgements

The first person to thank here is my supervisor, Mike Fourman. His diversity of intellectual interests and breadth of understanding have been a great source of inspiration to me, and I have enjoyed many wide-ranging discussions with him. He also introduced me to the area of research which has absorbed my interest for the last three years. My second supervisor, Gordon Plotkin, was also very helpful and supportive, and I have always found his incisive mental energy a great stimulation.

I have had useful and interesting discussions with many people, including Samson Abramsky, Marcelo Fiore, Martin Hofmann, Zhaohui Luo, James McKinna, Luke Ong, Andy Pitts, John Power, Thomas Streicher and Paul Taylor (whose diagram macros were a great help in the production of this thesis). But there remain three people whom I would like to thank individually. During his year in Edinburgh Wesley Phoa taught me a great deal in lectures and discussions that I would have found it very difficult to learn otherwise. Martin Hyland has been a great inspiration to me for many years now, and his support and involvement in my work have meant a lot to me. And Alex Simpson has spent literally hundreds of hours patiently discussing with me the ideas in this thesis at all stages in their development, and often in minute technical detail. His enthusiasm for my work frequently surpassed my own, and his sustained interest in my progress did a lot to keep me going.

Before coming to Edinburgh I spent a year working in the Formal Methods group at Roke Manor Research Ltd. (formerly Plessey Research and Technology). It was there that my interest in computer science was awakened, and in retrospect I feel that the time I spent there influenced my approach to the subject quite deeply.

I have made many friends through the computer science department in Edinburgh. I would particularly like to thank Benjamin Pierce for many enjoyable hours of musical collaboration, and Savi Maharaj, Peter Sewell and Alex Simpson for their company on hillwalking expeditions, as well as endless discussions about the Munros (and their recursive analogues).

My flatmates and other close friends over the last three years have helped me a great deal by their sympathy and companionship, and have provided distractions from work when necessary. A big thankyou to Steven Bradley, Louise Stephenson (now Bradley), Andrew Wilson, Stephen Fulton, Stephen Hughes, Catharine Sharman and especially Julie Mathieson.

Lastly, I would like to thank all my friends at St. Catherine's–Argyle Church, too numerous to mention, for their love and support. They have enriched my life immeasurably since I came to Edinburgh, and their concern was particularly appreciated at an emotionally difficult time during the final stages of writing this thesis.

Declaration

I hereby declare that this thesis was composed by myself, and the work described in it is my own except where otherwise stated.

John Longley

How precious to me are your thoughts, O God!
    How vast is the sum of them!
Were I to count them,
    they would outnumber the grains of sand.

Psalm 139:17–18.

Table of Contents

Chapter 0

Introduction

This thesis can be seen as a contribution to the study of *denotational semantics* for programming languages. Denotational semantics attempts to provide mathematically rigorous descriptions of programming languages by giving “interpretations” of programs in terms of previously understood mathematical notions. For instance, if P is a program that takes an input and produces an output, we might take its interpretation to be some kind of function from the set of possible input-values to the set of output-values. One of the aims of denotational semantics is to provide precise formal definitions of programming languages, in contrast to informal language definitions which often contain ambiguities. Another aim is to deepen our understanding of the *logic* of programming languages: one can use mathematical reasoning about the interpretations of programs to prove facts about the programs themselves. A very long-term goal is to provide a mathematical foundation for ways of reasoning about programs that could be used by programmers.

Different approaches to denotational semantics have made use of different mathematical structures in giving interpretations of programs, and the kinds of mathematics involved have been extremely varied. The most widely-known approach is that of traditional *domain theory*: here the mathematical structures involved are usually sets equipped with a partial ordering (or more than one!) satisfying certain conditions. This approach was initiated by Scott and Strachey in [91]; since then it has developed into a large and fruitful research field (see e.g. [35]). But other very different sorts of mathematics have also been used—as a random selection of examples we mention concrete data structures [11], functor categories [66], and games [3,43].

In this thesis we focus on one particular approach to denotational semantics, involving mathematical structures known as *realizability toposes*. These originally arose in the study of mathematical logic, but have recently started to find applications in computer science. As settings for denotational semantics they possess some interesting and distinctive features: for instance, they provide a kind of "domain theory" in which every function between domains is "computable" in some sense. We will explore both the pure-mathematical theory of realizability toposes and their applications to the semantics and logic of programming languages. The languages we consider will be very simple in comparison with "real" programming languages, but they will serve to illustrate certain theoretical issues.

The study of realizability toposes can involve some quite sophisticated mathematics, and so we should make an effort to explain why this is worth the trouble. In this introductory chapter we will try to provide some orientation and motivation for the reader, and indicate broadly what this thesis achieves. In Section 0.1 we outline the history of the ideas that led to the discovery of realizability toposes, and discuss some of the reasons why they are interesting from the point of view of computer science. We also explain the general motivations behind the thesis. In Section 0.2 we outline more specifically the contents of the thesis and give a high-level overview of each chapter. Throughout this introduction we will presuppose some familiarity with basic ideas from logic, recursion theory and category theory.

0.1 Background and motivation

0.1.1 History of realizability toposes

We first give a historical account of the ideas behind realizability toposes. It seems possible (in retrospect) to see them as representing the convergence of two distinct strands of thought, which we discuss separately.

Logic and realizability

The first strand is the study of *realizability interpretations* for constructive logic. This started with Kleene in [48], who gave such an interpretation for first-order Heyting arithmetic and also coined the term "realizability". Kleene's idea was to define a relation "n *realizes* φ" between natural numbers and logical sentences, intended to express the idea that n provides computational "evidence" for the constructive content of the formula φ:

- n realizes $t = t'$ iff $t = t'$ holds;
- n realizes $\varphi \wedge \psi$ iff $\mathit{fst}(n)$ realizes φ and $\mathit{snd}(n)$ realizes ψ;
- n realizes $\varphi \vee \psi$ iff either $\mathit{fst}(n) = 0$ and $\mathit{snd}(n)$ realizes φ or $\mathit{fst}(n) = 1$ and $\mathit{snd}(n)$ realizes ψ;
- n realizes $\varphi \Rightarrow \psi$ iff whenever m realizes φ, $\{n\}(m)$ exists and realizes ψ;
- n realizes $\neg\varphi$ iff nothing realizes φ;
- n realizes $\exists x.\varphi$ iff $\mathit{snd}(n)$ realizes $\varphi[\mathit{fst}(n)/x]$;
- n realizes $\forall x.\varphi$ iff for every m, $\{n\}(m)$ exists and realizes $\varphi[m/x]$.

(Here $\{n\}(m)$ denotes the result (if defined) of applying the nth partial recursive function to m, and fst, snd are the projections corresponding to some recursive and surjective "pairing" function $\mathrm{N} \times \mathrm{N} \to \mathrm{N}$.) This definition of realizability can be seen as a way of giving an interpretation of constructive arithmetic within *classical* mathematics. Kleene showed that all provable sentences of Heyting arithmetic were realizable; however, there are many realizable sentences that are *not* provable in Heyting arithmetic, and even some that are classically false. The realizability interpretation can thus be used to show that certain counter-classical assertions are consistent with Heyting arithmetic.

Many other notions of realizability were subsequently introduced—for instance, the "modified realizability" of Kreisel [51] and the "function realizability" of Kleene-Vesley [50]—though Kleene's original notion remained the "canonical" one.

The motivation for all these notions seems to have been mainly metamathematical: various notions of realizability could be used to provide interpretations of constructive logic showing the consistency of various intuitionistic principles. (For instance, the motivation for function realizability was to provide a classically understood model for second-order logic in which the "bar induction" principle was valid.) An extensive survey of applications of this nature is contained in [99].

One of the original motivations for the construction of realizability toposes was (or might have been) to show how these realizability interpretations could be extended to higher-order intuitionistic logic. A *topos* is a category that can be seen as a model for impredicative higher-order logic: types in the logic denote objects of the topos, and terms of the logic (including predicates) are interpreted by morphisms. In 1978, Hyland constructed the best-known example of a realizability topos, the *effective topos*, showing how Kleene's original interpretation of first-order arithmetic could be generalized to higher-order logic. (A similar construction was carried out by Powell at around the same time, yielding a "set-theoretic" rather than a categorical model.) Later Hyland, Johnstone and Pitts showed that a similar construction could be performed for a large class of other realizability interpretations, using an abstract framework known as *tripos theory* [76,41].

In fact, all the examples of realizability toposes explicitly considered in [41] were based on so-called *standard* realizability—a simple generalization of Kleene's original notion in which the realizing objects need not be natural numbers but may be elements of any *partial combinatory algebra* (cf. Section 1.1). The standard realizability toposes are certainly a natural class of models to consider, and we will concentrate almost exclusively on them in this thesis. Other realizability toposes have also been studied by other people, notably van Oosten [68], who has investigated them mainly from a logical and metamathematical angle.

"Datatypes" and effective computations

The second strand of thought that we wish to mention emphasizes *computation* rather than logic, and it is this strand that is closer to the concerns of this thesis. We may sketch the idea briefly as follows.

We would like a general theory of "computable functions" in the context of programming languages. Of course the theory of recursive functions gives us a good understanding of computations over the natural numbers, but what do we mean by a "computable function" over an arbitrary datatype? In order to address this question it is clearly not sufficient to understand the datatype simply as the set of its values, since this would give no way of distinguishing computable functions from non-computable ones.

The key observation is that the elements of any datatype are represented in a computer by more "primitive" objects (e.g. sequences of binary digits) for which we already have a good notion of computation, and "computations" on elements of the datatype actually consist of computations on these underlying representations. This suggests that we should understand a datatype as a set of values together with a set of "machine representations" for each value. More or less equivalently, we could understand it as an equivalence relation on some set of "machine values", two machine values being deemed equivalent iff they represent the same value of the datatype.

In an abstract mathematical theory it seems reasonable to take our "machine representations" to be natural numbers rather than actual sequences of bits, and to use the notion of computation given by basic recursion theory. We are thus led to view datatypes as *partial equivalence relations* (PERs) on the natural numbers, and computable functions as *morphisms* between PERs. The category of PERs can ultimately be traced back to Kreisel [51]; it also appeared in Girard's thesis [32], where it was used to give an interpretation of the datatypes of System F. The idea of using natural numbers to represent elements of datatypes seems more explicit in the work of Ershov [22], who considers in effect the category of *total* equivalence relations on the natural numbers.

In fact we can be slightly more general: instead of taking natural numbers as our machine values we could take elements of any partial combinatory algebra, e.g. a term model for the untyped λ-calculus. From a computer science point of view this would not be completely unrealistic: indeed implementations of functional languages are often based on reduction in the untyped λ-calculus.

We should now explain what all this has to do with realizability toposes. Hyland showed in [38] that the effective topos contains a full subcategory equivalent to the category of PERs on N (the situation for other standard realizability toposes is similar). This subcategory will play a very important role in this thesis, where it will be called the category of *modest sets*. The definition of modest set that we will give in Section 1.2 closely follows the above idea of a datatype as a set of values with machine representations for each value. A slightly larger subcategory is the category of *assemblies*—the only difference between these and modest sets is that in assemblies we allow the same machine value to represent more than one value of the datatype.

Subsequently, Freyd *et al* [15] showed that rather than constructing the realizability topos and then "discovering" these subcategories, one can take the category of assemblies as primary and then obtain the topos as its "completion" in a suitable abstract sense. This is the construction that we will use in this thesis (see Section 1.3)—for our purposes it seems preferable to the "logical" construction via tripos theory since it lays greater emphasis on the computational aspects of realizability toposes.

0.1.2 Realizability toposes in computer science

We have seen that notions of "computation" are built into the nature of realizability toposes. We now turn to examine the reasons why these toposes are interesting from the point of view of computer science itself, and some of the applications they have found there. Research in this field seems to fall roughly into two distinct (but overlapping) areas.

Modelling polymorphism

The first area of application has been in modelling *polymorphic* languages. Polymorphic programs embody algorithms that can be applied to values of many different datatypes: for example, the algorithm for reversing a list of values of type t is the same no matter what t is, and so this algorithm can be represented by a polymorphic program. In a typed programming language, such a polymorphic program will have a *polymorphic type*. Polymorphic types thus often involve an (explicit or implicit) quantification over all datatypes of the language, and in fact this is a serious obstacle to modelling them using classical set theory (see [82]). However, it turns out that polymorphic types *can* be interpreted "set-theoretically" using realizability toposes. (We will not go into the reason why this is so here.)

We mentioned earlier that Girard had used the category of PERs to model the second-order polymorphic λ-calculus (System F) in [32]. In 1985 Moggi and Hyland discovered that the subcategory of modest sets in the effective topos was closed under arbitrary products, at least from the point of view of the topos itself (see [39]). This was the technical breakthrough that made it possible to give explicitly "set-theoretic" interpretations of System F (see e.g. [77,56]). Later, Hyland *et al* [44] showed that the subcategory of modest sets was actually "complete" in an even stronger sense, thereby allowing set-theoretic interpretations of more powerful polymorphic systems such as the Calculus of Constructions (e.g. [19,95]). In a slightly different vein, Hyland and Ong [42] showed that strong normalization results for many polymorphic type theories could be proved using toposes based on *modified* realizability.

These "type-theoretic" applications of realizability toposes have been very successful, but we will not touch on them much in this thesis.

Synthetic domain theory

The concerns of this thesis are much closer to the second area of application: the use of realizability toposes as a source of "categories of domains" for denotational semantics. This idea goes back to Scott, who in a talk given in 1979 suggested that one should try to find "domains" that could be regarded as just sets in some constructive "universe" (e.g. a topos). Domain-theoretic constructions would then be reduced to familiar "set-theoretic" ones, and in particular we would be able to exploit the higher-order logic of the topos to reason about domains. The study of such categories of domains inside toposes became known as *synthetic domain theory* (the use of the adjective "synthetic" here stemmed from an analogy with synthetic differential geometry). From the beginning, the motivation seems to have been the hope that the "domains as sets" idea would yield a simpler and cleaner form of domain theory than the classical one.

It later turned out that realizability toposes did indeed provide examples of constructive universes in which suitable domain-like objects could be found. This accorded well with the idea that many of the objects in realizability toposes could be understood as "datatypes". In [60,59] McCarty showed that (essentially) the classical category of effective Scott domains arose as a full subcategory of the effective topos. The Ph.D. theses of Rosolini [85] and Phoa [71] studied particular categories of domains in the effective topos (and other models) whose definitions were completely "internal" (i.e. given entirely in the higher-order logic of the topos). Hyland [40] and Taylor [97] have subsequently proposed more "axiomatic" approaches to synthetic domain theory, with domains in the effective topos as the motivating example.

Other people have also studied categories of domains in realizability models but from a more "external" point of view, in the traditional spirit of classical mathematics (see e.g. [26,1,5]). A few people have actually used some of these domains for denotational semantics, e.g. Amadio [6] and Phoa [75], who has used realizability models to give an interesting proof of the adequacy of translations of PCF into the untyped λ-calculus.

The overall impression gained from the literature to date is that many realizability toposes do contain categories of domains with a great deal of interesting structure, rich enough to model strong polymorphism, recursive types, subtypes and non-determinism, for instance. However, much of the work cited above depends on some fairly sophisticated mathematical machinery, and synthetic domain theory sometimes appears to outsiders as obscure and impenetrable. So despite many interesting developments, it seems only fair to admit that the original aim of "simplicity" has not yet been achieved. Also it seems that the techniques used to study different realizability toposes, or different categories of domains within the same topos, have been rather *ad hoc* and have not exhibited much of a general pattern, in spite of the axiomatic work of Hyland and Taylor.

0.1.3 Motivation for this thesis

Having set the historical scene, we now explain the general motivation (both practical and philosophical) behind the work in this thesis.

In [25] Fourman and Phoa proposed that synthetic domain theory could be used to give a "set-theoretic" semantics for the functional fragment of the ML programming language. The idea was that this would provide a formal justification for a "naive" way of reasoning about programs, and could give a semantic foundation for program logics in the spirit of Extended ML [47]. As originally conceived, this thesis was to have been an attempt to work out the details of such a semantics for a fragment of ML in the effective topos, and hence to obtain a sound program logic for this fragment.

However, in pursuing this direction I became increasingly aware that the effective topos was probably *not* the best model to use, and that other realizability toposes deserved further attention. It was therefore natural to ask: what are the important similarities and differences between realizability toposes when it comes to doing denotational semantics? Most work in synthetic domain theory had concentrated on the effective topos, and even though some people had considered other realizability models (e.g. in [71,5]), there had been very little investigation of the *general* theory of realizability toposes. It was from a consideration of these more theoretical issues that the present thesis emerged.

It is worth recounting the particular observation that first sparked my interest in these questions. This concerned a difference between the effective topos and the realizability topos based on a term model for the untyped λ-calculus. Both these toposes are constructed from notions of "computability", and one might have thought that these notions were somehow "equivalent" in view of the fact that in each case the definable partial functions from N to N are precisely the partial recursive ones. (Indeed Freyd conjectured, perhaps for this reason, that the two toposes were actually equivalent, though this was refuted in [46].) However, although the two toposes indeed contain the same functions $\mathrm{N} \to \mathrm{N}_\perp$, it turns out that they give different classes of functions at more complex types: for instance, the effective topos contains a morphism corresponding to the "parallel-or" function, but the topos based on the λ-calculus does not (see [72]). Thus the two models appear to embody different "notions of computability". To understand the difference, It is instructive to think about the nature of "computations" in the two partial combinatory algebras. In the PCA consisting of the natural numbers with Kleene's application, a computation may perform any arithmetical operation whatsoever on its argument, and different computations can be performed "in parallel" by interleaving them. In the term model for the λ-calculus, however, there is no way in which a computation can "look inside" a λ-term and examine its syntax—the only way in which a computation can use its argument is as a "black box" (e.g. it can apply it to other arguments).

This fact came as a surprise to me, and I was intrigued by the nature of the difference between the two PCAs. It suggested some connection with the work of Plotkin [79] and Sazonov [86] on "sequential" and "parallel" versions of the programming language PCF. From a practical point of view, it seemed likely that the λ-term model would yield a better (e.g. more fully abstract) model than the effective topos for a sequential language such as ML.

The starting-point of this thesis, then, is the informal idea that a PCA represents a "notion of untyped computation". Of course, there are PCAs that are not intuitively "implementable", and so in these cases the notion of computation will not be a very realistic one! Nevertheless, it is possible to view any PCA as giving an *abstract* notion of computability: we can regard the PCA as *defining* what is meant by "computation", and proceed from there. In terms of our earlier remarks on modest sets and assemblies, we can think of a PCA as an "abstract machine" on top of which we can implement various interesting "datatypes" (as modest sets). The PCAs discussed above provide a good example of two very different "flavours" of computation. In this thesis we will develop some general theory that is common to all (or several) PCAs, and also investigate some of the differences between particular PCAs (and the corresponding realizability toposes). We will also study how our models relate to various versions of the language PCF.

Besides considering PCAs as abstract machines in isolation from one another, it is also natural to ask what are the relationships between different PCAs. We mentioned earlier that many functional languages are (or can be) implemented in terms of reduction algorithms for the untyped λ-calculus; but these algorithms are themselves implemented in terms of sequences of binary digits—thus our high-level language can be seen as implemented on either of two abstract machines, corresponding to two "levels of abstraction". Moreover, the fact that one of these abstract machines is implemented on the other suggests a way of relating the two PCAs. In the earlier chapters of this thesis we will introduce a notion of *applicative morphism* between PCAs that makes this idea precise. We will only study applicative morphisms from a pure-mathematical point of view here, but the definition was actually inspired by this metaphor from computer science.

0.2 Overview of this thesis

Having discussed the general motivation, we now describe more specifically what is contained in this thesis.

In the thesis we study the general theory of *standard* realizability toposes (and certain subcategories of them), and their application to the semantics and logic of *simply-typed* functional programming languages. The overall structure of the thesis exhibits a gradual progression from theory to practice. In Chapters 1–3 we study the abstract theory of realizability toposes and the relationships between them. This material can be viewed as pure mathematics, and there is hardly any discussion of computer science applications in these chapters. In Chapters 4–5 we start to move towards computer science, and develop some "domain theory" in realizability toposes. In Chapters 6–7 we use this theory to give denotational semantics for several versions of PCF. Finally in Chapter 8 we discuss how our models give rise to logics for reasoning about programs in these languages.

Prerequisites

First, a few words on the prerequisites for reading the thesis. I have tried to keep these to a minimum; in particular I have consciously avoided reference to sophisticated ideas from "advanced" category theory as far as possible. No familiarity with fibrations, internal categories, general topos theory or the internal logic of toposes is presupposed. However, it *has* been necessary to assume a reasonable degree of fluency in "basic" category theory: categories, functors, natural transformations, limits and colimits, adjunctions, and cartesian-closed categories (the first five chapters of [57] should suffice). A vague acquaintance with monads and with enriched categories is required in one or two places.

The only other formal prerequisites are a knowledge of basic recursion theory (e.g. the first seven chapters of [17]) and some knowledge of the untyped λ-calculus (e.g. Chapter 2 of [8]). In a few parts of the thesis (Sections 3.2, 5.3, 7.4) we use some rather more technical facts about the λ-calculus (e.g. the theory of Böhm trees), but the reader unfamiliar with these will not be at much of a disadvantage except in these sections.

Some knowledge of classical domain theory and denotational semantics would also be extremely helpful. For background in this area, we strongly recommend Plotkin's classic paper "LCF considered as a programming language" [79].

Synopsis

We now give a chapter-by-chapter summary of the contents of the thesis and point out some of the important landmarks.

The first three chapters are "pure theory". In **Chapter 1** we give a brief introduction to the general theory of partial combinatory algebras. For an arbitrary

PCA A we then construct the category $\mathbf{Ass}(A)$ of *assemblies* and the category $\mathbf{Mod}(A)$ of *modest sets* over A, and review some of their standard properties. Next we give Freyd's construction of the realizability topos $\mathbf{RT}(A)$ as a "completion" of $\mathbf{Ass}(A)$. We also obtain a new result saying that $\mathbf{Ass}(A)$ can itself be seen as the completion of $\mathbf{Mod}(A)$ in a suitable sense (Theorem 1.4.5).

In **Chapter 2** we introduce the notion of *applicative morphism* between PCAs (Definition 2.1.1), inspired by the idea of "implementing" one abstract machine in terms of another. Mathematically, the main interest in these lies in the fact that applicative morphisms $A \to B$ correspond precisely to certain functors $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$ (Theorem 2.2.20). Similar results hold for **RT** and **Mod**. A pleasing consequence of these results is a precise characterization of when two realizability toposes are equivalent (Corollary 2.3.8). We also investigate how certain properties of applicative morphisms correspond to certain properties of the functors they give rise to.

Chapter 3 is devoted to applications of the theory developed in Chapter 2. It turns out that applicative morphisms arise from a wide variety of situations in mathematics and computer science. Besides giving a selection of examples, we use our theory to prove two new *inequivalence* results. One of these states that term models for the call-by-name and call-by-value λ-calculi give rise to inequivalent realizability toposes (Theorem 3.2.18). The other states that the topos arising from Scott's *graph model* $\mathcal{P}\omega_{re}$ is not equivalent to one arising from a λ-term model. The proof of this latter result is especially interesting computationally, as it relies on the idea that λ-term models embody a "sequential" notion of computation while $\mathcal{P}\omega_{re}$ embodies a "parallel" one.

In **Chapter 4** we start to build the mathematical machinery we need in order to do denotational semantics. Since the languages we wish to model allow partial functions to be defined, we need a theory of partial maps in our categories—this is provided by Rosolini's theory of *dominances* (see [85]). We review this theory and discuss dominances within realizability models. We also introduce the notion of a *divergence* on a PCA (Definition 4.3.1). A divergence is essentially a subset of the PCA consisting of the elements (if any) that we wish to regard as representing non-terminating computations. We discover that every divergence gives rise to a dominance, and that several examples of dominances that have previously been considered can be obtained via the notion of divergence.

We also need the mathematical machinery to interpret *recursion* in our programming languages, so we would like a theory of *fixed points*. We develop such a theory in **Chapter 5**, working in the general setting of an arbitrary realizability model with dominance. Much of the material here is joint work with Alex Simpson—specific details of the attribution are given *in loco*. We show that in our setting one simple "Completeness Axiom" suffices to ensure that the model contains a category of "predomains" with very good closure properties—we call this the category of *well-complete* objects. We also show that several natural examples of models do indeed satisfy the Completeness Axiom, and hence that our approach offers a unified treatment of various models previously studied separately. In fact, our work here seems to be the first treatment of synthetic domain theory that unifies "sequential" and "parallel" models of computation.

In **Chapter 6** we use our models for some denotational semantics. We consider the sequential language PCF, as well as extensions PCF^{+} and PCF^{++} obtained by adding a "parallel-or" operator and an "exists" functional. Moreover, we consider *call-by-name*, *call-by-value* and *lazy* variants of each of these languages. We show that any realizability model equipped with a dominance satisfying the Completeness Axiom admits adequate interpretations of all three variants of sequential PCF, and consider the circumstances under which these extend to interpretations of the "parallel" languages. In fact, in this chapter we will not work concretely with realizability models, but with an abstract categorical notion of "PCF-model". There are many examples of PCF-models other than realizability models, and it is just as easy to prove our results at this greater level of generality.

We also obtain some general results concerning the related properties of full abstraction and universality. An interpretation of a programming language is called *fully abstract* if observationally equivalent terms of the language are always assigned the same denotation in the model; it is called *universal* if every element of the model (of an appropriate type) is denotable by some term of the language. We show that the call-by-name, call-by-value and lazy variants of each of our languages are "equivalent" in some sense, at least from the point of view of questions of full abstraction and universality. We obtain these results by means of syntactic translations between the variants. These results may be seen as a slight digression from the main line of the thesis in that they are not specific to realizability models; however, we apply them to realizability models in Chapter 7.

Chapter 7 can be seen as the climax of the thesis. Here we investigate more fully the idea that some PCAs embody "sequential" notions of computation while others embody "parallel" ones. We focus on three particular PCAs: the model K_1 consisting of the natural numbers with Kleene application; the Scott graph model $\mathcal{P}\omega_{re}$; and a term model $\Lambda^0/\mathcal{B}$ for the untyped λ-calculus. We show that the classical category of *effective Scott domains* can be regarded as a full subcategory of either $\mathbf{Mod}(K_1)$ or $\mathbf{Mod}(\mathcal{P}\omega_{re})$ (the former result is due essentially to McCarty), and hence that both these categories provide universal and fully abstract models of all three variants of PCF^{++}. We also discover a sense in which $\mathbf{Mod}(\mathcal{P}\omega_{re})$ is mathematically the nicer of the two models.

The main open problem raised by this thesis is the conjecture that the model $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ provides a universal (and hence fully abstract) model for sequential PCF. Although we have been unable to prove this conjecture, we obtain some partial results, for instance the fact that this model is fully abstract at type 3 (at least for call-by-value PCF). We discuss the evidence for the conjecture and some of our attempts at proving it.

In **Chapter 8** we consider logics for reasoning about programs. We examine two different ways in which our models give rise to program logics. The first of these involves a very simple-minded classical logic, for which we can give a (classical) set-theoretic interpretation using our models. The second involves a constructive logic, which we can interpret essentially as the internal logic of a realizability topos. In both cases we can give an alternative interpretation of the logic in terms of purely "operational" concepts intelligible to programmers. We investigate the problem of finding models in which the denotational interpretation of the logic coincides with the operational one—this property has been called *logical full abstraction* by Plotkin.

We conclude in **Chapter 9** with some philosophical discussion of our results, and some suggestions for further research.

Interdependence of chapters

The approximate interdependence of the chapters in this thesis is given by the partial order generated by $1 \preceq 2 \preceq 3, \quad 1 \preceq 4 \preceq 5 \preceq 6 \preceq 7 \preceq 8$. There are a few points at which this dependence relation is violated, but in these cases we give an explicit backward reference.

As mentioned earlier, the material in Chapters 2 and 3 is of mainly pure-mathematical interest and at present has no direct application in computer science. For readers whose main interest is in computer science, we therefore recommend the following short cuts. Sections 1.1 and 1.2 should be read in detail, but Section 1.3 could be skimmed, since in practice the topos $\mathbf{RT}(A)$ plays a much smaller role than $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$ in later chapters. Section 1.4 could be omitted completely. We would then suggest skipping the whole of the next two chapters and proceeding straight to Chapter 4. However, the reader unfamiliar with the construction of the Scott graph models $\mathcal{P}\omega$ and $\mathcal{P}\omega_{re}$ will need to refer to the beginning of Section 3.3, since these PCAs reappear in Section 5.3 and play major roles in Chapters 7 and 8.

In principle Chapter 6 is logically almost entirely independent of the rest of the thesis, and may contain material of interest to workers in other areas of semantics. It interfaces with the rest of the thesis via Paragraph 5.4.2, where the abstract categorical notion of "PCF-model" is introduced. It would thus be possible to read just Paragraph 5.4.2, ignoring all references to realizability models, and then the whole of Chapter 6. (Of course, we hope that by then the reader would want to understand the applications of this material in Chapter 7!)

Notation

The set-theoretic and categorical notation used in this thesis is mostly fairly standard. We draw a few points to the reader's attention here.

First some set-theoretic notation: If X and Y are sets, we write $f : X \rightharpoonup Y$ to mean that f is a partial function from X to Y. If f is a partial or total function, we write $\mathrm{Dom}\, f$, $\mathrm{Im}\, f$ to denote the set-theoretic *domain* and *image* of f respectively. We also write $r : X \twoheadrightarrow\!\!\!\!\!\!\!+\;\; Y$ to mean that r is a *total relation* from X to Y, i.e. $r \subseteq X \times Y$ satisfies $\forall x \in X.\, \exists y \in Y.\, r(x, y)$. We write $\mathcal{P}X$ to denote the powerset of X; $\mathcal{P}_1 X$ for the set of non-empty subsets of X; and $\mathcal{P}_{fin} X$ for the set of finite subsets of X. We take $(-, -)$ to be some standard pairing operation, and write $X \times Y$ for the set $\{(x, y) \mid x \in X, y \in Y\}$. We write $X + Y$ for the set $\{(0, x) \mid x \in X\} \cup \{(1, y) \mid y \in Y\}$. We also write $X \sqcup Y$ for the union of X and Y where these are implicitly assumed to be disjoint. We write N for the set of natural numbers (including 0); we sometimes write ω in place of N where this seems to suit the context better.

Now some categorical conventions: If f is a morphism in some category, we may write $\mathrm{dom}\, f$ and $\mathrm{cod}\, f$ for its domain and codomain in the categorical sense. Given two morphisms $f : A \to B$ and $g : B \to C$, we write their composition as either gf or $g \circ f$, without much consistency. We write id or id_A for the identity morphism on an object A, reserving the symbol 1 for the terminal object of a category. Given morphisms $f : C \to A$ and $g : C \to B$, we write $\langle f, g \rangle$ for the induced morphism $C \to A \times B$. Likewise, given $f : A \to C$ and $g : B \to C$ we write $[f, g]$ for the induced morphism $A + B \to C$. In general we use N (as distinct from N) to denote the natural number object in a category.

We sometimes abbreviate a finite list of variables $x_1, \ldots, x_n$ by a *vector* $\vec{x}$. If M and N are terms in some syntax, we write $M[N/x]$ for the result of substituting N for all occurrences of x in M (renaming bound variables if necessary).

Chapter 1

PCAs and realizability toposes

In this chapter we introduce the categories that we will study in this thesis, and summarize the facts about assemblies, modest sets and realizability toposes that we will need. We presuppose some elementary category theory and recursion theory, but no knowledge of topos theory is assumed.

The material in Sections 1–3 is not original. In Section 1.1 we give a brief review of the theory of partial combinatory algebras (PCAs), the structures from which the various categories of interest are built. In Section 1.2 we introduce two categories that can be constructed from a PCA A: the category $\mathbf{Ass}(A)$ of *assemblies* and the category $\mathbf{Mod}(A)$ of *modest sets*. In Section 1.3 we obtain the realizability topos $\mathbf{RT}(A)$ as the *exact completion* of $\mathbf{Ass}(A)$. Throughout these sections we draw freely on results from the literature, and give proofs only when these contribute extra information that we require later on.

In Section 1.4 we give a characterization of $\mathbf{Ass}(A)$ via a universal property. This result appears to be new; however, the proof is rather technical and we only use the result once in the rest of the thesis. The reader would therefore be well advised to skip the whole of Paragraph 1.4.2 on a first reading.

There are many ways in which the material in this chapter could be presented. We have given particular prominence to $\mathbf{Ass}(A)$ and $\mathbf{Mod}(A)$, since it is these categories that are most important from the point of view of programming language semantics. In order to keep our account as elementary as possible, we have also avoided using the sophisticated conceptual apparatus of categorical logic. Inevitably this has meant giving a rather one-sided picture. For readers interested in understanding the logical aspects of these categories, we warmly recommend the paper [41] as a supplement to our account.

1.1 Partial combinatory algebras

We begin with an introduction to the theory of partial combinatory algebras (PCAs). Most of this material is covered in Chapter VI of [10]; detailed accounts of the theory of *total* combinatory algebras may also be found in [7,8,36].

PCAs can be regarded loosely as "abstract machines", within which certain "computations" can be performed. It is this computational aspect of PCAs that we emphasize here, since this is what gives rise to the computational flavour of the categories that we will consider. We begin with some definitions and notation.

1.1.1 Definitions and examples

Definition 1.1.1 *A* partial applicative structure $(A, \cdot)$ *is just a set* A *equipped with a partial binary operation* $(x, y) \mapsto x \cdot y$*.*

We think of the operation $\cdot$ as a kind of "function application"; note, however, that we are in an untyped setting where functions can be applied to other functions and even to themselves. We normally denote application by left-associative juxtaposition, so that we write e.g. $xy(zx)$ in place of $(x \cdot y) \cdot (z \cdot x)$.

If $a, b \in A$, note that the expression ab need not denote an element of A; hence we need to be careful to distinguish elements of A from expressions over A. We can formalize the notion of expression as follows.

Definition 1.1.2 *Assume we have an infinite supply of* variables $x_0, x_1, \ldots$*, and for each* $a \in A$ *we have a* constant c_a*. The set* $\mathcal{E}(A)$ *of* formal expressions *over*

A is freely constructed from the variables and constants using juxtaposition:

$$x_i \in \mathcal{E}(A), \qquad c_a \in \mathcal{E}(A), \qquad e, e' \in \mathcal{E}(A) \;\Rightarrow\; (ee') \in \mathcal{E}(A).$$

When writing expressions we normally write just a in place of c_a, and omit unnecessary parentheses. We use $e, e', \ldots$ as meta-variables ranging over $\mathcal{E}(A)$. If e, e' are expressions and x is a variable, then $e[e'/x]$ means the expression obtained by substituting e' for x in e. We call a formal expression *closed* if it contains no variables. Note that a closed expression $e \in \mathcal{E}(A)$ may or may not denote an element of A; we write $e\!\downarrow$ to mean "e is defined" (i.e. e denotes an element of A), and $e\!\uparrow$ to mean "e is undefined". Note that if $ee'\!\downarrow$ then both $e\!\downarrow$ and $e'\!\downarrow$. If e, e' are closed, we write $e = e'$ to mean "e, e' are both defined and denote the same value" (strict equality), and $e \simeq e'$ to mean "if either e or e' is defined then so is the other and they denote the same value" (Kleene equality). More generally, if e, e' are expressions whose variables are among $x_1, \ldots, x_n$, we write

$$\begin{array}{lll} e\!\downarrow & \text{to mean} & e[\vec{a}/\vec{x}]\!\downarrow \;\text{ for all } a_1, \ldots, a_n \in A; \\ e \simeq e' & \text{to mean} & e[\vec{a}/\vec{x}] \simeq e'[\vec{a}/\vec{x}] \;\text{ for all } a_1, \ldots, a_n \in A \end{array}$$

and so on. Using this notation we can now give our main definition.

Definition 1.1.3 (PCA) A partial combinatory algebra (PCA) *is an applicative structure* $(A, \cdot)$ *such that there exist elements* $k, s \in A$ *satisfying*

$$kxy = x, \qquad sxyz \simeq xz(yz), \qquad sxy\!\downarrow$$

(where x, y, z are variables). A total combinatory algebra *is just a PCA in which application is total (i.e. $ab\!\downarrow$ for all $a, b \in A$).*

Remarks 1.1.4 (i) By abuse of language we normally say "A is a PCA", where the application operation is understood.

(ii) Note that in this definition k, s need not be unique, and the choice of k, s is not part of the data for a PCA. Nevertheless, we often implicitly assume that PCAs do come equipped with a chosen k and s, and so use these symbols (or sometimes k_A, s_A) without apology as constants denoting elements of A.

(iii) It is traditional in the definition of PCA to require also that $k \neq s$; however, for our purposes it seems more natural not to insist this. It is easy to show that $k = s$ iff A contains just one element. We write $\mathbf{1}$ to denote the *trivial* (one-element) PCA.

(iv) We will write i to denote the element skk. Note that i represents the identity function: $ix = x$ for all $x \in A$.

We now give a couple of examples of PCAs for motivation. Many more examples will be introduced in Chapter 3.

Examples 1.1.5 (i) Consider the set N of natural numbers equipped with *Kleene application*: $m \cdot n \simeq \{m\}(n)$, where $\{m\}$ denotes the partial recursive function coded by m. The existence of k, s with the required properties is an immediate consequence of the S-m-n theorem (see e.g. [17, Chapter 4]). We refer to this PCA as K_1, or *Kleene's first model*. Technically, the definition of K_1 is dependent on a particular encoding of the partial recursive functions as natural numbers; however, for most purposes the choice of encoding will be irrelevant.

(ii) Let Λ denote the set of terms of the untyped lambda-calculus (over a countably infinite set of variables). Let Λ/β be its quotient modulo β-equality (see e.g. [8, Chapter 4]), with the inherent application. This defines a total combinatory algebra: for k, s we may take (the equivalence classes of) the lambda-terms $\mathbf{k} \equiv \lambda xy.x$, $\mathbf{s} \equiv \lambda xyz.xz(yz)$.

1.1.2 Combinatory completeness

The definition of PCA may seem somewhat mysterious. It is motivated by the fact that the axioms for a PCA imply the following crucial property, which says that all "definable functions on A" are representable by an element of A itself.

Proposition 1.1.6 (Combinatory completeness) *Let A be a PCA. Then for any $e \in \mathcal{E}(A)$ there is a formal expression $\lambda^* x.e \in \mathcal{E}(A)$, whose variables are just those of e excluding x, such that $(\lambda^* x.e)\downarrow$ and $(\lambda^* x.e)a \simeq e[a/x]$ for all $a \in A$.*

Proof We define $\lambda^* x.e$ by induction on the structure of e: $\lambda^* x.x$ is i; $\lambda^* x.y$ is ky if y is a constant or variable other than x; and $\lambda^* x.uv$ is $s(\lambda^* x.u)(\lambda^* x.v)$. $\Box$

Remarks 1.1.7 (i) It is easy to see that any partial applicative structure with the combinatory completeness property is in fact a PCA.

(ii) If $e \in \mathcal{E}(A)$, the notation $\lambda^* x.e$ is to be seen as meta-syntactic sugar for some expression e' involving k and s. A meta-expression such as $\lambda^* x.\lambda^* y.e$ can be translated to a formal expression by first translating $\lambda^* y.e$ to e', then translating $\lambda^* x.e'$ to e''. We abbreviate $\lambda^* x.\lambda^* y.e$ to $\lambda^* xy.e$.

(iii) For any $e \in \mathcal{E}(A)$, we have $\lambda^* x.e\downarrow$ whether or not $e\downarrow$. However, note that we may have $e \simeq e'$ but not $\lambda^* x.e \simeq \lambda^* x.e'$ (consider $e \equiv s$, $e' \equiv kss$).

(iv) The following is easy to prove by structural induction on e: if $e'\downarrow$ and x does not appear in e', then $(\lambda^* x.e)e' \simeq e[e'/x]$.

(v) Note that in the PCA Λ/β the meanings of $\lambda x.e$, $\lambda^* x.e$ coincide. However, we retain the distinction in notation since the lambda-term given by the definition of $\lambda^* x.e$ is in general not syntactically identical to $\lambda x.e$.

We emphasize that the λ^*-notation is meta-notation and is not part of the syntax of formal expressions. Consequently there are pitfalls associated with the use of λ^*: for instance, it is *not* true in general that $(\lambda^* xy.x)e \simeq \lambda^* y.e$ for every closed expression e, even if A is total (consider $e \equiv ss$). Since we would prefer to use λ^*-notation where possible rather than long expressions involving k and s, it is worth taking some trouble to establish what instances of the "β-rule" are valid. (Other expositions seem to gloss over this rather delicate issue.)

Let $\mathcal{M}(A)$ be the set of *formal meta-expressions* over A, defined inductively as follows:

$$x_i \in \mathcal{M}(A), \qquad c_a \in \mathcal{M}(A), \qquad M, M' \in \mathcal{M}(A) \;\Rightarrow\; (MM') \in \mathcal{M}(A),$$
$$M \in \mathcal{M}(A) \;\Rightarrow\; (\lambda^* x_i.M) \in \mathcal{M}(A).$$

(Here M and M' are meta-variables ranging over $\mathcal{M}(A)$.) We define the notions of free and bound variable in the obvious way, and call a meta-expression *closed* if it contains no free variables. We also write $M[M'/x]$ to mean the meta-expression obtained by substituting M' for x in M (renaming bound variables to avoid clashes). Clearly every meta-expression "evaluates" to a unique expression (which in turn might denote an element of A); we write $M \twoheadrightarrow e$ to mean "M evaluates to e". If $M \twoheadrightarrow e$ and $M' \twoheadrightarrow e'$, we can write $M\downarrow$ and $M \simeq M'$ to mean $e\downarrow$ and $e \simeq e'$ respectively. It is worth observing that $\lambda^* x.M\downarrow$ for all $M \in \mathcal{M}(A)$.

Unfortunately, substitution at the levels of expressions and meta-expressions do not always agree: for example, we have $(\lambda^* y.x) \twoheadrightarrow kx$ and $ss \twoheadrightarrow ss$ but not $(\lambda^* y.ss) \twoheadrightarrow k(ss)$. However, the two meanings of substitution do agree in certain instances:

Lemma 1.1.8 *Suppose $M, M' \in \mathcal{M}(A)$, $e, e' \in \mathcal{E}(A)$, x, y are variables and a is a constant or variable. Then*

(i) if $\lambda^* y.e \twoheadrightarrow e'$ then $\lambda^* y.(e[a/x]) \twoheadrightarrow e'[a/x]$;

(ii) if $M \twoheadrightarrow e$ then $M[a/x] \twoheadrightarrow e[a/x]$;

(iii) if $M \twoheadrightarrow e$, $M' \twoheadrightarrow e'$ and no free occurrence of x in M occurs under a λ^*-abstraction, then $M[M'/x] \twoheadrightarrow e[e'/x]$.

Proof (i) Easy by induction on the structure of e. The cases $e \equiv x$, $e \equiv y$ and e some other variable need to be treated separately.

(ii) By induction on the structure of M. The cases $M \equiv x$, M a constant or variable other than x and $M \equiv M_1 M_2$ are easy. For the case $M \equiv \lambda^* y.M'$, note that if $M' \twoheadrightarrow e'$ then $\lambda^* y.e' \twoheadrightarrow e$. Hence by (i) we have $\lambda^* y.(e'[a/x]) \twoheadrightarrow e[a/x]$, and by the induction hypothesis we have $M'[a/x] \twoheadrightarrow e'[a/x]$. Hence $M[a/x] \equiv \lambda^* y.(M'[a/x]) \twoheadrightarrow e[a/x]$.

(iii) Easy by induction on the structure of M. Again the case $M \equiv x$ has to be treated separately. $\square$

Theorem 1.1.9 (Valid β-rules) *Suppose $M, M' \in \mathcal{M}(A)$, x is a variable and a is a constant or variable. Then*

(β1) $(\lambda^* x.M)a \;\simeq\; M[a/x]$;

(β2) if $M'\downarrow$, $x \notin \mathrm{FV}(M')$ and no free occurrence of x in M occurs under a λ^*-abstraction, then $(\lambda^* x.M)M' \;\simeq\; M[M'/x]$.

PROOF (i) First note that if $M \twoheadrightarrow e$ then regarding e as a meta-expression we have $M \simeq e$. So by Proposition 1.1.6 and part (ii) of the above lemma we have

$$(\lambda^* x.M)a \;\simeq\; (\lambda^* x.e)a \;\simeq\; e[a/x] \;\simeq\; M[a/x]$$

(note that $e[a/x]$ has the same meaning whether e is regarded as an expression or a meta-expression).

(ii) Suppose $M \twoheadrightarrow e$ and $M' \twoheadrightarrow e'$. Then similarly by Remark 1.1.7(iv) and part (iii) of the above lemma we have

$$(\lambda^* x.M)M' \;\simeq\; (\lambda^* x.e)e' \;\simeq\; e[e'/x] \;\simeq\; M[M'/x]. \qquad \square$$

We will use these two rules incessantly when performing calculations with combinators (i.e. elements of a PCA defined by meta-expressions). Note that $M \simeq M'$ implies $MN \simeq M'N$ and $NM \simeq NM'$, and so we can legally perform β-reductions of the above kinds on certain subterms of meta-expressions. However, we are not entitled to perform β-reductions underneath a λ^*-abstraction, since $M \simeq M'$ does not imply $\lambda^* x.M \simeq \lambda^* x.M'$ (cf. Remark 1.1.7(iii)).

Although we will continue to make use of formal meta-expressions, henceforth we will never need to refer explicitly to the sets $\mathcal{M}(A), \mathcal{E}(A)$ or the relation $\twoheadrightarrow$.

1.1.3 Some important combinators

The relatively simple axioms for a PCA yield a surprising amount of "computational" structure. For instance, it turns out that all the partial recursive functions can be simulated in any PCA (other than **1**). We now explore some of this structure in the setting of a general PCA.

The following proposition typically illustrates the use of the above β-rules in calculations involving combinators.

Proposition 1.1.10 (Pairing and booleans) *Let A be any PCA.*

(i) There exist elements pair, fst, snd $\in A$ such that

$$\mathit{pair}\; x\; y\!\downarrow, \qquad \mathit{fst}\,(\mathit{pair}\; x\; y) \;=\; x, \qquad \mathit{snd}\,(\mathit{pair}\; x\; y) \;=\; y$$

(ii) There exist true, false, if $\in A$ *such that*

$$\mathit{if}\ x\ y\downarrow, \qquad \mathit{if}\ \mathit{true}\ y\ z \;=\; y, \qquad \mathit{if}\ \mathit{false}\ y\ z \;=\; z$$

Proof (i) Take $\mathit{pair} = \lambda^* xyz.zxy$, $\mathit{fst} = \lambda^* v.v(\lambda^* xy.x)$, $\mathit{snd} = \lambda^* v.v(\lambda^* xy.y)$. First note that by rule $\beta 1$ we have $\mathit{pair}\ x\ y \simeq (\lambda^* yz.zxy)y \simeq \lambda^* z.zxy$, and so $\mathit{pair}\ x\ y\downarrow$. Next note that

$$\begin{array}{rcll} \mathit{fst}\,(\mathit{pair}\ x\ y) & \simeq & (\lambda^* v.v(\lambda^* xy.x))(\lambda^* z.zxy) & \\ & \simeq & (\lambda^* z.zxy)(\lambda^* xy.x) & (\text{by } \beta 2) \\ & \simeq & (\lambda^* xy.x)xy & (\text{by } \beta 2) \\ & \simeq & (\lambda^* y.x)y & (\text{by } \beta 1) \\ & \simeq & x & (\text{by } \beta 1). \end{array}$$

But clearly $x\downarrow$, so $\mathit{fst}\,(\mathit{pair}\ x\ y) = x$. Similarly we can show $\mathit{snd}\,(\mathit{pair}\ x\ y) = y$.

(ii) Take $\mathit{true} = \lambda^* yz.y$, $\mathit{false} = \lambda^* yz.z$, $\mathit{if} = \lambda^* xyz.xyz$. By a similar style of reasoning we can verify that these have the above properties. □

We will sometimes write pair_A, true_A etc. if A is not evident from the context. We will also write $\langle M, M'\rangle$ as syntactic sugar for $(\mathit{pair}\ M\ M')$.

We now show how the natural numbers can be represented in a PCA. Many different representations are possible, but the following is especially convenient:

Definition 1.1.11 (Curry numerals) *Let A be a PCA. For each $n \in \mathrm{N}$, we define an element $\overline{n} \in A$ as follows: $\overline{0} = i$, $\overline{n+1} = \langle \mathit{false}, \overline{n}\rangle$. We call $\overline{n}$ the* numeral *for n.*

Proposition 1.1.12 *There exist elements succ, pred, iszero* $\in A$ *such that*

$$\begin{array}{c} \mathit{succ}\ \overline{n} \;=\; \overline{n+1}, \qquad \mathit{pred}\ \overline{0} \;=\; \overline{0}, \qquad \mathit{pred}\ \overline{n+1} \;=\; \overline{n}, \\ \mathit{iszero}\ \overline{0} \;=\; \mathit{true}, \qquad \mathit{iszero}\ \overline{n+1} \;=\; \mathit{false} \end{array}$$

for all $n \in N$.

Proof Take $\mathit{succ} = \lambda^* x.\langle \mathit{false}, x\rangle$, $\mathit{iszero} = \mathit{fst}$, $\mathit{pred} = \lambda^* x.\mathit{if}(\mathit{iszero}\ x)\,\overline{0}\,(\mathit{snd}\ x)$. Using Proposition 1.1.10 it is routine to check that these combinators have the required properties. □

Next we will show how to define functions by *recursion* in PCAs. The following combinators are very powerful tools for performing recursive definitions.

Proposition 1.1.13 (Fixed point combinators) *There exist elements $Y, Z \in A$ such that*

$$Yf \simeq f(Yf), \qquad Zf{\downarrow}, \qquad (Zf)z \simeq f(Zf)z$$

Proof Let $W = \lambda^* xy.y(xxy)$, $X = \lambda^* xyz.y(xxy)z$ and take $Y = WW$, $Z = XX$. Then we have

$$\begin{aligned} Yf &\simeq (\lambda^* xy.y(xxy))Wf \\ &\simeq (\lambda^* y.y(WWy))f \qquad (\text{by } \beta 1) \\ &\simeq f(WWf) \qquad (\text{by } \beta 1) \\ &\simeq f(Yf) \end{aligned}$$

Similarly we can show that Z has the required properties. □

(Note in passing that in this proof we genuinely need the rule $\beta 1$, whereas in the previous proofs we in fact only needed $\beta 2$.)

One use for the combinator Z is to define a "primitive recursion" operator for the natural numbers. This gives us a convenient way of defining many functions on the numerals.

Proposition 1.1.14 *There exists an element $rec \in A$ such that*

$$rec\ x\ f\ \overline{0} = x, \qquad rec\ x\ f\ \overline{n+1} \simeq f\ \overline{n}\ (rec\ x\ f\ \overline{n}).$$

Proof Let $R = \lambda^* r. \lambda^* xfm.\ \mathit{if}(\mathit{iszero}\, m)\,(kx)\,(\lambda^* y.\ f\,(\mathit{pred}\, m)\,(r x f\,(\mathit{pred}\, m)\, i))$ and take $rec = \lambda^* xfm.(ZR)xfmi$. It is a good exercise in the use of β-rules to check that rec has the above two properties. □

In future we will often write down some combinator and claim that it satisfies some equations, leaving it to the conscientious reader to check that these can be proved using $\beta 1$ and $\beta 2$.

Notice that (in any PCA except **1**) $\overline{n_1} = \overline{n_2}$ implies $n_1 = n_2$ and so our PCA contains a faithful copy of the natural numbers. We mentioned above that other representations are possible: for instance, in the case $A = K_1$ we could simply take $\overline{n} = n$. In fact it is immaterial what representation we choose, providing there exist elements corresponding to *succ* and *rec*:

Proposition 1.1.15 *Suppose we have elements $\overline{0}', \overline{1}', \ldots \in A$, and there are elements $succ', rec' \in A$ satisfying*

$$succ'\ \overline{n}' = \overline{n+1}', \qquad rec'\ x\ f\ \overline{0}' = x, \qquad rec'\ x\ f\ \overline{n+1}' = f\ \overline{n}'\ (rec'\ x\ f\ \overline{n}').$$

Then there exist $c, d \in A$ such that $c\,\overline{n} = \overline{n}'$ and $d\,\overline{n}' = \overline{n}$ for all n.

Proof Take $c = rec\ \overline{0}'\ succ'$ and $d = rec'\ \overline{0}\ succ$. $\square$

Using the constant *rec*, it is easy to see how any primitive recursive function can be represented in a PCA. In fact, with a little more work, one can show how to represent any *partial recursive* function. This gives some support to the informal idea that "PCAs are models of untyped computation."

Theorem 1.1.16 *Let A be a PCA. Then any partial recursive $\varphi : \mathrm{N}^k \rightharpoonup \mathrm{N}$ is* (weakly) representable *in A, i.e. there is $e \in A$ such that for all $m_1, \ldots, m_k, n \in \mathrm{N}$ we have*

$$\varphi(m_1, \ldots, m_k) \;=\; n \quad \Longrightarrow \quad e\,\overline{m_1} \ldots \overline{m_k} \;=\; \overline{n}.$$

Proof See [10, Section VI.2.8]. $\square$

1.2 Assemblies and modest sets

Before constructing the realizability topos over a PCA, we need to discuss two other important categories: the category of assemblies and the category of modest sets. In this section we construct these categories and summarize some of their important properties. For a more thorough account of this material (worked out for the PCA K_1) we refer the reader to [39] or [74].

1.2.1 The categories Ass(A) and Mod(A)

Throughout this section A denotes a PCA. We first define the *category of assemblies on A*.

Definition 1.2.1 (Assemblies) *(i) An* assembly *X on A consists of a set $|X|$ together with a function assigning to each element $x \in |X|$ a non-empty subset $\|x\|$ of A. We call $|X|$ the* underlying set *of X; and if $a \in \|x\|$, we say a* realizes *(or* is a realizer for*) x. To avoid ambiguity we sometimes write $\|x\|_X$ or $\|x \in X\|$ for $\|x\|$.*

(ii) Suppose X, Y are two assemblies on A. A function $f : |X| \to |Y|$ is said to be tracked *by an element $r \in A$ if for all $x \in |X|$ and $a \in A$ we have*

$$a \in \|x\|_X \quad \Longrightarrow \quad ra \in \|fx\|_Y.$$

A morphism *$f : X \to Y$ is a function $f : |X| \to |Y|$ that is tracked by some $r \in A$. We write $\mathbf{Ass}(A)$ for the category of assemblies on A and morphisms between them.*

Remarks 1.2.2 (i) It is easy to see that assemblies and their morphisms do form a category: for any assembly X the identity function $\mathrm{id}_{|X|}$ is tracked by i; and if f,g are composable morphisms tracked by r, t respectively, then gf is tracked by $\lambda^* a.t(ra)$.

(ii) In the above definition, the membership symbol $\in$ is to be understood as a *strict* relation. Hence $ra \in \|fx\|_Y$ implies $ra\downarrow$.

(iii) The category $\mathbf{Ass}(K_1)$ is frequently called the category of *ω-sets* in the literature (see e.g. [56]).

One way of understanding the definition of $\mathbf{Ass}(A)$ intuitively is as follows. We may think of an assembly X as a "datatype", with $|X|$ as the set of "values" of the datatype and $\|x\|$ as the set of "machine-level representations" of the value x. A function f between datatypes maps values to values. If r tracks f, we may think of r as a "machine implementation" of f, since it operates on representations of

x to produce representations of $f(x)$. Thus, in $\mathbf{Ass}(A)$, the only morphisms that exist between datatypes are those that are "implementable".

Following this intuition, it is natural to consider the class of "good" datatypes for which a value is determined by any of its machine representations. This motivates the following definition:

Definition 1.2.3 (Modest sets) *An assembly X is called a* modest set *if for all $x, x' \in |X|$ we have*

$$x \neq x' \quad \Longrightarrow \quad \|x\| \cap \|x'\| = \emptyset$$

Let $\mathbf{Mod}(A)$ denote the full subcategory of $\mathbf{Ass}(A)$ consisting of modest sets, and let J denote the inclusion functor $\mathbf{Mod}(A) \hookrightarrow \mathbf{Ass}(A)$.

It is not difficult to see that $\mathbf{Mod}(A)$ is equivalent to the perhaps more familiar category of partial equivalence relations (PERs) on A.

We now review some categorical facts about $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$, and give explicit descriptions of some of the structure. Our proofs are somewhat sketchy—for more details see e.g. [74].

Proposition 1.2.4 *Both $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$ are cartesian-closed categories, and the inclusion J preserves the cartesian-closed structure.*

Proof In both $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$, the terminal object 1 is given by $|1| = \{*\}$, $\| * \| = A$. The product of X and Y is given by

$$\begin{array}{rcl} |X \times Y| & = & |X| \times |Y|, \\ \|(x, y) \in X \times Y\| & = & \{ \langle a, b\rangle \mid \ a \in \|x\|_X, \ b \in \|y\|_Y \} \end{array}$$

and the two projections are tracked by *fst*, *snd*. The exponential Y^X is given by

$$\begin{array}{rcl} |Y^X| & = & \{f : X \to Y \mid f \text{ is tracked by some } r \in A\}, \\ \|f \in Y^X\| & = & \{r \in A \mid r \text{ tracks } f\} \end{array}$$

and the evaluation function $X \times Y^X \to Y$ is tracked by $\lambda^* a.(\mathit{snd}\ a)(\mathit{fst}\ a)$. □

Proposition 1.2.5 *In both $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$, a morphism $f : X \to Y$ is mono iff $f : |X| \to |Y|$ is injective.*

PROOF Straightforward. □

Proposition 1.2.6 *Both* **Mod**(A), **Ass**(A) *have finite limits (and* J *preserves them).*

PROOF Finite products were discussed above, so we only need to consider equalizers. Given $f, g : X \rightrightarrows Y$, let Z be defined by

$$|Z| \;=\; \{x \in |X| \;\mid fx = gx\} \qquad \|x\|_Z \;=\; \|x\|_X.$$

Clearly Z is a modest set if X is. The inclusion function $e : |Z| \to |X|$ is tracked by i; and it is easy to verify that e is the equalizer of f, g. □

Proposition 1.2.7 *Both* **Mod**(A), **Ass**(A) *have finite sums (and* J *preserves them).*

PROOF The initial object 0 is defined by $|0| = \emptyset$. The sum of X and Y is given by

$$\begin{aligned} |X+Y| &= |X| + |Y|, \\ \|(0,x) \in X+Y\| &= \{\, \langle \mathit{true}, a\rangle \mid \; a \in \|x\|_X\}, \\ \|(1,y) \in X+Y\| &= \{\, \langle \mathit{false}, a\rangle \mid \; a \in \|x\|_X\}. \;\square \end{aligned}$$

Proposition 1.2.8 *Both* **Mod**(A), **Ass**(A) *have a natural number object (and* J *preserves it).*

PROOF Let N be the modest set given by

$$|N| \;=\; \mathrm{N}, \qquad \|n \in N\| \;=\; \{\overline{n}\}$$

and let $1 \xrightarrow{Z} N \xrightarrow{S} N$ be the morphisms tracked by $\lambda^* a.\overline{0}$, *succ* respectively. If X is an assembly and we have maps $1 \to X \to X$ tracked by r, t respectively, then the unique mediating morphism $N \to X$ is tracked by $\mathit{rec}\;(ri)(kt)$. So N is a natural number object in both categories. □

1.2.2 Regular categories

One important aspect of **Mod**(A) and **Ass**(A) is that they are both examples of *regular* categories—this is the aspect that we will concentrate on in the first three chapters of this thesis. Here we give only the definition of a regular category and a few very basic facts. For more background on regular categories and their significance in other areas of mathematics, see [29] or [9].

The following remarks may help to motivate the definition. Given $f : X \to Y$, recall that $h, k : Z \rightrightarrows X$ are a *kernel-pair* of f iff the diagram

$$\begin{tikzcd} Z \arrow[r, "k"] \arrow[d, "h"'] & X \arrow[d, "f"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

is a pullback. In this case we can think of the coequalizer of h, k as giving a categorical definition of the "image" of f. In **Set**, for instance, if $c : X \to W$ is the coequalizer of the kernel-pair of f, then c is the quotient of X by the equivalence relation induced by f; hence $W \cong \mathrm{Im}\, f$. We can view regular categories as categories in which good "images" exist:

Definition 1.2.9 (Regular category) *A category is* regular *if it has finite limits and coequalizers of kernel-pairs, and furthermore the pullback of a regular epi along any morphism is a regular epi.*

Recall that a regular epi is one that arises as a coequalizer. It is easy to check that (in any category with pullbacks) a morphism is a regular epi iff it is the coequalizer of its own kernel-pair. We may regard the "pullback" condition in the definition as saying that images are "well-behaved".

The next two propositions, together with Proposition 1.2.6, show that $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$ are regular categories.

Proposition 1.2.10 *Both* $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$ *have coequalizers of kernel-pairs (and* J *preserves them).*

Proof If $f : X \to Y$ is tracked by t, its kernel-pair $h, k : Z \rightrightarrows X$ is given by:

$$\begin{aligned} |Z| &= \{(x, x') \mid fx = fx'\}, \\ \|(x, x') \in Z\| &= \{\, \langle a, a' \rangle \mid \ a \in \|x\|_X,\ a' \in \|x'\|_X \}, \end{aligned}$$

h, k are the evident projections tracked by *fst*, *snd* respectively.

Now define the assembly W by

$$|W| = \mathrm{Im}\, f, \qquad \|y \in W\| = \bigcup_{fx=y} \|x \in X\|.$$

If X is a modest set then so is W. The evident function $c : X \to W$ is tracked by i, and it is easy to check that c is the coequalizer of h, k. □

Proposition 1.2.11 *Given a diagram in* $\mathbf{Ass}(A)$ *of the form*

$$\begin{tikzcd} Z' \arrow[r, shift left, "h'"] \arrow[r, shift right, "k'"'] \arrow[d] & X' \arrow[r, "c'"] \arrow[d] & W' \arrow[d, "g"'] \\ Z \arrow[r, shift left, "h"] \arrow[r, shift right, "k"'] & X \arrow[r, "c"] & W \end{tikzcd}$$

in which c is the coequalizer of its kernel-pair h, k, and the three evident squares are pullbacks. Then c' is a coequalizer of its kernel-pair h', k'.

Proof Without loss of generality we may suppose h, k, c are as in the proof of Proposition 1.2.10. Given $g : W' \to W$ we may obtain explicit descriptions of the pullbacks; it is then routine to verify that h', k' are a kernel-pair of c' and that c' is a coequalizer of h', k'. □

In diagrams we will frequently use arrows $\rightarrowtail$ for monos and $\twoheadrightarrow$ for regular epis. We mention here a few general facts about regular categories that will be useful later. The proofs are straightforward diagram-chases.

Proposition 1.2.12 *The following hold in any regular category.*

(i) Regular epis are closed under composition;

(ii) Every morphism f can be factored, uniquely up to isomorphism, as a regular epi followed by a mono. (We refer to this as the image factorization *of f);*

(iii) Given a commutative diagram

$$\begin{tikzcd} X \arrow[r, two heads, "e"] \arrow[d] & W \arrow[r, tail, "m"] & Y \arrow[d] \\ X' \arrow[r, two heads, "e'"] & W' \arrow[r, tail, "m'"] & Y' \end{tikzcd}$$

there is a unique fill-in morphism $W \to W'$ making both squares commute. $\square$

The relevant structure-preserving maps between regular categories are *exact* functors (sometimes called *regular* functors). A functor is exact iff it preserves finite limits and coequalizers of kernel-pairs (or equivalently, preserves finite limits and regular epis). For example, the above propositions tell us that the inclusion $J : \mathbf{Mod}(A) \hookrightarrow \mathbf{Ass}(A)$ is an exact functor. Exact functors play a very important role in what follows.

1.2.3 Uniform assemblies

We now turn our attention to a subcategory of $\mathbf{Ass}(A)$ that is equivalent to a much more familiar category: the category of sets!

Definition 1.2.13 *Let $\nabla : \mathbf{Set} \to \mathbf{Ass}(A)$ be the functor defined as follows:*

- $|\nabla X| = X$ *and* $\|x \in \nabla X\| = A$ *for all* $x \in X$.
- *If* $f : X \to Y$ *then* $\nabla f = f$ *(this is tracked for instance by* $i \in A$*).*

We sometimes write ∇_A for ∇ to avoid ambiguity.

It is easy to see that ∇ is a full, faithful and exact functor and preserves exponentials (note that it does *not* preserve finite sums or the natural number object).

We call an assembly X *uniform* if its elements are "uniformly realized" by some $a \in A$, i.e. we have $a \in \|x\|$ for all $x \in |X|$. It is simple to show that X is uniform iff it is isomorphic to an assembly in the image of ∇; hence the full subcategory of uniform assemblies is equivalent to $\mathbf{Set}$. Uniform objects and modest sets in some sense represent opposite extremes in the class of assemblies—in fact, the only objects in both subcategories are (those isomorphic to) the initial and terminal objects 0 and 1. Note also that if X is uniform and Y is modest then every morphism $X \to Y$ is constant.

There is another important functor that we need to introduce:

Definition 1.2.14 *Let* $\Gamma : \mathbf{Ass}(A) \to \mathbf{Set}$ *be the functor defined as follows:*

- $\Gamma X = |X|$.
- *If* $f : X \to Y$ *then* $\Gamma f = f$.

We sometimes write Γ_A *for* Γ *to avoid ambiguity.*

It is easy to see that Γ is naturally isomorphic to the "global elements" functor $\mathrm{Hom}(1, -)$. It is also clear that Γ is faithful, exact and preserves finite sums and the natural number object (however it does not preserve exponentials). The crucial fact about Γ is the following:

Proposition 1.2.15 *The functor* $\Gamma : \mathbf{Ass}(A) \to \mathbf{Set}$ *is left adjoint to* ∇, *and* $\Gamma\nabla = \mathrm{id}$.

PROOF The fact that $\Gamma\nabla = \mathrm{id}$ is obvious. For the unit of the adjunction, let $\eta_X : X \to \nabla\Gamma X$ be the morphism determined by $\mathrm{id}_{|X|}$ and tracked by i. Given any morphism $f : X \to \nabla Y$, clearly f is the unique set-theoretic function $f' : |X| \to Y$ such that $f = (\nabla f') \circ \eta_X$. (Note that $\eta : \mathrm{id} \to \nabla\Gamma$ is a natural transformation and that each component η_X is mono.) $\Box$

Remarks 1.2.16 (i) Alternatively we could define Γ to be the functor $\mathrm{Hom}(1, -)$; this is of course a more "categorical" definition. However, the definition we have given is technically more convenient, since we have $\Gamma\nabla = \mathrm{id}$ "on the nose".

(ii) In what follows we will write $\eta : \mathrm{id} \to \nabla\Gamma$ to denote the unit of the adjunction. However, we will also frequently abuse notation and write $\eta_X : X \to Y$ where Y is some object canonically isomorphic to $\nabla\Gamma X$.

1.3 Realizability toposes

We now embark on the construction of the realizability topos $\mathbf{RT}(A)$. Many different (equivalent) constructions have been given (see e.g. [41,15,84])—our approach is based on that in [15], where $\mathbf{RT}(A)$ is defined as the exact completion of $\mathbf{Ass}(A)$. Actually, the fact that $\mathbf{RT}(A)$ *is* a topos will not really be exploited in this thesis, except perhaps in Section 8.2. In this section we concentrate mostly on some of the more basic categorical properties of $\mathbf{RT}(A)$.

1.3.1 The exact completion of a regular category

We begin by recalling some definitions concerning *relations* in a regular category. All the concepts here are straightforward abstractions of familiar concepts in the category of sets. In any regular category, a relation from X to Y is represented simply by a monomorphism $\rho : R \rightarrowtail X \times Y$. For example, for any object X we always have the *diagonal* relation $\delta_X : X \rightarrowtail X \times X$. We say that ρ, ρ' represent the *same* relation iff they are isomorphic as subobjects of $X \times Y$ (usually we will blur the distinction between representatives and their equivalence classes). If ρ, σ are two relations from X to Y, we write $\rho \subseteq \rho'$ if $\rho = \rho'\theta$ for some θ. Given $\rho : R \rightarrowtail X \times Y$, we write ρ° for the corresponding *reciprocal* relation $R \rightarrowtail Y \times X$. Finally, given relations $\rho : R \rightarrowtail X \times Y$, $\sigma : S \rightarrowtail Y \times Z$, we may define their *relational composition* $\sigma.\rho$ as follows: take the pullback

$$\begin{tikzcd} P \arrow[r, "k"] \arrow[d, "h"'] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, "\pi\sigma"] \\ R \arrow[r, "\pi'\rho"] & Y \end{tikzcd}$$

and let $\sigma.\rho : T \rightarrowtail X \times Z$ be the mono appearing in the image factorization of the morphism $\langle \pi\rho h, \pi'\sigma k \rangle : P \to X \times Z$.

We are now in a position to give a categorical definition of an equivalence relation. Again this is a simple generalization of the set-theoretic notion:

Definition 1.3.1 *In any category with pullbacks, an* equivalence relation *on an object X is a relation $\rho : R \rightarrowtail X \times X$ that is*

$$\begin{array}{ll} \textit{reflexive:} & \delta_X \subseteq \rho, \\ \textit{symmetric:} & \rho^\circ \subseteq \rho, \\ \textit{transitive:} & \rho.\rho \subseteq \rho. \end{array}$$

It is easy to check that if $h, k : Z \rightrightarrows X$ are a kernel-pair for $f : X \to Y$ then $\langle h, k\rangle : Z \to X \times X$ is an equivalence relation. If c is the coequalizer of h and k, we may think of c as the "quotient" of X by this equivalence relation. We will now consider categories in which good quotients exist for all equivalence relations.

Definition 1.3.2 *A regular category is called* exact *if for every equivalence relation $R \xrightarrow{r} X \times X$, the pair of morphisms $\pi r, \pi' r : R \rightrightarrows X$ arises as a kernel-pair (and hence has a coequalizer).*

Note that in [29] exact categories are called "effective regular categories".

It can easily be seen that $\mathbf{Ass}(A)$ is *not* an exact category. The idea now is to turn it into one by formally "adding" good quotients of equivalence relations. We do this by means of a general construction that works for any regular category.

Definition 1.3.3 (Exact completion) *Given a regular category $\mathcal{C}$, let $\mathbf{EC}(\mathcal{C})$ be the category defined as follows:*

- objects *are pairs (X, ρ), where X is an object of $\mathcal{C}$ and ρ is an equivalence relation on X;*
- morphisms $(X, \rho) \to (Y, \sigma)$ *are relations $\tau : T \rightarrowtail X \times Y$ that are*

$$\begin{array}{ll} \textit{extensional:} & \sigma.\tau.\rho \subseteq \tau, \\ \textit{total:} & \rho \subseteq \tau^\circ.\tau, \\ \textit{single-valued:} & \tau.\tau^\circ \subseteq \sigma; \end{array}$$

- *the identity on (X, ρ) is the relation ρ from X to X;*
- *composition of morphisms is given by relational composition.*

Remark 1.3.4 The intuition behind the definition of morphism in $\mathbf{EC}(\mathcal{C})$ is as follows. Recall that, in the category of sets, a relation $\tau : T \rightarrowtail X \times Y$ is the graph of a function iff it is single-valued and total. But τ is single-valued precisely when $\tau.\tau^\circ \subseteq \delta_Y$, and total precisely when $\delta_X \subseteq \tau^\circ.\tau$. The conditions in the above definition are obtained by replacing the equality relation δ by a general equivalence relation.

It is easy to check that $\mathbf{EC}(\mathcal{C})$ is indeed a category, and that there is a faithful functor $I : \mathcal{C} \to \mathbf{EC}(\mathcal{C})$ defined by

$$I(X) = (X, \delta_X); \qquad I(f : X \to Y) = \langle \mathrm{id}_X, f \rangle : X \rightarrowtail X \times Y.$$

The reader can now verify the following directly, or consult [29, §2.16]:

Proposition 1.3.5 $\mathbf{EC}(\mathcal{C})$ *is an exact category, and* I *is an exact functor.* $\square$

Note that if $\rho : R \rightarrowtail X \times X$ is an equivalence relation in $\mathcal{C}$, then the two morphisms $I(\pi\rho), I(\pi'\rho) : IR \rightrightarrows IX$ have as their coequalizer the evident morphism $(X, \delta_X) \xrightarrow{\rho} (X, \rho)$. This justifies the remark that $\mathbf{EC}(\mathcal{C})$ is obtained by "adding quotients" to $\mathcal{C}$.

The next theorem says that these quotients have in fact been added in a "universal" way. Here $\mathbf{Ex}[\mathcal{D}, \mathcal{E}]$ denotes the category of exact functors $\mathcal{D} \to \mathcal{E}$ and all natural transformations between them.

Theorem 1.3.6 *For any exact category* $\mathcal{E}$*, the functor* $(- \circ I) : \mathbf{Ex}[\mathbf{EC}(\mathcal{C}), \mathcal{E}] \to \mathbf{Ex}[\mathcal{C}, \mathcal{E}]$ *is part of an equivalence of categories. In particular, for any exact functor* $F : \mathcal{C} \to \mathcal{E}$*, there is an exact functor* $\tilde{F} : \mathbf{EC}(\mathcal{C}) \to \mathcal{E}$*, unique up to isomorphism, such that the triangle*

$$\begin{tikzcd} \mathcal{C} \arrow[d, "I"'] \arrow[dr, "F"] & \\ \mathbf{EC}(\mathcal{C}) \arrow[r, "\tilde{F}"] & \mathcal{E} \end{tikzcd}$$

commutes up to natural isomorphism.

Proof See [29, §2.169]. $\square$

Note that $\mathbf{Ex}(\mathcal{C})$ is determined uniquely up to equivalence by this universal property.

1.3.2 The topos RT(A)

We now apply the above construction to the regular category $\mathbf{Ass}(A)$. The surprise is that the category we obtain is not merely exact but in fact a *topos* (that is, a category with finite limits, exponentials and a subobject classifier).

Definition 1.3.7 *The category* $\mathbf{EC}(\mathbf{Ass}(A))$ *is called the* (standard) realizability topos *on* A*, and is denoted by* $\mathbf{RT}(A)$*. We call* $\mathbf{RT}(K_1)$ *the* effective topos*, and denote it also by* $\mathcal{E}\!f\!f$*.*

The proof that $\mathbf{RT}(A)$ is a topos is non-trivial—see e.g. [29, §2.4]. (In fact the proof there is given just for K_1, but it is easily adapted to work for a general PCA.) For the reader wanting to learn about topos theory, a good recent text is [58]; other references include [45,33,9,52,29]. For most of this thesis, however, the following facts about $\mathbf{RT}(A)$ will suffice:

Theorem 1.3.8 $\mathbf{RT}(A)$ *has exponentials, finite sums and a natural number object. Moreover, the functor* $I : \mathbf{Ass}(A) \to \mathbf{RT}(A)$ *is full and preserves all this structure.*

Proof See e.g. [38]. $\Box$

There is one more piece of information about $\mathbf{RT}(A)$ that we need. Recall from Section 1.2.3 that $\nabla : \mathbf{Set} \to \mathbf{Ass}(A)$ is right adjoint to $\mathrm{Hom}(1, -)$. This situation extends to the whole of $\mathbf{RT}(A)$ as follows:

Proposition 1.3.9 *Let* $\Gamma' = \mathrm{Hom}(1, -) : \mathbf{RT}(A) \to \mathbf{Set}$ *and* $\nabla' = I\nabla$*. Then* Γ', ∇' *are exact,* $\Gamma \cong \Gamma' I$*,* $\Gamma'\nabla' \cong \mathrm{id}$ *and* $\Gamma' \dashv \nabla'$*.*

Proof Clearly ∇' is exact since ∇ and I are exact, and it is easy to check that Γ' preserves finite limits. To see that Γ' preserves regular epis, we use the fact that for any object $Z \not\cong 0$ of $\mathbf{RT}(A)$ there exists a morphism $1 \to Z$. Given $f : X \to Y$ regular epi and $y \in \mathrm{Hom}(1, Y)$, let $w : Z \to X$ be the pullback of Y along f. Then $Z \to 1$ is regular epi, so in particular $Z \not\cong 0$. Take any morphism $z : 1 \to Z$ and let $x = wz : 1 \to X$; then $fx = y$. Hence $\mathrm{Hom}(1, f)$ is surjective, i.e. a regular epi in **Set**.

Clearly $\Gamma \cong \Gamma' I$ since $\Gamma \cong \mathrm{Hom}(1, -)$ and I is full. Hence $\Gamma'\nabla' \cong \Gamma' I \nabla \cong \Gamma\nabla \cong \mathrm{id}$. Also $\nabla'\Gamma' I \cong I\nabla\Gamma$, so by Theorem 1.3.6 the natural transformation $I\eta : I \to I\nabla\Gamma$ extends to a natural transformation $\eta' : \mathrm{id} \to \nabla'\Gamma'$. The triangle laws for the adjunction $\Gamma' \dashv \nabla'$ are easy to verify using Theorem 1.3.6. $\square$

Our final proposition summarizes the most important facts from the last two sections.

Proposition 1.3.10 *We have a diagram of regular categories and exact functors:*

$$\begin{tikzcd} \mathbf{Mod}(A) \arrow[r, hook] & \mathbf{Ass}(A) \arrow[r, "I"] \arrow[d, shift right, "\Gamma"'] & \mathbf{RT}(A) \\ & \mathbf{Set} \arrow[u, shift right, "\nabla"'] & \end{tikzcd}$$

in which

- *all the categories are cartesian-closed and have finite sums and a NNO;*
- *all the functors are faithful;*
- *all the functors except Γ are full and preserve exponentials;*
- *all the functors except ∇ preserve finite sums and the NNO;*
- $\Gamma \cong \mathrm{Hom}(1, -)$, $\Gamma \dashv \nabla$ *and* $\Gamma\nabla \cong \mathrm{id}$*;*
- $\mathbf{RT}(A)$ *is a topos and is the exact completion of* $\mathbf{Ass}(A)$*.* $\square$

1.4 A universal property of Ass(A)

In the last section we gave an explicit description of $\mathbf{RT}(A)$, and saw how it could be characterized by a universal property involving $\mathbf{Ass}(A)$. In Section 1.2 we gave explicit descriptions of $\mathbf{Mod}(A)$ and $\mathbf{Ass}(A)$, and so it is natural to ask: Can $\mathbf{Ass}(A)$ be characterized by a universal property involving $\mathbf{Mod}(A)$? In this section we give a positive answer to this question. The material here seems closely related to that in [14], though we have not yet explored the connections.

The following remarks may help to explain why the above question is interesting. We can use $\mathbf{Mod}(A)$ to provide a model e.g. for first-order arithmetic corresponding to the (standard) realizability interpretation over A (see [99,100] for background). One of the original motivations for constructing $\mathbf{RT}(A)$ was to provide a model of full higher-order logic (i.e. a topos) that extended this realizability interpretation. So it would be nice to know that $\mathbf{RT}(A)$ is in some sense the *canonical* way to extend $\mathbf{Mod}(A)$ to a topos. The results of the last section show how $\mathbf{RT}(A)$ canonically extends $\mathbf{Ass}(A)$; this section closes the gap and shows how $\mathbf{Ass}(A)$ canonically extends $\mathbf{Mod}(A)$. (Of course, since $\mathbf{Ass}(A)$ contains $\mathbf{Set}$ as a subcategory, this extension can only really be "canonical" relative to an ambient universe of sets!)

1.4.1 Γ-categories and ∇Γ-categories

Before stating our universal property, we introduce some technical definitions. This material is needed both for this section and for Chapter 2.

First recall that $\mathbf{Mod}(A)$ comes equipped with an exact functor $\Gamma : \mathbf{Mod}(A) \to \mathbf{Set}$, whereas $\mathbf{Ass}(A)$ comes equipped with exact functors Γ, ∇ satisfying certain conditions. We now define 2-categories in which $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$ live as objects.

Definition 1.4.1 (Γ-category) *A* Γ-category *is a regular category $\mathcal{C}$ equipped with an exact functor $\Gamma_C : \mathcal{C} \to \mathbf{Set}$. A* Γ-functor *between Γ-categories is an exact* *functor* $F : \mathcal{C} \to \mathcal{D}$ *such that* $\Gamma_D F \cong \Gamma_C$. *We write* $\Gamma\mathbf{Reg}$ *for the 2-category of* Γ*-categories,* Γ*-functors and natural transformations.*

Definition 1.4.2 ($\nabla\Gamma$-category) *A* $\nabla\Gamma$-category *is a regular category* $\mathcal{C}$ *equipped with exact functors* $\Gamma_C : \mathcal{C} \to \mathbf{Set}$ *and* $\nabla_C : \mathbf{Set} \to \mathcal{C}$ *such that* $\Gamma \dashv \nabla$ *and* $\Gamma\nabla \cong \mathrm{id}$. *A* $\nabla\Gamma$-functor *between* $\nabla\Gamma$*-categories is an exact functor* $F : \mathcal{C} \to \mathcal{D}$ *such that* $\Gamma_D F \cong \Gamma_C$ *and* $F\nabla_C \cong \nabla_D$. *We write* $\nabla\Gamma\mathbf{Reg}$ *for the 2-category of* $\nabla\Gamma$*-categories,* $\nabla\Gamma$*-functors and natural transformations.*

Remark 1.4.3 $\mathbf{Mod}(A)$ is a Γ-category, $\mathbf{Ass}(A)$ is a $\nabla\Gamma$-category, and the inclusion $J : \mathbf{Mod}(A) \hookrightarrow \mathbf{Ass}(A)$ is a Γ-functor. Also it follows from Theorem 1.3.8 and Proposition 1.3.9 that $\mathbf{RT}(A)$ is a $\nabla\Gamma$-category and $I : \mathbf{Ass}(A) \to \mathbf{RT}(A)$ is a $\nabla\Gamma$-functor.

The following proposition says that a $\nabla\Gamma$-functor automatically preserves the unit of the adjunction $\Gamma \dashv \nabla$ modulo the obvious isomorphisms.

Proposition 1.4.4 *If* η, η' *are the units of the adjunctions* $\Gamma_C \dashv \nabla_C$, $\Gamma_D \dashv \nabla_D$ *respectively and* $F : \mathcal{C} \to \mathcal{D}$ *is a* $\nabla\Gamma$*-functor, then* $F\eta \cong \eta'_F$ *via the isomorphism* $F\nabla_C\Gamma_C \cong \nabla_D\Gamma_D F$.

Proof Naturality of η' at $F\eta_X$ gives

$$\begin{tikzcd}
FX \arrow[r, "\eta'_{FX}"] \arrow[d, "F\eta_X"'] & \nabla_D\Gamma_D FX \arrow[d, "\nabla_D\Gamma_D F\eta_X"] \\
F\nabla_C\Gamma_C X \arrow[r, "\eta'_{F\nabla_C\Gamma_C X}"'] & \nabla_D\Gamma_D F\nabla_C\Gamma_C X.
\end{tikzcd}$$

But $\nabla_D\Gamma_D F\eta_X \cong F\nabla_C\Gamma_C\eta_X$, where $\Gamma_C\eta_X$ is an isomorphism; also we have $\eta'_{F\nabla_C\Gamma_C X} \cong \eta'_{\nabla_D\Gamma_D FX}$ which is an isomorphism. Hence $F\eta_X \cong \eta'_{FX}$ via the isomorphism $F\nabla_C\Gamma_C X \cong \nabla_D\Gamma_D FX$. Moreover, this isomorphism is natural in X, and so $F\eta \cong \eta'_F$. $\square$

We are now ready to state the universal property of $\mathbf{Ass}(A)$. The rest of this section will devoted to its proof.

Theorem 1.4.5 *For $(\mathcal{B}, \Gamma_B, \nabla_B)$ any $\nabla\Gamma$-category, the functor*

$$(- \circ J): \ \nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathcal{B}] \ \longrightarrow \ \Gamma\mathbf{Reg}\,[\mathbf{Mod}(A), \mathcal{B}]$$

is part of an equivalence of categories. In particular, for any Γ-functor F : $\mathbf{Mod}(A) \to B$, there is a $\nabla\Gamma$-functor $\overline{F} : \mathbf{Ass}(A) \to B$, unique up to isomorphism, such that the diagram

$$\begin{tikzcd} \mathbf{Mod}(A) \arrow[d, "J"'] \arrow[dr, "F"] & \\ \mathbf{Ass}(A) \arrow[r, "\overline{F}"'] & \mathcal{B} \end{tikzcd}$$

commutes up to natural isomorphism.

Clearly this property determines $\mathbf{Ass}(A)$ uniquely up to equivalence.

1.4.2 Proof of the universal property

This paragraph is devoted entirely to the proof of Theorem 1.4.5. We start with some important definitions:

Definition 1.4.6 *Suppose X is an object of $\mathbf{Ass}(A)$.*

(i) Let A_X denote the set $\bigcup_{x \in |X|} \|x\|$. Let R_X be the modest set defined by

$$|R_X| = A_X, \qquad \|a \in R_X\| = \{a\}.$$

We call R_X the object of realizers *for X.*

(ii) Let X_0 be the assembly defined by

$$|X_0| = \{(x, a) \ \mid \ x \in |X|, \ a \in \|x\|\}, \qquad \|(x, a) \in X_0\| = \{a\}.$$

Note the obvious "projection" morphisms $\pi : X_0 \to X$, $\pi' : X_0 \to R_X$ (both tracked by i). It is easy to see that the square

$$\begin{tikzcd} X_0 \arrow[r, "\eta_{X_0}"] \arrow[d, "\pi'"'] & \nabla|X_0| \arrow[d, "\nabla\pi'"] \\ R_X \arrow[r, "\eta_{R_X}"'] & \nabla A_X \end{tikzcd}$$

is a pullback, and that X is the object appearing in the image factorization of the composite

$$X_0 \xrightarrow{\eta_{X_0}} \nabla|X_0| \xrightarrow{\nabla\pi} \nabla|X|.$$

The idea behind the proof of Theorem 1.4.5 is as follows: Given F, the action of $\overline{F}$ on both modest sets and uniform assemblies is already "determined"; hence $\overline{F}X_0$ is determined since $\overline{F}$ must preserve pullbacks. But now the action of $\overline{F}$ on the above map $X_0 \to \nabla|X|$ is determined; hence $\overline{F}X$ is determined since $\overline{F}$ must preserve image factorizations. We use these facts to construct $\overline{F}$, and then show that it is indeed a $\nabla\Gamma$-functor.

For the rest of this section $F : \mathbf{Mod}(A) \to \mathcal{B}$ is some fixed Γ-functor. Given X an assembly, the object $\overline{F}X$ is defined as follows:

- Let F_0X_0 be the object appearing in the pullback

$$\begin{tikzcd} F_0X_0 \arrow[r, "\theta_X"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \nabla_B|X_0| \arrow[d, "\nabla_B\pi'"] \\ FR_X \arrow[r, "\eta_{FR_X}"'] & \nabla_B A_X \end{tikzcd}$$

- Let $\overline{F}X$ be the object appearing in the image factorization of $\nabla\Gamma\pi \circ \theta_X$ (see Proposition 1.2.12(ii)):

$$\begin{tikzcd} F_0X_0 \arrow[r, "\theta_X"] \arrow[d, two heads] & \nabla_B|X_0| \arrow[d, "\nabla_B\pi"] \\ \overline{F}X \arrow[r, tail] & \nabla_B|X| \end{tikzcd}$$

Given a morphism $f : X \to Y$, choose $t \in A$ such that t tracks f, and let $\alpha_t : A_X \to A_Y$ be the function determined by t. Clearly α_t is a morphism $R_X \to R_Y$ tracked by t. There is also an obvious function $f_0 : |X_0| \to |Y_0|$ given by

$$f_0(x, a) = (fx, ta).$$

We can now define the morphism $\overline{F}f$ as follows:

- Let $F_0 f_0 : F_0 X_0 \to F_0 Y_0$ be the unique morphism making the following diagram commute:

$$\begin{tikzcd}
F_0X_0 \arrow[rr] \arrow[dd] \arrow[dr, "F_0f_0"] & & \nabla_B|X_0| \arrow[dd] \arrow[dr, "\nabla_B f_0"] & \\
& F_0Y_0 \arrow[rr, crossing over] & & \nabla_B|Y_0| \arrow[dd] \\
FR_X \arrow[rr] \arrow[dr, "F\alpha_t"] & & \nabla_B A_X \arrow[dr, "\nabla_B\alpha_t"] & \\
& FR_Y \arrow[rr] \arrow[from=uu, crossing over] & & \nabla_B A_Y
\end{tikzcd}$$

- Let $\overline{F}f : \overline{F}X \to \overline{F}Y$ be the unique morphism making the following diagram commute (see Proposition 1.2.12(iii)):

$$\begin{tikzcd}
F_0X_0 \arrow[r, two heads] \arrow[d, "F_0f_0"'] & \overline{F}X \arrow[r, tail] \arrow[d, "\overline{F}f"] & \nabla_B|X| \arrow[d, "\nabla_B f"] \\
F_0Y_0 \arrow[r, two heads] & \overline{F}Y \arrow[r, tail] & \nabla_B|Y|
\end{tikzcd}$$

Note here that $\overline{F}f$ is independent of the choice of t since $\overline{F}Y \hookrightarrow \nabla_B|Y|$ is mono. Note also that $\overline{F}$ is a functor for a similar reason, and that F_0 is a functor on the subcategory of $\mathbf{Ass}(A)$ consisting of morphisms f_0. We now work towards showing that $\overline{F}$ is a $\nabla\Gamma$-functor. The details are not particularly illuminating and the majority of readers may prefer to skip them.

Proposition 1.4.7 $\Gamma_B\overline{F} \cong \Gamma_A$.

PROOF Since Γ_B preserves pullbacks, for any X the following diagram is a pullback in **Set**:

$$\begin{tikzcd}
\Gamma_B F_0 X_0 \arrow[r, "\Gamma_B\theta_X"] \arrow[d] & \Gamma_B \nabla_B |X_0| \arrow[d, "\Gamma\nabla\pi"] \\
\Gamma_B R_X \arrow[r] & \Gamma_B \nabla_B A_X
\end{tikzcd}$$

Here the bottom arrow is a bijection, hence so is $\Gamma_B\theta_X$. Since Γ_B preserves image factorizations, we have a diagram

$$\begin{tikzcd}
\Gamma_B F_0 X_0 \arrow[r, two heads, "\Gamma_B\theta_X"] \arrow[d] & \Gamma_B \nabla_B |X_0| \arrow[d] \arrow[r, phantom, "\cong"] & \Gamma_A X_0 \arrow[d, "\Gamma_A\pi"] \\
\Gamma_B \overline{F} X \arrow[r, tail] & \Gamma_B \nabla_B |X| \arrow[r, phantom, "\cong"] & \Gamma_A X.
\end{tikzcd}$$

But $\Gamma_A\pi$ is surjective, and so the bottom arrow is a bijection; hence $\Gamma_B\overline{F}X \cong \Gamma_A X$. Naturality follows from the commutativity of

$$\begin{tikzcd}
\overline{F}X \arrow[r, tail] \arrow[d, "\overline{F}f"'] & \nabla_B|X| \arrow[d, "\nabla_B f"] \\
\overline{F}Y \arrow[r, tail] & \nabla_B|Y|.
\end{tikzcd}$$

□

Proposition 1.4.8 $\overline{F}\nabla_A \cong \nabla_B$.

PROOF First define a functor $\nabla' : \mathbf{Set} \to \mathbf{Ass}(A)$ by

$$|\nabla' Z| = Z, \qquad \|z \in \nabla' Z\| = \{i\}, \qquad \nabla'(g : Z \to W) = g.$$

Clearly $\nabla' \cong \nabla_A$. Suppose $X = \nabla' Z$; then we have $A_X \cong 1$, $R_X \cong 1$ and $X_0 \cong X$. So we have

$$F_0 X_0 \;\cong\; \nabla_B|X_0| \;\cong\; \nabla_B|X| \;=\; \nabla_B Z,$$

and hence $\overline{F}\nabla_A Z \cong \overline{F}X \cong \nabla_B Z$. Naturality follows from the commutativity of

$$
\begin{tikzcd}
\overline{F}\nabla' Z \arrow[r] \arrow[d, "\overline{F}\nabla' g"'] & \nabla_B|\nabla' Z| \arrow[d, "\nabla_B g"] \\
\overline{F}\nabla' W \arrow[r] & \nabla_B|\nabla' W|.
\end{tikzcd}
\qquad \square
$$

Proposition 1.4.9 *$\overline{F}$ preserves finite products.*

Proof Clearly $\overline{F}$ preserves the terminal object, since $\overline{F}1 \cong \overline{F}\nabla_A 1 \cong \nabla_B 1 \cong 1$. Given assemblies X, Y, it is easy to see from Proposition 1.2.4 that

$$A_{X\times Y} \;\cong\; A_X \times A_Y, \qquad R_{X\times Y} \;\cong\; R_X \times R_Y, \qquad (X\times Y)_0 \cong X_0 \times Y_0.$$

Hence, as pullbacks commute with products, we have $F_0(X_0 \times Y_0) \cong F_0X_0 \times F_0Y_0$. Also the product of two monos [resp. regular epis] is a mono [resp. regular epi], and so we obtain morphisms

$$F_0(X_0 \times Y_0) \twoheadrightarrow \overline{F}X \times \overline{F}Y \rightarrowtail \nabla_B|X\times Y|.$$

By the uniqueness of image factorizations, we thus have $\overline{F}(X\times Y) \cong \overline{F}X \times \overline{F}Y$. Clearly $\overline{F}$ also preserves the projections modulo this isomorphism. $\square$

Proposition 1.4.10 *$\overline{F}$ preserves equalizers.*

Proof An exhilarating diagram-chase. Suppose $e : Z \to X$ is the equalizer of $f, g : X \rightrightarrows Y$ in $\mathbf{Ass}(A)$ as in the proof of Proposition 1.2.6. We begin with two simple observations:

(1) It is easy to check by direct calculation that the diagram

$$
\begin{tikzcd}
Z_0 \arrow[r, "e_0"] & X_0 \arrow[r, shift left, "\pi f_0"] \arrow[r, shift right, "\pi g_0"'] & Y
\end{tikzcd}
$$

is an equalizer. But Γ_A and ∇_B are exact, and so

$$
\begin{tikzcd}
\nabla_B|Z_0| \arrow[r, "\nabla_B e_0"] & \nabla_B|X_0| \arrow[r, shift left, "\nabla_B(\pi f_0)"] \arrow[r, shift right, "\nabla_B(\pi g_0)"'] & \nabla_B|Y|
\end{tikzcd}
$$

is an equalizer in $\mathcal{B}$.

(2) It is also easy to verify directly that the square

$$\begin{tikzcd} R_Z \arrow[r, tail] \arrow[d, "\alpha_i"'] & \nabla_A A_Z \arrow[d, "\nabla_A \alpha_i"] \\ R_X \arrow[r, tail] & \nabla_A A_X \end{tikzcd}$$

is a pullback. Hence, since F is exact, the bottom face of the appropriate instance of the cube on page 54 is a pullback. But the front and back faces are also pullbacks, and so by the pullback lemma the top face

$$\begin{tikzcd} F_0 Z_0 \arrow[r, "\theta_Z"] \arrow[d, "F_0 e_0"'] & \nabla_B |Z_0| \arrow[d, "\nabla_B e_0"] \\ F_0 X_0 \arrow[r, "\theta_X"'] & \nabla_B |X_0| \end{tikzcd}$$

is a pullback in $\mathcal{B}$.

We now claim that $F_0 e_0$ is the equalizer of the two composites:

$$\begin{tikzcd} F_0 X_0 \arrow[r, "\theta_X"'] & \nabla_B |X_0| \arrow[r, shift left, "\nabla_B(\pi f_0)"] \arrow[r, shift right, "\nabla_B(\pi g_0)"'] & \nabla_B |Y_0| \arrow[r, "\nabla_B \pi"] & \nabla_B |Y|. \end{tikzcd}$$

For suppose $h : W \to F_0 X_0$ is such that the two composite morphisms $W \rightrightarrows \nabla_B |Y|$ are equal. Then observation (1) gives us a unique $h' : W \to \nabla_B |Z_0|$ such that $(\nabla_B e_0) h' = \theta_X h$. But now observation (2) gives us a unique $h'' : W \to F_0 Z_0$ such that $(F_0 e_0) h'' = h$ and $\theta_Z h'' = h'$. Clearly h'' is also the unique morphism such that $(F_0 e_0) h'' = h$.

Next note that we have the commutative diagram

$$\begin{tikzcd} F_0 X_0 \arrow[r, "\theta_X"] \arrow[d, shift right, "F_0 f_0"'] \arrow[d, shift left, "F_0 g_0"] & \nabla_B |X_0| \arrow[d, shift right, "\nabla_B f_0"'] \arrow[d, shift left, "\nabla_B g_0"] \\ F_0 Y_0 \arrow[r, "\theta_Y"'] \arrow[d, two heads, "\epsilon_Y"'] & \nabla_B |Y_0| \arrow[d, "\nabla_B \pi"] \\ \overline{F} Y \arrow[r, tail, "\mu_Y"'] & \nabla_B |Y|. \end{tikzcd}$$

Let $\phi_f = \epsilon_Y(F_0 f_0)$, $\phi_g = \epsilon_Y(F_0 g_0)$. Then $F_0 e_0$ is the equalizer of $\mu_Y \phi_f$ and $\mu_Y \phi_g$, and hence of ϕ_f, ϕ_g since μ_Y is mono.

We now have the following diagram in which all three rows are image factorizations, and the left and right columns are equalizers:

$$
\begin{tikzcd}
F_0 Z_0 \arrow[r, two heads, "\epsilon_Z"] \arrow[d, "F_0 e_0"'] & \overline{F}Z \arrow[r, tail, "\mu_Z"] \arrow[d, "\overline{F}e"'] & \nabla_B|Z| \arrow[d, "\nabla_B e"] \\
F_0 X_0 \arrow[r, two heads, "\epsilon_X"] \arrow[d, shift right, "\phi_f"'] \arrow[d, shift left, "\phi_g"] & \overline{F}X \arrow[r, tail, "\mu_X"] \arrow[d, shift right, "\overline{F}f"'] \arrow[d, shift left, "\overline{F}g"] & \nabla_B|X| \arrow[d, shift right, "\nabla_B f"'] \arrow[d, shift left, "\nabla_B g"] \\
\overline{F}Y \arrow[r, equal] & \overline{F}Y \arrow[r, tail, "\mu_Y"'] & \nabla_B|Y|
\end{tikzcd}
$$

We wish to show that the middle column is also an equalizer. Clearly we have $(\overline{F}f)(\overline{F}e) = (\overline{F}g)(\overline{F}e)$. So suppose we are given $h : W \to \overline{F}X$ such that $(\overline{F}f)h = (\overline{F}g)h$. We first form the pullback

$$
\begin{tikzcd}
V \arrow[r, two heads, "a"] \arrow[d, "b"'] \arrow[dr, phantom, "\lrcorner", very near start] & W \arrow[d, "h"] \\
F_0 X_0 \arrow[r, two heads, "\epsilon_X"'] & \overline{F}X.
\end{tikzcd}
$$

It is easy to see that $\phi_f b = \phi_g b$, so we have a morphism $c : V \to F_0 Z_0$ such that $(F_0 e_0)c = b$. Also we have $(\nabla_B f)\mu_X h = (\nabla_B g)\mu_X h$ and so there is a morphism $d : W \to \nabla_B|Z|$ such that $(\nabla_B e)d = \mu_X h$. We now have the diagram

$$
\begin{tikzcd}
V \arrow[r, two heads, "a"] \arrow[d, "c"'] & W \arrow[r, "d"] & \nabla_B|Z| \arrow[d, equal] \\
F_0 Z_0 \arrow[r, two heads, "\epsilon_Z"'] & \overline{F}Z \arrow[r, tail, "\mu_Z"'] & \nabla_B|Z|.
\end{tikzcd}
$$

To see that this commutes, verify that $(\nabla_B e)da = (\nabla_B e)\mu_Z \epsilon_Z c$ and observe that $\nabla_B e$ is mono. Let $W \xrightarrow{j} U \xrightarrow{i} \nabla_B|Z|$ be the image factorization of d; then by

Proposition 1.2.12(i,iii) we have a morphism $k : U \to \overline{F}Z$ such that $\mu_Z k = i$. Let $h' = kj$. Then $\mu_X(\overline{F}e)h' = \mu_X h$, so $(\overline{F}e)h' = h$ since μ_X is mono.

For uniqueness, suppose we have $(\overline{F}e)h'_1 = (\overline{F}e)h'_2 = h$. Then $(\nabla_B e)\mu_Z h'_1 = (\nabla_B e)\mu_Z h'_2 = \mu_X h$. But $(\nabla_B f)\mu_X h = (\nabla_B g)\mu_X h$ as above, and so $\mu_Z h_1 = \mu_Z h_2$ since the right-hand column is an equalizer. But μ_Z is mono, so $h_1 = h_2$. □

Proposition 1.4.11 *$\overline{F}$ preserves regular epis.*

Proof Suppose $f : X \to Y$ is the coequalizer of its own kernel-pair. By Proposition 1.2.10 we may assume that f is tracked by i and that $\|y\|_Y = \bigcup_{fx=y} \|x\|_X$. Hence $A_X = A_Y$, $R_X = R_Y$ and the morphism $\alpha_i : R_X \to R_Y$ is the identity. Thus the bottom face of the relevant instance of the cube on page 54 is trivially a pullback. Hence by the pullback lemma the top square

$$\begin{tikzcd} F_0X_0 \arrow[r, "\theta_X"] \arrow[d, "F_0f_0"'] & \nabla_B|X_0| \arrow[d, "\nabla_B f_0"] \\ F_0Y_0 \arrow[r, "\theta_Y"] & \nabla_B|Y_0| \end{tikzcd}$$

is a pullback. Now it is easy to see that the obvious function $f_0 : |X_0| \to |Y_0|$ is surjective; hence $\nabla_B f_0$ is regular epi and so $F_0 f_0$ is regular epi. So we have the commutative square

$$\begin{tikzcd} F_0X_0 \arrow[r, two heads, "F_0f_0"] \arrow[d, "\epsilon_X"'] & F_0Y_0 \arrow[d, "\epsilon_Y"] \\ \overline{F}X \arrow[r, "\overline{F}f"] & \overline{F}Y. \end{tikzcd}$$

To show $\overline{F}f$ is regular epi, suppose by Proposition 1.2.12(i) that $\epsilon_Y(F_0 f_0)$ is the coequalizer of some $h, k : Z \rightrightarrows F_0X_0$. We claim $\overline{F}f$ is the coequalizer of $\epsilon_X h, \epsilon_X k$. Clearly $(\overline{F}f)\epsilon_X h = (\overline{F}f)\epsilon_X k$. So suppose we have $g : \overline{F} \to W$ such that $g\epsilon_X h = g\epsilon_X k$. Then there is a unique $g' : \overline{F}Y \to W$ such that

$$g\epsilon_X \;=\; g'\epsilon_Y(F_0 f_0) \;=\; g'(\overline{F}f)\epsilon_X.$$

But ϵ_X is epi, so $g = g'\epsilon_X$; and clearly g' is the unique such morphism. □

Proposition 1.4.12 $\overline{F}J \cong F$.

Proof We first check that $\overline{F}X \cong FX$ for objects X of the form R_Y. Note that $R_X = X$ for such objects, and the projection $X_0 \to R_X$ is an isomorphism, so $\nabla_B|X| \cong \nabla_B A_X$ and $F_0X_0 \cong FR_X$. Moreover $\eta_{FR_X} : FR_X \to \nabla_B A_X$ is mono, since $F\eta_{R_X}$ is mono and $\eta_{FR_X} \cong F\eta_{R_X}$ by Proposition 1.4.4. Also $\pi = \pi' : |X_0| \to A_X$. Thus $(\nabla_B\pi)\theta_X : F_0X_0 \to \nabla_B A_X$ is mono, and so we have $\overline{F}X \cong F_0X_0 \cong FR_X = FX$. Moreover, if $f : X_1 \to X_2$ is a morphism between such objects, it is easy to check that this isomorphism identifies $\overline{F}f$ with Ff.

For a general modest set X, let $r_X : R_X \to X$ be the morphism tracked by i. The kernel-pair of r_X consists of the two projections $p_X, p'_X : P_X \to R_X$, where

$$|P_X| = \{(a,a') \mid r_X a = r_X a'\}, \qquad \|(a,a') \in P_X\| = \{\langle a,a'\rangle\},$$

and clearly r_X is the coequalizer of this pair in $\mathbf{Mod}(A)$. But also $P_X \cong R_{P_X}$, and so by the above we have $\overline{F}P_X \cong FP_X$, $\overline{F}R_X \cong FR_X$ and $\overline{F}p_X \cong Fp_X$, $\overline{F}p'_X \cong Fp'_X$. But both $F, \overline{F}J$ preserve coequalizers of kernel-pairs, so $\overline{F}J(X) \cong F(X)$. It is routine to check that this isomorphism is natural in X. □

Proposition 1.4.13 *$\overline{F}$ is, up to isomorphism, the unique $\nabla\Gamma$-functor such that $\overline{F}J \cong F$.*

Proof By the above six propositions, $\overline{F}$ is a $\nabla\Gamma$-functor and $\overline{F}J = F$, so it only remains to check uniqueness. Given any $\nabla\Gamma$-functor $F' : \mathbf{Ass}(A) \to \mathcal{B}$ such that $F'J \cong F$, for any object X we have the pullback

$$\begin{tikzcd}
F'X_0 \arrow[r, "F'\theta_X"] \arrow[d, "F'\pi'"'] \arrow[dr, phantom, "\lrcorner", very near start] & F'\nabla_A|X_0| \arrow[d, "F'\nabla_A\pi'"] \\
F'R_X \arrow[r, "F'\eta_{R_X}"'] & F'\nabla_A A_X.
\end{tikzcd}$$

But $F'\eta_{R_X} \cong \eta_{F'R_X} \cong \eta_{FR_X}$ by Proposition 1.4.4, and $F'\nabla_A\pi' \cong \nabla_B\pi'$. Hence $F'X_0 \cong F_0X_0$ by an isomorphism that identifies the morphism $F'((\nabla_A\pi)\eta_{X_0})$ with $(\nabla_B\pi)\theta_X$. But the objects appearing in the image factorizations of these two morphisms are $F'X$ and $\overline{F}X$ respectively, so $F'X \cong \overline{F}X$. Again, naturality in X is routine, so $F' \cong \overline{F}$ as required. □

To complete the proof of Theorem 1.4.5, we have to show that the correspondence between F and $\overline{F}$ extends to an equivalence of categories. The following small lemma will be helpful.

Lemma 1.4.14 *Suppose $F, G : \mathbf{Mod}(A) \to \mathcal{B}$ [resp. $\mathbf{Ass}(A) \to \mathcal{B}$] are two Γ-functors, and $\beta : F \to G$ is any natural transformation. Then for Y any modest set [resp. assembly], the following triangle commutes:*

$$\begin{tikzcd}
FY \arrow[r, "\eta_{FY}"] \arrow[d, "\beta_Y"'] & \nabla_B|Y| \\
GY \arrow[ur, "\eta_{GY}"'] &
\end{tikzcd}$$

Proof For every morphism $y : 1 \to Y$ we have $\beta_Y(Fy) = (Gy)\beta_1$ by naturality of β, and so the morphism $\Gamma_B \beta_Y$ is identified with $\mathrm{id}_{|Y|}$ under the isomorphisms $\Gamma_B F \cong \Gamma_A \cong \Gamma_B G$. Thus $\nabla_B \Gamma_B \beta_Y \cong \mathrm{id}_{\nabla_B |Y|}$, and so the triangle commutes by naturality of η at β_Y. $\Box$

Proposition 1.4.15 *Suppose given Γ-functors $F, G : \mathbf{Mod}(A) \to \mathcal{B}$ and a natural transformation $\beta : F \to G$. Then there is a unique $\overline{\beta} : \overline{F} \to \overline{G}$ such that $\overline{\beta} J$ is canonically isomorphic to β.*

Proof Let X be any assembly. Applying the lemma to the modest set R_X, define β_{0X_0} to be the unique morphism making the diagram

$$\begin{tikzcd}
F_0X_0 \arrow[rr, "\theta_X"] \arrow[dr, "\beta_{0X_0}"'] \arrow[dd] & & \nabla_B|X_0| \arrow[dd, "\nabla_B \pi'"] \\
& G_0X_0 \arrow[ur, "\theta'_X"'] \arrow[dd] & \\
FR_X \arrow[rr, "\eta_{FR_X}", crossing over] \arrow[dr, "\beta_{R_X}"'] & & \nabla_B A_X \\
& GR_X \arrow[ur, "\eta_{GR_X}"'] &
\end{tikzcd}$$

commute, and define $\overline{\beta}_X$ to be the unique morphism making the diagram

$$\begin{tikzcd}
F_0X_0 \arrow[r, two heads] \arrow[d, "\beta_{0X_0}"'] & \overline{F}X \arrow[r, tail] \arrow[d, "\overline{\beta}_X"] & \nabla_B|X_0| \arrow[d, equal] \\
G_0X_0 \arrow[r, two heads] & \overline{G}X \arrow[r, tail] & \nabla_B|X_0|
\end{tikzcd}$$

commute. (Note that the outer rectangle commutes since we have $(\nabla_B\pi)\theta_X = p(\nabla_B\pi)\theta'_X\beta_{0X_0}$.) To see that $\overline{\beta}$ is natural, given $f : X \to Y$ we have $\beta_{R_Y}(Ff) = (Gf)\beta_{R_X}$ by naturality of β, so $\beta_{0Y_0}(F_0f_0) = (G_0f_0)\beta_{0X_0}$ by uniqueness of mediating morphisms, hence $\overline{\beta}_Y(\overline{F}f) = (\overline{G}f)\overline{\beta}_X$ by uniqueness of fill-in morphisms.

To check that $\overline{\beta}J \cong \beta$ under the canonical isomorphisms, note that for X a modest set we have

$$(\nabla_B\pi)\theta_X \;\cong\; (F\nabla_A\pi)(F\eta_{X_0}) \;\cong\; (F\eta_X)(F\pi) \;\cong\; \eta_{FX}(Fr_X)(F\pi')$$

and similarly for G, so by uniqueness of image factorizations and fill-ins it suffices to check the commutativity of

$$\begin{tikzcd}
F_0X_0 \arrow[r, two heads, "F\pi'"] \arrow[d, "\beta_{0X_0}"'] & FR_X \arrow[r, two heads, "Fr_X"] \arrow[d, "\beta_{R_X}"'] & FX \arrow[r, tail, "\eta_FX"] \arrow[d, "\beta_X"] & \nabla_B|X| \arrow[d, equal] \\
G_0X_0 \arrow[r, two heads, "G\pi'"'] & GR_X \arrow[r, two heads, "Gr_X"'] & GX \arrow[r, tail, "\eta_GX"'] & \nabla_B|X|.
\end{tikzcd}$$

But here the left square is the left face of the above prism; the middle square commutes by naturality of β; and the right square commutes by the above lemma.

To show uniqueness, suppose we have $\beta' : \overline{F} \to \overline{G}$ such that $\beta'J \cong \beta$. For any assembly X, by the proof of Proposition 1.4.13 we have $F_0X_0 \cong \overline{F}X_0$, $\theta_X \cong \eta_{\overline{F}X_0}$ and similarly for G. Hence, replacing β_{0X_0} by $\beta'X_0$ in the above prism, the top face commutes by the lemma and the left face by naturality since $\beta_{R_X} \cong \beta_{R_X}$. So $\beta'_{X_0} = \beta_{0X_0}$. But now replacing $\overline{\beta}_X$ by β'_X in the rectangle used to define $\overline{\beta}_X$, we see that the left square commutes by naturality of β' and the right square by the lemma. Hence $\beta'_X = \overline{\beta}_X$. $\Box$

It is now clear that the functor

$$(- \circ J): \ \nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathcal{B}] \ \longrightarrow \ \Gamma\mathbf{Reg}\,[\mathbf{Mod}(A), \mathcal{B}]$$

is part of an equivalence of categories: by Proposition 1.4.13 it is essentially bijective on objects; hence Proposition 1.4.15 implies that it is also full and faithful. The proof of Theorem 1.4.5 is now complete.

Remark 1.4.16 It would be interesting to know whether the existence of the universal $\nabla\Gamma$-category $\mathbf{Ass}(A)$ reflects a particular property of $\mathbf{Mod}(A)$, or whether the passage from $\mathbf{Mod}(A)$ to $\mathbf{Ass}(A)$ is an instance of some general construction of the "free $\nabla\Gamma$-category on a ∇-category". In particular, the above proof is in many respects strongly reminiscent of the Freyd glueing or *scone* construction (see [29, §1(10)] or [52, II.22]), though we have not yet made this connection precise.

Chapter 2

Morphisms between PCAs

In this chapter we introduce a new kind of morphism between PCAs, and show how these morphisms give rise to $\nabla\Gamma$-functors between the corresponding categories of assemblies. Quite surprisingly, it turns out that essentially *every* $\nabla\Gamma$-functor between the categories of assemblies arises in this way, and so we obtain an equivalence between PCA morphisms $A \to B$ and $\nabla\Gamma$-functors $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$. One application of this result is to show that the categories $\mathbf{Ass}(A)$, $\mathbf{Ass}(B)$ are equivalent iff the PCAs A, B are "equivalent" in a certain sense. We also obtain similar results for **RT** and **Mod**.

The first three sections of this chapter run roughly parallel to Sections 1–3 of Chapter 1. In Section 2.1 we consider PCAs by themselves, and introduce the 2-category **PCA** of PCAs, *applicative morphisms* and *applicative transformations*. In Section 2.2 we consider categories of assemblies, and show how the construction **Ass** of Chapter 1 extends to a 2-functor from **PCA** to the 2-category $\nabla\Gamma\mathbf{Reg}$; we also prove the equivalence result mentioned above. Most of the hard work is done in this section. In Section 2.3 we show how these results can be extended to realizability toposes without much effort.

It is natural to ask whether particular properties of applicative morphisms are reflected in properties of the functors they give rise to. In Section 2.4 we consider three such properties, and give categorical characterizations of the corresponding functors. In Section 2.5 we conclude the chapter with a discussion of *adjoint pairs* of applicative morphisms and some of the properties they enjoy.

2.1 Applicative morphisms

We start by introducing a new notion of morphism between PCAs. One might think that the natural choice of "morphisms" between PCAs would be *PCA homomorphisms* (i.e. functions that respect k, s and application); however, our notion is much more general than this, and seems more appropriate for studying the categories introduced in Chapter 1.

Definition 2.1.1 (Applicative morphism) *Let A, B be two PCAs. An* applicative morphism *from A to B is a total relation $\gamma : A \longrightarrow\!\!\!\!\!\!+\;\, B$ such that there exists $r \in B$ satisfying*

$$\begin{aligned} \gamma(a,b) &\implies rb\downarrow; \\ \gamma(a,b) \wedge \gamma(a',b') \wedge aa'\downarrow &\implies \gamma(aa', rbb') \end{aligned}$$

for all $a, a' \in A,\ b, b' \in B$. We say that such an element r realizes *γ.*

Remarks 2.1.2 (i) The condition $\gamma(a,b) \implies rb\downarrow$ is not strictly necessary, since if r satisfies just the second condition then $\lambda^* xy.rxy$ satisfies both conditions (see Proposition 1.1.6). Nevertheless, when we say "r realizes γ" we mean that r satisfies both conditions.

(ii) We will overload our notation and write $\gamma(a)$ for $\{b \in B \mid \gamma(a,b)\}$. If $P \subseteq A$, we write $\gamma^+(P)$ for $\bigcup_{a \in P} \gamma(a)$.

The second condition in Definition 2.1.1 may be summarized by the slogan: "application in A is effective in B." If we think of A, B informally as "abstract machines", we may think of γ as specifying an "implementation" of A on B. The predicate $\gamma(a,b)$ may be read as "b is a representation of a in B", and r may be thought of as a kind of "interpreter" for A in B. We now give some examples for motivation—many more examples will appear in Chapter 3.

Examples 2.1.3 (i) Let $\ulcorner - \urcorner : \Lambda \to \mathrm{N}$ be some effective encoding of lambda-terms as natural numbers. From $\ulcorner - \urcorner$ we can define a total relation $\gamma : \Lambda/\beta \longrightarrow\!\!\!\!\!\!+\;\, \mathrm{N}$

by $\gamma(m) = \{\ulcorner M \urcorner \mid M \in m\}$. If our encoding is sensibly chosen, juxtaposition of lambda-terms is *effective*—there is a partial recursive function $\varphi : \mathrm{N} \times \mathrm{N} \rightharpoonup \mathrm{N}$ such that $\varphi(\ulcorner M \urcorner, \ulcorner N \urcorner) = \ulcorner MN \urcorner$. Hence γ is an applicative morphism $\Lambda/\beta \longrightarrow\!\!\!\!\!+ K_1$: for a realizer take any $r \in K_1$ such that $r \cdot x \cdot y \simeq \varphi(x, y)$.

(ii) Any PCA homomorphism can be seen as an applicative morphism. In fact, suppose $f : A \to B$ is any function satisfying

$$aa'\downarrow \implies f(aa') = f(a) \cdot f(a')$$

(f need not respect k and s). Then f regarded as a total relation is an applicative morphism: indeed $\lambda^* xy.xy \in B$ is a realizer for f.

It is easy to see that PCAs and applicative morphisms form a category. The identity applicative morphism on A is given by the identity relation—this is realized by $\lambda^* xy.xy$. Given $\gamma : A \longrightarrow\!\!\!\!\!+ B$, $\delta : B \longrightarrow\!\!\!\!\!+ C$ realized by q, r respectively, their composition is defined by $(\delta\gamma)(a) = \delta^+(\gamma(a))$, and $\delta\gamma$ is realized by $\lambda^* xy.r(rq'x)y$ for any $q' \in \delta(q)$. We now show how to make this category into a 2-category.

Definition 2.1.4 (Applicative transformation) *Let $\gamma, \delta : A \longrightarrow\!\!\!\!\!+ B$ be two applicative morphisms. We say there is an* applicative transformation *from γ to δ (and write $\gamma \preceq \delta$) if there exists $t \in B$ such that*

$$\gamma(a, b) \implies \delta(a, tb)$$

for all $a, b \in B$. (In this case we say that t realizes *$\gamma \preceq \delta$.) We write $\gamma \sim \delta$ if both $\gamma \preceq \delta$ and $\delta \preceq \gamma$.*

Continuing our analogy with abstract machines, $\gamma \preceq \delta$ can be read as "there is a program that translates γ-implementations into δ-implementations". Note that there can be at most one applicative transformation from γ to δ; hence there is usually no need to give letters as names for applicative transformations.

Example 2.1.5 Let $\ulcorner - \urcorner_1, \ulcorner - \urcorner_2 : \Lambda \to \mathrm{N}$ be two effective encodings of closed lambda-terms as natural numbers, with $\gamma_1, \gamma_2 : \Lambda/\beta \longrightarrow\!\!\!\!\!+ K_1$ the resulting applicative morphisms as above. Then there are partial recursive functions α, β such that for all $M \in \Lambda$ we have $\alpha(\lceil M \rceil_1) = \lceil M \rceil_2$ and $\beta(\lceil M \rceil_2) = \lceil M \rceil_1$. It follows easily that $\gamma_1 \sim \gamma_2$.

Proposition 2.1.6 *PCAs, applicative morphisms and applicative transformations form a preorder-enriched category* **PCA**.

PROOF We have already seen that PCAs and applicative morphisms form a category. The set of applicative morphisms $A \longrightarrow\!\!\!\!\!\!+\; B$ is preordered by $\preceq$, since $\gamma \preceq \gamma$ is realized by i, and if $\gamma \preceq \delta$, $\delta \preceq \epsilon$ are realized by t, u respectively then $\gamma \preceq \epsilon$ is realized by $\lambda^* x.u(tx)$. Finally, composition of applicative morphisms respects $\preceq$: if t realizes $\gamma \preceq \gamma' : A \longrightarrow\!\!\!\!\!\!+\; B$, u realizes $\delta \preceq \delta' : B \longrightarrow\!\!\!\!\!\!+\; C$ and r realizes δ, then $\lambda^* x.u(rt'x)$ realizes $\delta\gamma \preceq \delta'\gamma'$ for any $t' \in \delta(t)$. $\Box$

From **PCA** we obtain the poset-enriched category **PCA**$/\sim$ by identifying γ, δ whenever $\gamma \sim \delta$. The categories **PCA**, **PCA**$/\sim$ both seem to be very poor in their categorical structure, but it is worth observing the following:

Proposition 2.1.7 *The PCA* **1** *is both initial and terminal in* **PCA**$/\sim$.

PROOF Terminality is trivial (in fact **1** is also terminal in **PCA**). For initiality, any non-empty subset $A' \subseteq A$ determines an applicative morphism $1 \longrightarrow\!\!\!\!\!\!+\; A$ (realized by $\lambda^* xy.a$ for any $a \in A'$). And given $\gamma, \gamma' : 1 \longrightarrow\!\!\!\!\!\!+\; A$, we have $\gamma \preceq \gamma' \preceq \gamma$ (these are realized by $\lambda^* x.a$, $\lambda^* x.a'$ respectively, where $a \in \gamma(*)$, $a' \in \gamma'(*)$). $\Box$

It is also perhaps worth remarking that the morphisms from A to B in **PCA**$/\sim$ form a semilattice when ordered by $\preceq$: given $\gamma, \delta : A \longrightarrow\!\!\!\!\!\!+\; B$, their meet $\gamma \wedge \delta$ may be defined by

$$(\gamma \wedge \delta)(a, b) \iff \gamma(a, \mathit{fst}\ b) \ \wedge\ \delta(a, \mathit{snd}\ b).$$

If q, r realize γ, δ respectively then $\gamma \wedge \delta$ is realized by the element

$$\lambda^* xy.\ \langle\, q(\mathit{fst}\ x)(\mathit{fst}\ y),\ r(\mathit{snd}\ x)(\mathit{snd}\ y)\,\rangle.$$

However, **PCA**$/\sim$ is *not* semilattice-enriched with respect to $\preceq_\Sigma$, since composition does not in general preserve meets.

2.2 Functors between categories of assemblies

Our purpose in this section is to show how applicative morphisms between PCAs are related to functors between the corresponding categories of assemblies. This section is independent of Section 1.3.

2.2.1 Building functors from applicative morphisms

We first show how an applicative morphism $\gamma : A \longrightarrow\!\!\!\!\!\!\!\!+\;\; B$ gives rise to a functor $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$. The idea is very simple:

Definition 2.2.1 *Suppose $\gamma : A \longrightarrow\!\!\!\!\!\!\!\!+\;\; B$ is an applicative morphism. We define the functor $\gamma_* : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ by*

$$\begin{aligned} |\gamma_* X| &= |X|, \qquad \|x \in \gamma_* X\| = \gamma^+ \|x \in X\|; \\ \gamma_*(f : X \to Y) &= f : \gamma_* X \to \gamma_* Y. \end{aligned}$$

To see that this is a good definition, note that if r realizes γ and the morphism $f : X \to Y$ is tracked by t, then $f : \gamma_* X \to \gamma_* Y$ is tracked by rt' for any $t' \in \gamma(t)$. The fact that γ_* preserves identities and composition is trivial. The next three propositions give some important properties of γ_*.

Proposition 2.2.2 *The functor γ_* preserves finite limits.*

Proof *Finite products:* It is easy to see that $\gamma_*(1) \cong 1$. Given X, Y in $\mathbf{Ass}(A)$, consider the two objects

$$P = \gamma_* X \times \gamma_* Y, \qquad Q = \gamma_*(X \times Y).$$

From the proof of Proposition 1.2.4, we see that $|P| = |Q| = |X| \times |Y|$, but that

$$\begin{aligned} \|(x,y) \in P\| &= \{\langle b, b' \rangle \mid b \in \gamma^+ \|x\|,\ b' \in \gamma^+ \|y\|\}, \\ \|(x,y) \in Q\| &= \gamma^+ \{\langle a, a' \rangle \mid a \in \|x\|,\ a' \in \|y\|\}. \end{aligned}$$

To see that $P \cong Q$, note that the identity map $|P| \to |Q|$ is tracked by

$$\lambda^* b.\ r(r\ pair'\ (fst_B\ b))(snd_B\ b),$$

where r realizes γ and $pair' \in \gamma(pair_A)$; and its inverse is tracked by

$$\lambda^* b.\ \langle (r\ fst'\ b),\ (r\ snd'\ b) \rangle,$$

where $fst' \in \gamma(fst_A)$, $snd' \in \gamma(snd_A)$. Clearly γ_* also preserves the projections up to this isomorphism.

Equalizers: Given $f, g : X \rightrightarrows Y$ in $\mathbf{Ass}(A)$, we see from 1.2.6 that the equalizer of $\gamma_*(f)$, $\gamma_*(g)$ is $e : Z \to \gamma_*(X)$, where

$$|Z| = \{x \in X \mid fx = gx\}, \qquad \|x\|_Z = \gamma^+ \|x\|_X,$$

and e is the evident morphism tracked by i. But $Z \xrightarrow{e} \gamma_*(X)$ is precisely $\gamma_*(d)$, where d is the equalizer of f, g as described in 1.2.6. Thus γ_* preserves these "chosen" equalizers on the nose. $\square$

Proposition 2.2.3 *The functor γ_* preserves coequalizers of kernel-pairs.*

Proof Given $f : X \to Y$ in $\mathbf{Ass}(A)$, we see from 1.2.10 that the coequalizer of the kernel-pair of $\gamma_*(f)$ is $c : \gamma_*(X) \to W$, where

$$|W| = \mathrm{Im}\ f, \qquad \|y\|_W = \bigcup_{fx=y} \gamma^+ \|x\|_X = \gamma^+ \Big(\bigcup_{fx=y} \|x\|_X \Big),$$

and c is the evident morphism tracked by i. But $\gamma_*(X) \xrightarrow{c} W$ is precisely $\gamma_*(b)$, where b is the coequalizer of the kernel-pair of f. Thus γ_* preserves chosen coequalizers of kernel-pairs on the nose. $\square$

Proposition 2.2.4 *The following diagrams commute up to natural isomorphism:*

$$\begin{tikzcd}
\mathbf{Ass}(A) \arrow[r, "\gamma_*"] & \mathbf{Ass}(B) \\
\mathbf{Set} \arrow[u, "\nabla_A"] \arrow[ur, "\nabla_B"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{Ass}(A) \arrow[r, "\gamma_*"] \arrow[d, "\Gamma_A"'] & \mathbf{Ass}(B) \arrow[dl, "\Gamma_B"] \\
\mathbf{Set} &
\end{tikzcd}$$

Proof For Z any set, the identity map $|\gamma_* \nabla_A Z| \to |\nabla_B Z|$ is tracked by i, and its inverse is tracked by $\lambda^* x.a$ for any $a \in \gamma^+(A)$. Thus $\gamma_*(\nabla_A Z) \cong \nabla_B Z$. Naturality in X is trivial, so $\gamma_* \nabla_A \cong \nabla_B$. The second diagram actually commutes on the nose by definition of γ_* and ∇. $\square$

The above three propositions together say that γ_* is a $\nabla\Gamma$-functor (see Definition 1.4.2). It is easy to see that $\mathrm{id}_* = \mathrm{id}$ and $(\delta\gamma)_* = \delta_*\gamma_*$, and so by setting $\mathbf{Ass}(\gamma) = \gamma_*$ we obtain a functor $\mathbf{Ass} : \mathbf{PCA} \to \nabla\Gamma\mathbf{Reg}$. It is now easy to extend this functor to a 2-functor:

Definition 2.2.5 *Given $\gamma, \delta : A \multimap B$ in* **PCA** *and an applicative transformation $\zeta : \gamma \preceq \delta$, let $\zeta_* : \gamma_* \to \delta_*$ be the natural transformation defined by*

$$\zeta_{*X} = \mathrm{id}_{|X|} : \gamma_* X \to \delta_* X.$$

This is a good definition: if $t \in B$ realizes $\gamma \preceq \delta$ then t tracks each ζ_{*X}, and the naturality condition is trivial. The construction of ζ_* from ζ is clearly functorial: $(\mathrm{id}_\gamma)_* = \mathrm{id}_{\gamma_*}$, and if $\zeta : \gamma \preceq \delta$ and $\xi : \delta \preceq \epsilon$ then $(\xi\zeta)_* = \xi_*\zeta_*$. Finally, the construction $\zeta \mapsto \zeta_*$ trivially respects horizontal composition of 2-cells: given $\gamma, \gamma' : A \rightrightarrows B$, $\delta, \delta' : B \rightrightarrows C$ and $\zeta : \gamma \preceq \gamma'$, $\xi : \delta \preceq \delta'$, we have $(\xi \mid \zeta)_* = \xi_* \mid \zeta_*$. So we have proved

Theorem 2.2.6 *Setting $\mathbf{Ass}(\zeta) = \zeta_*$ for ζ an applicative transformation, we obtain a 2-functor $\mathbf{Ass} : \mathbf{PCA} \to \nabla\Gamma\mathbf{Reg}$.* $\square$

Remark 2.2.7 Recall that $\mathbf{1}$ is both initial and terminal in $\mathbf{PCA}/\sim$. It is easy to check that $\mathbf{Ass}(\mathbf{1}) \simeq \mathbf{Set}$, and for any A the essentially unique applicative morphisms $\mathbf{1} \multimap A \multimap \mathbf{1}$ give rise to the functors ∇_A and Γ_A respectively.

2.2.2 Recovering applicative morphisms

We now work towards the main result of this section: every $\nabla\Gamma$-functor $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$ is naturally isomorphic to γ_* for some $\gamma : A \multimap B$, and every natural transformation between such functors arises from an applicative transformation. Throughout this paragraph F will denote some fixed $\nabla\Gamma$-functor $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$. First we show how we can recover an applicative morphism $A \multimap B$ from F.

Definition 2.2.8 *(Cf. Definition 1.4.6.) Let R be the assembly (over A) defined by*

$$|R| \;=\; A; \qquad \|a \in R\| \;=\; \{a\}$$

(We call R the object of realizers *in $\mathbf{Ass}(A)$.) Let $\iota : A = \Gamma_A R \to \Gamma_B(FR)$ denote the canonical bijection, and let $\mathbf{App}(F)$ be the total relation $A \twoheadrightarrow B$ defined by*

$$\mathbf{App}(F)(a) \;=\; \|\iota(a) \in FR\|.$$

To show that $\mathbf{App}(F)$ is an applicative morphism, it is helpful to introduce the following definition:

Definition 2.2.9 (Cartesian morphism) *A morphism $f : X \to Y$ of $\mathbf{Ass}(A)$ is* cartesian *if the following square is a pullback:*

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\
\nabla\Gamma X \arrow[r, "\nabla\Gamma f"'] & \nabla\Gamma Y
\end{tikzcd}$$

Lemma 2.2.10 *(i) A morphism $f : X \to Y$ is cartesian iff $f \cong f' : X' \to Y$ for some f' such that $\|x \in X'\| = \|f'x \in Y\|$ for all $x \in |X'|$.*

(ii) Any $\nabla\Gamma$-functor $H : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ preserves cartesian morphisms.

Proof (i) For both implications, it suffices to note that for any f there is a "canonical" pullback $f' : X' \to Y$ of $\nabla\Gamma f$ along η_Y described by:

$$|X'| = |X|, \qquad \|x \in X'\| = \|fx \in Y\|, \qquad f' = f : |X'| \to |Y|.$$

(ii) Note that $H\eta_X \cong \eta_{HX}$ by Proposition 1.4.4 and H preserves pullbacks. $\Box$

Proposition 2.2.11 $\mathbf{App}(F)$ *is an applicative morphism.*

Proof Write γ for $\mathbf{App}(F)$; we just need to obtain a realizer for γ. Let $C \subseteq A \times A$ be the set $C = \{(a, a') \mid aa'\!\downarrow\}$, and define $P, Q \in \mathbf{Ass}(A)$ by

$$|P| \;=\; |Q| \;=\; C, \qquad \|(a, a') \in P\| \;=\; \{\langle a, a'\rangle\}, \qquad \|(a, a') \in Q\| \;=\; \{aa'\}.$$

Let $p : P \to R$, $q : Q \to R$ be the cartesian morphisms tracked by $i \in A$. By the Lemma, Fp and Fq are cartesian and so are isomorphic to morphisms $P' \xrightarrow{p'} FR$, $Q' \xrightarrow{q'} FR$ respectively, where $|P'| = |Q'| = C$ and p', q' are tracked by i. Now let $h : P \to Q$ be the morphism determined by id_C (and tracked by $\lambda^* xy.xy$), and let h' be the evident morphism $P' \to Q'$ such that $h' \cong Fh$. Also let $j : R \times R \to R$ be the map $(a, a') \mapsto \langle a, a' \rangle$ (this is tracked by i); and let l be the morphism $FR \times FR \cong F(R \times R) \xrightarrow{Fj} FR$.

Suppose $t \in B$ tracks h' and u tracks l. We claim $r = \lambda^* xy.t(u\langle x, y \rangle)$ is a realizer for γ. For suppose $b \in \gamma(a)$, $b' \in \gamma(a')$ and $aa' \downarrow$. Then $b \in \|a \in FR\|$, $b' \in \|a' \in FR\|$ and so $u\langle b, b' \rangle \in \|\langle a, a' \rangle \in FR\|$. Thus $u\langle b, b' \rangle \in \|(a, a') \in P'\|$ by choice of P'. Hence $t(u\langle b, b' \rangle) \in \|(a, a') \in Q'\|$. But now we have $\|(a, a') \in Q'\| = \|aa' \in FR\| = \gamma(aa')$, so $rbb' \in \gamma(aa')$. $\square$

We now show that, for given A, B, the operation **App** can be made into a functor. In the following propositions, F, G both denote $\nabla\Gamma$-functors $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$, and $\alpha : F \to G$ is a natural transformation.

Lemma 2.2.12 *For any set X, the morphism $\alpha_{\nabla X} : F\nabla X \to G\nabla X$ is isomorphic to $\mathrm{id}_{\nabla X}$ via the canonical isomorphisms $F\nabla_A \cong \nabla_B \cong G\nabla_A$.*

Proof Suppose $\beta : \nabla_B X \to \nabla_B X$ is the morphism isomorphic to $\alpha_{\nabla X}$. For every function $x : 1 \to X$, naturality of α gives $\beta(x(*)) = x(*)$. Hence $\beta = \mathrm{id}$. $\square$

Proposition 2.2.13 $\mathbf{App}(F) \preceq \mathbf{App}(G)$.

Proof Consider the diagram

$$\begin{tikzcd} FR \arrow[r, "\alpha_R"] \arrow[d, "F\eta_R"'] & GR \arrow[d, "G\eta_R"] \\ F\nabla\Gamma R \arrow[r, "\alpha_{\nabla\Gamma R}"'] & G\nabla\Gamma R. \end{tikzcd}$$

Note first that all four of these objects have underlying set isomorphic to A. So by applying the functor Γ_B we obtain a commutative square in **Set** in which all four objects are A. In this diagram, the bottom arrow is the identity by the Lemma, and the two sides are easily seen to be the identity since $F\eta_R \cong \eta_{FR}$ etc. Hence the function underlying α_R is isomorphic to id_A. Suppose t tracks α_R; then for all $a \in A$ and all $b \in \mathbf{App}(F)(a)$ we have $tb \in \mathbf{App}(G)(a)$. So t realizes $\mathbf{App}(F) \preceq \mathbf{App}(G)$. $\Box$

Since the hom-category $\mathbf{PCA}\,[A, B]$ is a preorder, it is now immediate that $\mathbf{App}$ is a functor from $\nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)]$ to $\mathbf{PCA}\,[A, B]$. Moreover, it is trivial from the definitions that $\mathbf{App}(\gamma_*) = \gamma$ for any applicative morphism γ, and so $\mathbf{App} \circ \mathbf{Ass} = \mathrm{id}$.

2.2.3 The equivalence theorem

In this paragraph we show that $\mathbf{Ass} \circ \mathbf{App} \cong \mathrm{id}$, completing the proof of the "equivalence" between applicative morphisms and $\nabla\Gamma$-functors. Again, $F : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ denotes some fixed $\nabla\Gamma$-functor, and we write γ for $\mathbf{App}(F)$.

First we wish to show that $\gamma_* \cong F$. We start by showing that $\gamma_* X \cong FX$ for two particular objects X:

Definition 2.2.14 *(i) Let E be the object of* $\mathbf{Ass}(A)$ *defined by*

$$|E| \;=\; \{(a, S) \mid a \in S \subseteq A\}, \qquad \|(a, S) \in E\| \;=\; \{a\}.$$

(ii) Let D be the object of $\mathbf{Ass}(A)$ *defined by*

$$|D| \;=\; \mathcal{P}_1 A, \qquad \|S \in D\| \;=\; S.$$

We call D the generic object *of* $\mathbf{Ass}(A)$.

Lemma 2.2.15 *There are canonical isomorphisms $FE \cong \gamma_* E$ and $FD \cong \gamma_* D$.*

Proof Write $\pi : E \to R$ for the obvious projection tracked by i. Note that π is cartesian, and so by Lemma 2.2.10(ii) both the following squares are pullbacks:

$$\begin{array}{ccc} FE & \xrightarrow{F\pi} & FR \\ \Big\downarrow{\scriptstyle \eta_{FE}} & & \Big\downarrow{\scriptstyle \eta_{FR}} \\ \nabla_B|E| & \xrightarrow[\nabla_B\pi]{} & \nabla_B A \end{array} \qquad \begin{array}{ccc} \gamma_* E & \xrightarrow{\gamma_*\pi} & \gamma_* R \\ \Big\downarrow{\scriptstyle \eta_{\gamma_* E}} & & \Big\downarrow{\scriptstyle \eta_{\gamma_* R}} \\ \nabla_B|E| & \xrightarrow[\nabla_B\pi]{} & \nabla_B A. \end{array}$$

But $FR \cong \gamma_* R$ by definition of γ_*, hence $\eta_{FR} \cong \eta_{\gamma_* R}$, and so $FE \cong \gamma_* E$.

Now let $\pi' : E \to D$ be the projection $(a, S) \mapsto S$ (this is tracked by i). Note that

$$\|S \in D\| \;=\; \bigcup_{\pi'(a,S')=S} \|(a, S') \in D\|,$$

and so π' is regular epi. Also $\eta_{FD} \cong F\eta_D$ and η_D is mono, so η_{FD} is mono. So we have the commutative squares

$$\begin{array}{ccc} FE & \xrightarrow{\eta_{FE}} & \nabla_B|E| \\ \Big\downarrow{\scriptstyle F\pi'} & & \Big\downarrow{\scriptstyle \nabla_B\pi'} \\ FD & \xrightarrow[\eta_{FD}]{} & \nabla_B|D| \end{array} \qquad \begin{array}{ccc} \gamma_* E & \xrightarrow{\eta_{\gamma_* E}} & \nabla_B|E| \\ \Big\downarrow{\scriptstyle \gamma_*\pi'} & & \Big\downarrow{\scriptstyle \nabla_B\pi'} \\ \gamma_* D & \xrightarrow[\eta_{\gamma_* D}]{} & \nabla_B|D|. \end{array}$$

But $FE \cong \gamma_* E$ so $\eta_{FE} \cong \eta_{\gamma_* E}$; hence $FD \cong \gamma_* D$ by uniqueness of image factorizations. $\square$

To obtain the $FX \cong \gamma_* X$ for a general object X, we exploit the following property of D. (It is because of this property that D is called a generic object.)

Lemma 2.2.16 *For any object X of* $\mathbf{Ass}(A)$*, there exists a cartesian morphism $\phi : X \to D$. (Note that ϕ is not unique in general!)*

Proof Define $\phi : |X| \to \mathcal{P}_1 A$ by $\phi(x) = \|x\|_X$; note that ϕ is tracked by i. Clearly $\|x\|_X = \|\phi(x)\|_D$ for all $x \in |X|$, so ϕ is cartesian by Lemma 2.2.10(i). $\square$

Proposition 2.2.17 $F \cong \gamma_*$.

PROOF Given X in $\mathbf{Ass}(A)$, take $\phi : X \to D$ cartesian as in Lemma 2.2.16. Then both the following squares are pullbacks:

$$\begin{tikzcd} FX \arrow[r, "F\phi"] \arrow[d, "\eta_{FX}"'] & FD \arrow[d, "\eta_{FD}"] \\ \nabla_B|X| \arrow[r, "\nabla_B\phi"'] & \nabla_B|D| \end{tikzcd} \qquad \begin{tikzcd} \gamma_* X \arrow[r, "\gamma_*\phi"] \arrow[d, "\eta_{\gamma_* X}"'] & \gamma_* D \arrow[d, "\eta_{\gamma_* D}"] \\ \nabla_B|X| \arrow[r, "\nabla_B\phi"'] & \nabla_B|D|. \end{tikzcd}$$

But $FD \cong \gamma_* D$ so $\eta_{FD} \cong \eta_{\gamma_* D}$; hence these two pullbacks are isomorphic and $FX \cong \gamma_* X$.

To show naturality, suppose we are given $f : Y \to X$. Then we have

$$\begin{tikzcd} FY \arrow[r, "Ff"] \arrow[d, "\eta_{FY}"'] & FX \arrow[r, "F\phi"] \arrow[d, "\eta_{FX}"'] \arrow[dr, phantom, "\lrcorner", very near start] & FD \arrow[d, "\eta_{FD}"] \\ \nabla_B|Y| \arrow[r, "\nabla_B f"'] & \nabla_B|X| \arrow[r, "\nabla_B\phi"'] & \nabla_B|D| \end{tikzcd}$$

and a similar diagram for γ_*. Now $\eta_{FY} \cong \eta_{\gamma_* Y}$ and $\eta_{FD} \cong \eta_{\gamma_* D}$ under the canonical isomorphisms, and η_{FD} is mono, so we have $\eta_{FX}(Ff) \cong \eta_{\gamma_* X}(\gamma_* f)$ and $(F\phi)(Ff) \cong (\gamma_*\phi)(\gamma_* f)$. But also $\eta_{FX} \cong \eta_{\gamma_* X}$ and $F\phi \cong \gamma_*\phi$; hence $Ff \cong \gamma_* f$ by uniqueness of the mediating morphism $FY \to FX$. Hence we have $F \cong \gamma_*$. □

We now wish to show that the isomorphism $F \cong (\mathbf{App}(F))_*$ is natural in F. Suppose that F, G are two $\nabla\Gamma$-functors and $\alpha : F \to G$ is a natural transformation. Let us write $\gamma = \mathbf{App}(F)$ and $\delta = \mathbf{App}(G)$. By Proposition 2.2.13 there is an applicative transformation $\zeta : \gamma \preceq \delta$. We need to show that $\zeta_* \cong \alpha$ via the canonical isomorphisms $\gamma_* \cong F$, $\delta_* \cong G$.

Lemma 2.2.18 *$\alpha_E \cong \zeta_{*E}$ and $\alpha_D \cong \zeta_{*D}$ under the above isomorphisms.*

PROOF Let $j : E \to R \times \nabla_A|D|$ be the obvious monomorphism (tracked by $\lambda^* x.\langle x, i\rangle$). By naturality the following squares commute:

$$\begin{array}{ccc} FE & \xrightarrow{Fj} & F(R \times \nabla_A|D|) \\ \downarrow{\scriptstyle \alpha_E} & & \downarrow{\scriptstyle \alpha_{R\times\nabla|D|}} \\ GE & \xrightarrow[Gj]{} & G(R \times \nabla_A|D|) \end{array} \qquad \begin{array}{ccc} \gamma_* E & \xrightarrow{\gamma_* j} & \gamma_*(R \times \nabla_A|D|) \\ \downarrow{\scriptstyle \zeta_{*E}} & & \downarrow{\scriptstyle \zeta_{*R\times\nabla|D|}} \\ \delta_* E & \xrightarrow[\delta_* j]{} & \delta_*(R \times \nabla_A|D|) \end{array}$$

But $\zeta_{*R} \cong \alpha_R$ by definition, and $\zeta_{*\nabla|D|} \cong \alpha_{\nabla|D|} \cong \mathrm{id}_{\nabla|D|}$ by Lemma 2.2.12. So under the canonical isomorphisms we have

$$Fj \;\cong\; \gamma_* j, \qquad Gj \;\cong\; \delta_* j, \qquad \alpha_{R\times\nabla|D|} \;\cong\; \zeta_{*R\times\nabla|D|}.$$

Hence also $\alpha_E \cong \zeta_{*E}$, since Gj is mono.

Next note that the following squares commute by naturality:

$$\begin{array}{ccc} FE & \xrightarrow{(\nabla\pi')\eta_{FE}} & \nabla_B|D| \\ \downarrow{\scriptstyle \alpha_E} & & \downarrow{\scriptstyle \mathrm{id}} \\ GE & \xrightarrow[(\nabla\pi')\eta_{GE}]{} & \nabla_B|D| \end{array} \qquad \begin{array}{ccc} \gamma_* E & \xrightarrow{(\nabla\pi')\eta_{\gamma_* E}} & \nabla_B|D| \\ \downarrow{\scriptstyle \zeta_{*E}} & & \downarrow{\scriptstyle \mathrm{id}} \\ \delta_* E & \xrightarrow[(\nabla\pi')\eta_{\delta_* E}]{} & \nabla_B|D|. \end{array}$$

Moreover under the canonical isomorphisms we have

$$\alpha_E \;\cong\; \zeta_{*E}, \qquad (\nabla\pi')\eta_{FE} \;\cong\; (\nabla\pi')\eta_{\gamma_* E}, \qquad (\nabla\pi')\eta_{GE} \;\cong\; (\nabla\pi')\eta_{\delta_* E}.$$

Hence, by taking image factorizations of the top and bottom of each of the above squares, we obtain $\alpha_D \cong \zeta_{*D}$ as required. □

Proposition 2.2.19 *For any object X of* **Ass**(A)*, we have $\alpha_X \cong \zeta_{*X}$ under the canonical isomorphisms.*

Proof Let $\phi : X \to D$ be a cartesian morphism. The isomorphisms identify the two pullback squares in the proof of Proposition 2.2.17, as well as a similar pair for G and δ_*. Moreover, by naturality of α we have the equations $\eta_{FX} = \eta_{GX}\alpha_X$ and $\alpha_D(F\phi) = (G\phi)\alpha_X$; and by naturality of ζ_* we have $\eta_{\gamma_* X} = \eta_{\delta_* X}\zeta_{*X}$ and $\zeta_{*D}(\gamma_*\phi) = (\delta_*\phi)\zeta_{*X}$. But $\alpha_D \cong \zeta_{*D}$ by the Lemma, and so $\alpha_X \cong \zeta_{*X}$ by uniqueness of the mediating morphism $FX \to GX$. □

Propositions 2.2.17 and 2.2.19 together say that $\mathbf{Ass} \circ \mathbf{App}$ is equivalent to the identity functor on $\nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)]$. The proof of the following theorem is now complete:

Theorem 2.2.20 *The 2-functor* $\mathbf{Ass} : \mathbf{PCA} \to \nabla\Gamma\mathbf{Reg}$ *is* locally an equivalence, *that is, for any* A, B *the functor*

$$\mathbf{Ass} : \ \mathbf{PCA}\,[A, B] \longrightarrow \nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)]$$

is part of an equivalence of categories. $\square$

Corollary 2.2.21 *If* A, B *are PCAs then* $\mathbf{Ass}(A) \simeq \mathbf{Ass}(B)$ *iff there are applicative morphisms* $\gamma : A \longrightarrow\!\!\!\!\!\!\!+\;\; B$ *and* $\delta : B \longrightarrow\!\!\!\!\!\!\!+\;\; A$ *such that* $\delta\gamma \sim \mathrm{id}_A$ *and* $\gamma\delta \sim \mathrm{id}_B$. *(In this case we say that* A, B *are* equivalent*, and write* $A \simeq B$.*)*

Proof Given γ, δ as above, it is clear that γ_*, δ_* constitute an equivalence of categories. Conversely, if $F : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ and $G : \mathbf{Ass}(B) \to \mathbf{Ass}(A)$ are the two halves of an equivalence then clearly F, G are exact functors and $\Gamma_B F \cong \Gamma_A$, $\Gamma_A G \cong \Gamma_B$. Hence, by taking right adjoints, we also have $G\nabla_B \cong \nabla_A$, $F\nabla_A \cong \nabla_B$. So F, G are both $\nabla\Gamma$-functors. It now follows easily from Theorem 2.2.20 that $\mathbf{App}(F)$, $\mathbf{App}(G)$ constitute an equivalence of PCAs. $\square$

2.3 Functors between realizability toposes

We now extend the results of the last section to realizability toposes. First we need a technical result which allows us to state Theorem 2.2.20 in a slightly sharper form. We can then obtain the corresponding theorem for $\mathbf{RT}$ cheaply via the characterization of $\mathbf{RT}(A)$ in Section 1.3.

We start by introducing another definition in the spirit of 1.4.1 and 1.4.2:

Definition 2.3.1 (∇-category) *A* ∇-category *is a regular category* $\mathcal{C}$ *equipped with an exact functor* $\nabla : \mathbf{Set} \to \mathcal{C}$. *A* ∇-functor *between* ∇*-categories* $(\mathcal{C}, \nabla_C)$ *and* $(\mathcal{D}, \nabla_D)$ *is an exact functor* $F : \mathcal{C} \to \mathcal{D}$ *such that* $F\nabla_C \cong \nabla_D$. *We write* $\Gamma\mathbf{Reg}$ *for the 2-category of* Γ*-categories,* Γ*-functors and natural transformations.*

Note that if $\mathcal{C}, \mathcal{D}$ are $\nabla\Gamma$-categories, then $F : \mathcal{C} \to \mathcal{D}$ is a $\nabla\Gamma$ functor iff it is both a ∇-functor and a Γ-functor.

The result we need is that every ∇-functor $\mathbf{Ass}(A) \to \mathbf{Ass}(B)$ is a Γ-functor. This fact follows immediately from the following proposition:

Proposition 2.3.2 *If $F : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ is a ∇-functor, then $\Gamma_B F \cong \Gamma_A$.*

PROOF Given an object X of $\mathbf{Ass}(A)$, an element of $\Gamma_A X$ is a morphism $x : 1 \to X$. Define $\alpha_X : \Gamma_A X \to \Gamma_B F X$ by $\alpha_X(x) = Fx$. Then α_X is natural in X, since given $f : X \to Y$ we have $\alpha_Y((\Gamma_A f)x) = F(fx) = (Ff)(Fx) = \Gamma_B(Ff)(\alpha_X x)$. It suffices to show that each α_X is a bijection.

To see that α_X is injective, suppose $Fx = Fx'$ for $x, x' : 1 \to X$. Since η_1 is iso, naturality of η gives $\nabla_A \Gamma_A x \cong \eta_X x$, and similarly for x'. So we have $\nabla_B \Gamma_A x \cong F\nabla_A \Gamma_A x \cong F(\eta_X x) \cong (F\eta_X)(Fx)$ and similarly for x'; hence $\nabla_B \Gamma_A x = \nabla_B \Gamma_A x'$. But ∇_B and Γ_A are faithful, so $x = x'$.

To see that α_X is surjective, given $y \in \Gamma_B F X$ we have

$$(F\eta_X)y : \ 1 \longrightarrow F\nabla_A \Gamma_A X \ \cong \ \nabla_B \Gamma_A X.$$

Since ∇_B is full we obtain a morphism $1 \to \Gamma_A X$ and hence a morphism $x : 1 \to X$. But now $(F\eta_X)(Fx) = F(\eta_X x) \cong F\nabla_A \Gamma_A x \cong \nabla_B \Gamma_A x \cong (F\eta_X)y$. Since η_X is mono and F preserves monos, we have $Fx = y$. $\square$

For the record, we note that the following converse to Proposition 2.3.2 is also true. The proof we give is non-constructive.

Proposition 2.3.3 *If $F : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ is a Γ-functor, then $F\nabla_A \cong \nabla_B$.*

PROOF Given a set X, take Y such that $\mathrm{card}\, Y \geq \mathrm{card}\, X \times \mathrm{card}\, A$, and consider the assembly $F\nabla_A Y$. We have $\Gamma_B F \nabla_A(Y) \cong \Gamma_A \nabla_A Y \cong Y$, and so there are $b \in B$ and $Z \subseteq |F\nabla_A Y|$ such that $\mathrm{card}\, Z \geq \mathrm{card}\, X$ and $b \in \|z\|$ for all $z \in Z$. Let Y' be the subset of Y corresponding to Z under the above bijection. Choose functions $X \xrightarrow{f} Y \xrightarrow{g} X$ such that $\mathrm{Im}\, f \subseteq Y'$ and $gf = \mathrm{id}$. Suppose $r \in B$ tracks $F\nabla_A(g)$. Then clearly $rb \in \|x\|$ for all $x \in |F\nabla_A X|$. Thus $F\nabla_A X$ is uniform.

But $\Gamma_B F \nabla_A X \cong X$, and so $F\nabla_A X \cong \nabla_B X$. Naturality in X follows easily from naturality of $\Gamma_B F \nabla_A \cong \mathrm{id}$. $\Box$

We can thus give two alternative formulations of Theorem 2.2.20 :

Theorem 2.3.4 *The 2-functor* $\mathbf{Ass} : \mathbf{PCA} \to \nabla\mathbf{Reg}$ *[resp.* $\mathbf{PCA} \to \Gamma\mathbf{Reg}$*] is locally an equivalence.* $\Box$

It is now easy to obtain the analogous result for $\mathbf{RT}$. We regard $\mathbf{RT}(A)$ as a $\nabla\Gamma$-category, equipped with exact functors $\nabla' = I\nabla$ and $\Gamma' = \mathrm{Hom}(1, -)$ (see Proposition 1.3.9).

Lemma 2.3.5 *For any A, B, the canonical functor*

$$(I_B \circ -) : \ \nabla\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)] \longrightarrow \nabla\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{RT}(B)].$$

is part of an equivalence of categories.

PROOF Clearly $(I \circ -)$ is faithful since I is faithful. We now show that it is essentially surjective. Given a ∇-functor $G : \mathbf{Ass}(A) \to \mathbf{RT}(B)$, for any object X of $\mathbf{Ass}(A)$ we have a mono $X \rightarrowtail \nabla'_A \Gamma'_A X$ and hence a mono $GX \rightarrowtail \nabla'_B \Gamma'_A X$. But by the remark following Proposition 1.3.5 we have a regular epi $IY \twoheadrightarrow GX$ for some Y in $\mathbf{Ass}(B)$. Now I_B is full, so the composite $IY \twoheadrightarrow GX \rightarrowtail \nabla'_B \Gamma'_A X$ is equal to Ig for some $g : Y \to \nabla_B \Gamma'_A X$ in $\mathbf{Ass}(B)$. Let HX be the object appearing in the image factorization of g. Since I preserves image factorizations, we have $GX \cong IHX$. Now since I is full, for any morphism $f : X \to Y$ there is a morphism $Hf : HX \to HY$ such that IHf is identified with Gf. So we obtain a functor $H : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ such that $G \cong IH$. It is clear that H is an exact ∇-functor.

It is now easy to see that $(I \circ -)$ is also full: given $G, G' : \mathbf{Ass}(A) \to \mathbf{RT}(B)$, any natural transformation $G \to G'$ restricts to a natural transformation $H \to H'$ since I is full. Thus $(I \circ -)$ is part of an equivalence. $\Box$

Proposition 2.3.6 *There is a canonical equivalence of categories*

$$\nabla\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)] \ \simeq \ \nabla\mathbf{Reg}\,[\mathbf{RT}(A), \mathbf{RT}(B)].$$

In particular, for every ∇-functor $F : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$ there is an essentially unique ∇-functor $\hat{F} : \mathbf{RT}(A) \to \mathbf{RT}(B)$ such that $\hat{F}I_A \cong I_B F$; and for every natural transformation $\alpha : F \to G$ between ∇-functors there is a unique natural transformation $\hat{\alpha} : \hat{F} \to \hat{G}$ such that $\hat{\alpha}I_A \cong I_B\alpha$ under this isomorphism.

Proof By Theorem 1.3.6, the functor $(- \circ I_A)$ is part of an equivalence

$$\mathbf{Ex}[\mathbf{Ass}(A), \mathbf{RT}(B)] \;\simeq\; \mathbf{Ex}[\mathbf{RT}(A), \mathbf{RT}(B)].$$

Moreover, if we have an exact functor $\tilde{F} : \mathbf{RT}(A) \to \mathbf{RT}(B)$ and $F \cong \tilde{F}I_A$ then $\tilde{F}\nabla'_A \cong F\nabla_A$; hence $\tilde{F}$ is a ∇-functor iff F is a ∇-functor. So we have an equivalence

$$\nabla\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{RT}(B)] \;\simeq\; \nabla\mathbf{Reg}\,[\mathbf{RT}(A), \mathbf{RT}(B)].$$

Combining this with the Lemma we obtain the desired equivalence. □

Let us define $\mathbf{RT} : \mathbf{PCA} \to \nabla\mathbf{Reg}$ by setting $\mathbf{RT}(\gamma) = \hat{\gamma_*}$ for γ an applicative morphism, and $\mathbf{RT}(\zeta) = \hat{\zeta_*}$ for ζ an applicative transformation. (From now on we will write $\hat{\gamma}$ and $\hat{\zeta}$ instead of $\hat{\gamma_*}$ and $\hat{\zeta_*}$.) Then it is trivial to check using the above proposition that $\mathbf{RT}$ is a 2-functor; moreover, combining the above proposition with Theorem 2.3.4 we obtain

Theorem 2.3.7 *The 2-functor* $\mathbf{RT} : \mathbf{PCA} \to \nabla\mathbf{Reg}$ *is locally an equivalence.* □

Corollary 2.3.8 *If A, B are PCAs, we have* $\mathbf{RT}(A) \simeq \mathbf{RT}(B)$ *iff* $A \simeq B$.

Proof Similar to Corollary 2.2.21. □

Remarks 2.3.9 (i) If $\gamma : A \longrightarrow\!\!\!\!\!\!\mapsto B$ is an applicative morphism, it is easy to check that $\hat{\gamma} : \mathbf{RT}(A) \to \mathbf{RT}(B)$ is isomorphic to the functor described explicitly by

$$\hat{\gamma}(X, \rho) = (\gamma_* X, \gamma_* \rho),$$
$$\hat{\gamma}\left((X, \rho) \xrightarrow{\tau} (Y, \sigma)\right) = \gamma_*(T \overset{\tau}{\rightarrowtail} X \times Y).$$

(ii) It can easily be shown that the analogue of Proposition 2.3.2 holds for $\mathbf{RT}$: every ∇-functor $\mathbf{RT}(A) \to \mathbf{RT}(B)$ is a Γ-functor. Moreover, it seems plausible that the converse is also true (cf. Proposition 2.3.3), though we have not been able to settle this question.

2.4 Some special morphisms

We now turn to investigate certain properties of applicative morphisms, and how these are reflected in properties of the corresponding functors. One might expect that *categorical* properties of applicative morphisms, such as being a mono or an epi in **PCA** or **PCA**/$\sim$, would play some special role. However, this appears not to be the case, perhaps because of the poor categorical structure of **PCA**. (In fact, it is surprisingly difficult even to identify the monos in **PCA** or **PCA**/$\sim$!) It thus seems more fruitful to consider concrete properties of morphisms. In this section we examine three natural classes of applicative morphisms: *discrete, projective* and *decidable* morphisms, and give categorical characterizations of the functors they give rise to.

2.4.1 Discrete morphisms

In Section 2.1 we remarked that an applicative morphism $A \longrightarrow\!\!\!\!\!\!\!+\;\, B$ may be thought of informally as an "implementation" of A in B. It seems natural to consider the class of morphisms for which no element of B may implement two different elements of A.

Definition 2.4.1 (Discrete morphism) *An applicative morphism $\gamma : A \longrightarrow\!\!\!\!\!\!\!+\;\, B$ is called* discrete *if*

$$\gamma(a, b) \wedge \gamma(a', b) \implies a = a'$$

for all $a, a' \in A$, $b \in B$. We write **dPCA** *for the 2-category of PCAs, discrete morphisms and applicative transformations.*

We now characterize the functors γ_*, $\hat{\gamma}$ arising from discrete morphisms. For this we need the following notion (introduced in [44]):

Definition 2.4.2 (Discrete object) *An object X of* **Ass**(A) *or* **RT**(A) *is* discrete *if the canonical morphism $X \to X^{\nabla 2}$ (or $X \to X^{\nabla' 2}$) is iso.*

Clearly X is discrete in $\mathbf{Ass}(A)$ iff IX is discrete in $\mathbf{RT}(A)$. Note that if X is discrete then in particular every morphism $\nabla 2 \to X$ (or $\nabla' 2 \to X$) is constant (i.e. factors through 1). Discrete objects are extensively studied in [44], where it is shown that they form an "internally complete" subcategory of $\mathbf{RT}(A)$. In that paper the following characterization of discrete objects was given:

Proposition 2.4.3 *(i) An object X of* $\mathbf{Ass}(A)$ *is discrete iff it is a modest set.*

(ii) An object Y of $\mathbf{RT}(A)$ *is discrete iff there exist a modest set X and a regular epi $IX \twoheadrightarrow Y$.*

Proof (i) Suppose $2 = \{0, 1\}$; then $\|0 \in \nabla 2\| = \|1 \in \nabla 2\| = A$. If X is modest, then for any $f : \nabla 2 \to X$ tracked by r say, we have $ri \in \|x\|_X$ for a unique x and so $f0 = f1 = x$. Hence the morphism $X^{\nabla 2} \to X$ given by evaluation at 0 is inverse to the canonical morphism $X \to X^{\nabla 2}$, so X is discrete. Conversely, if X is not modest, take $x \neq x' \in |X|$ with $a \in \|x\| \cap \|x'\|$. Then there is a morphism $f : \nabla 2 \to X$ tracked by ka such that $f0 = x, f1 = x'$. Hence the morphism $|X| \to |X^{\nabla 2}|$ is not surjective, so X is not discrete.

(ii) See [44, Section 6]. □

The notion of discrete object may thus be seen as generalizing the notion of modest set. We may now identify the functors arising from discrete morphisms:

Theorem 2.4.4 *The following are equivalent for an applicative morphism $\gamma : A \longrightarrow\!\!\!\!\!\!\mapsto B$:*

(i) γ is discrete;

(ii) γ_* preserves modest sets;

(iii) $\hat{\gamma}$ preserves discrete objects.

Proof (i) $\Rightarrow$ (ii): Suppose γ is discrete and X is modest in $\mathbf{Ass}(A)$. Given $x \neq x' \in |X|$ we have $\|x\| \cap \|x'\| = \emptyset$ so $\gamma^+\|x\| \cap \gamma^+\|x'\| = \emptyset$, thus $\gamma_* X$ is modest.

(ii) $\Rightarrow$ (iii): Suppose γ_* preserves modest sets and Y is discrete in $\mathbf{RT}(A)$. Take X modest and $e : IX \twoheadrightarrow Y$ regular epi by Proposition 2.4.3(ii). Then $\gamma_* X$ is modest and $\hat{\gamma}(e) : I(\gamma_* X) \to \hat{\gamma} Y$ is regular epi since $\hat{\gamma}$ is exact. Hence $\hat{\gamma} Y$ is discrete.

(iii) $\Rightarrow$ (ii): Trivial, since we know that $I_B \gamma_* \cong \hat{\gamma} I_A$.

(ii) $\Rightarrow$ (i): The object of realizers R in $\mathbf{Ass}(A)$ is modest (see Definition 2.2.8); and if $\gamma_* R$ is modest then clearly γ is discrete. $\Box$

Using discrete morphisms we can formulate the analogue of Theorems 2.2.20 and 2.3.7 for $\mathbf{Mod}$. First note that the functor $\Gamma : \mathbf{Mod}(A) \to \mathbf{Set}$ is exact, so $\mathbf{Mod}(A)$ is a Γ-category in the sense of Definition 1.4.1. Next, if $\gamma : A \longrightarrow\!\!\!\!\!\!+\; B$ is a discrete morphism then it is clear from the above proposition that γ_* restricts to a Γ-functor $\mathbf{Mod}(\gamma) : \mathbf{Mod}(A) \to \mathbf{Mod}(B)$. Moreover, if $\zeta : \gamma \preceq \delta$ then ζ_* restricts to a natural transformation $\mathbf{Mod}(\zeta) : \mathbf{Mod}(\gamma) \to \mathbf{Mod}(\delta)$ since $\mathbf{Mod}(B)$ is full in $\mathbf{Ass}(B)$. Thus we have a 2-functor $\mathbf{Mod} : \mathbf{dPCA} \to \Gamma\mathbf{Reg}$. We can now prove the following without much effort:

Theorem 2.4.5 *The 2-functor* $\mathbf{Mod}$ *is locally an equivalence.*

Proof By Theorems 1.4.5 and 2.2.20 we have equivalences

$$\Gamma\mathbf{Reg}\,[\mathbf{Mod}(A), \mathbf{Ass}(B)] \;\simeq\; \nabla\Gamma\mathbf{Reg}\,[\mathbf{Ass}(A), \mathbf{Ass}(B)] \;\simeq\; \mathbf{PCA}\,[A, B].$$

But Proposition 2.4.4 shows that a Γ-functor $\mathbf{Mod}(A) \to \mathbf{Ass}(B)$ factors through I_B iff the corresponding morphisms $A \longrightarrow\!\!\!\!\!\!+\; B$ is discrete. So the above equivalence restricts to an equivalence between the full subcategories:

$$\Gamma\mathbf{Reg}\,[\mathbf{Mod}(A), \mathbf{Mod}(B)] \;\simeq\; \mathbf{dPCA}\,[A, B]. \qquad \Box$$

Corollary 2.4.6 *If A, B are PCAs, we have* $\mathbf{Mod}(A) \simeq \mathbf{Mod}(B)$ *iff* $A \simeq B$.

Proof It is not hard to show that if γ, δ form an equivalence of PCAs then both γ, δ are discrete; hence we obtain an equivalence $\mathbf{Mod}(A) \simeq \mathbf{Mod}(B)$. For the converse, note that if F, G form an equivalence $\mathbf{Mod}(A) \simeq \mathbf{Mod}(B)$ then both F, G are Γ-functors; hence by the Theorem we have $A \simeq B$. $\Box$

2.4.2 Projective morphisms

When working with applicative morphisms, it is convenient to deal wherever possible with *single-valued* morphisms $A \longrightarrow\!\!\!\!\!\!+\; B$—that is, with functions $A \to B$—

since these are much easier to calculate with than general applicative morphisms. We therefore make the following definition:

Definition 2.4.7 (Projective morphism) *A morphism $\gamma : A \multimap B$ is* projective *if there is a morphism $\gamma' \sim \gamma$ such that γ' is single-valued (i.e. for each $a \in A$ the set $\gamma'(a)$ is a singleton).*

It turns out that the functors arising from projective morphisms can be characterized as those that preserve *projective* objects:

Definition 2.4.8 (Projective object) *An object X in any category is called* projective *(or sometimes* regular projective*) if for any regular epimorphism $e : Y \twoheadrightarrow Z$ and any morphism $f : X \to Z$ there is a morphism $g : X \to Y$ such that $f = eg$.*

Proposition 2.4.9 *(i) An object X is projective in $\mathbf{Ass}(A)$ iff IX is projective in $\mathbf{RT}(A)$.*

(ii) If Y is projective in $\mathbf{RT}(A)$ then $Y \cong IX$ for some X in $\mathbf{Ass}(A)$.

Proof (i) The right-to-left implication is trivial since I is full. Conversely, suppose X is projective in $\mathbf{Ass}(A)$, $e : Y \twoheadrightarrow Z$ is a regular epi in $\mathbf{RT}(A)$, and we are given $f : IX \to Z$. Let Z' in $\mathbf{Ass}(A)$ be such that there is a regular epi $h_Z : IZ' \twoheadrightarrow Z$; let Y'' be the object obtained by pulling back h_Z along e; and let Y' in $\mathbf{Ass}(A)$ be such that there is a regular epi $IY' \twoheadrightarrow Y''$. Since pullbacks preserve regular epis, we have a square

$$\begin{tikzcd} IY' \arrow[r, two heads, "Ie'"] \arrow[d, two heads, "h_Y"'] & IZ' \arrow[d, two heads, "h_Z"] \\ Y \arrow[r, two heads, "e"'] & Z. \end{tikzcd}$$

Now let $X'' \to IX$ be the pullback of h_Z along f, and take X' in $\mathbf{Ass}(A)$ such that there is a regular epi $IX' \twoheadrightarrow X''$. We now have a square

$$\begin{tikzcd} IX' \arrow[r, "If'"] \arrow[d, "Ih_X"'] & IZ' \arrow[d, "h_Z"] \\ IX \arrow[r, "f"'] & Z. \end{tikzcd}$$

Clearly I reflects regular epis, and so h_X is regular epi. Since X is projective, we obtain $k : X \to X'$ such that $h_X k = \mathrm{id}_X$. Now we have $f'k : X \to Z'$ and $e' : Y' \twoheadrightarrow Z'$ regular epi, so we have $j : X \to Y'$ such that $e'j = f'k$. Let $g = h_Y(Ij)$. Then $eg = eh_Y(Ij) = h_Z(Ie')(Ij) = h_Z(If')(Ik) = f(Ih_X)(Ik) = f$ as required.

(ii) Suppose Y is projective in $\mathbf{RT}(A)$; take X in $\mathbf{Ass}(A)$ such that there is a regular epi $h : IX \twoheadrightarrow Y$. Take $k : Y \to IX$ such that $hk = \mathrm{id}_Y$. Then k is mono. Now suppose $kh = If : IX \to IX$, and let Z be the object appearing in the image factorization of f. Then IZ appears in the image factorization of kh, so $Y \cong IZ$ by uniqueness of factorizations. □

It is now easy to characterize the projective objects concretely:

Proposition 2.4.10 *X is projective in $\mathbf{Ass}(A)$ iff there is $X' \cong X$ such that for each $x \in |X'|$ the set $\|x\|_{X'}$ is a singleton. (We say that such an object X' has the* singleton property*.)*

Proof Suppose X' satisfies the singleton property, $e : Y \twoheadrightarrow Z$ is regular epi and we are given $f : X' \to Z$ tracked by $r \in A$. By the proof of Proposition 1.2.10, we may assume w.l.o.g. that e is tracked by i and that $\|z\|_Z = \bigcup_{ey=z} \|y\|_Y$ for all $z \in |Z|$. We define a function $g : |X'| \to |Y|$ as follows: for each $x \in |X'|$ let a_x be the unique element of $\|x\|_{X'}$, and let gx be some element of $|Y|$ such that $e(gx) = fx$ and $ra_x \in \|gx\|_Y$ (here we use the axiom of choice in the meta-theory). Clearly $g : X' \to Y$ is tracked by r and $eg = f$. Thus X' is projective, and so if $X \cong X'$ then X is projective.

Conversely, suppose X is projective. As in Definition 1.4.6, let X_0 be the assembly defined by

$$|X_0| = \{(x,a) \mid x \in |X|,\ a \in \|x\|\}, \qquad \|(x,a) \in X_0\| = \{a\}$$

and let $e : X_0 \to X$ be the obvious projection tracked by i. Clearly e is regular epi, and so there is a morphism $g : X \to X_0$ such that $eg = \mathrm{id}_X$. Let $X' \rightarrowtail X_0$ be the subobject defined by

$$|X'| = \mathrm{Im}\ g, \qquad \|x\|_{X'} = x_{X_0}.$$

Then clearly we have morphisms $e' : X' \to X$ and $g' : X \to X'$ such that $e'g' = \mathrm{id}_X$ and g' is a bijection on the underlying sets. Hence $X \cong X'$. But X_0 has the singleton property, hence so does X'. □

Remark 2.4.11 Note that if X is a modest set then so is X_0, and if $X' \cong X$ then X' is also modest. Thus the above proposition also holds with $\mathbf{Ass}(A)$ replaced by $\mathbf{Mod}(A)$. It follows that an object X is projective in $\mathbf{Mod}(A)$ iff JX is projective in $\mathbf{Ass}(A)$.

It is now easy to characterize the functors arising from projective morphisms:

Theorem 2.4.12 *The following are equivalent for an applicative morphism $\gamma : A \longrightarrow B$:*

(i) γ is projective;

(ii) γ_* preserves projectives;

(iii) $\hat{\gamma}$ preserves projectives.

Proof (i) ⇒ (ii): Suppose $\gamma \sim \gamma'$ where γ' is single-valued, and X is projective in $\mathbf{Ass}(A)$. Take $X' \cong X$ where X' has the singleton property. Now $\gamma_* X \cong \gamma'_* X \cong \gamma'_* X'$, and clearly $\gamma'_* X'$ has the singleton property. Thus $\gamma_* X$ is projective.

(ii) ⇒ (i): Let R be the object of realizers in $\mathbf{Ass}(A)$; then R has the singleton property and so is projective. Suppose γ_* preserves projectives; let $S = \gamma_* R$ and take $S' \cong S$ such that S' has the singleton property. We may assume that $|S| = |S'| = A$. By definition of S we have $\gamma(a) = \|a \in S\|$ for each $a \in A$. Now let γ' be the relation defined by $\gamma'(a) = \|a \in S'\|$. Suppose r realizes γ, and p, q realize the isomorphisms $S \to S'$, $S' \to S$ respectively. Then $\gamma' : A \multimap B$ is an applicative morphism as it is realized by $\lambda^* xy.p(r(qx)(qy))$; moreover p realizes $\gamma \preceq \gamma'$ and q realizes $\gamma' \preceq \gamma$. So we have $\gamma \sim \gamma'$ where γ' is single-valued. Thus γ is projective.

(ii) $\Leftrightarrow$ (iii) is immediate from Proposition 2.4.9. $\square$

Remark 2.4.13 It is interesting to note that an applicative morphism γ is projective iff the object $\gamma_* R$ is projective, and discrete iff $\gamma_* R$ is discrete.

2.4.3 Decidable morphisms

The properties of being discrete or single-valued were purely "set-theoretic" properties of an applicative morphism $A \multimap B$—that is, they were defined without reference to the application operation in A or B. We now turn to consider a more "recursion-theoretic" property of applicative morphisms.

Definition 2.4.14 (Decidable morphism) *A morphism $\gamma : A \multimap B$ is* decidable *if there is an element $d \in B$ (called a* decider *for γ) such that for all $b \in B$ we have*

$$b \in \gamma(true_A) \implies db = true_B, \qquad b \in \gamma(false_A) \implies db = false_B.$$

Remarks 2.4.15 (i) Note that there is always an element $c \in B$ such that $c\ true_B \in \gamma(true_A)$, $c\ false_B \in \gamma(false_A)$ (take $c = \lambda^* x.\ if\ x\ b\ b'$ where $b \in \gamma(true_A)$, $b' \in \gamma(false_A)$). Thus γ is decidable iff the booleans in B are "equivalent" to the image of the booleans in A.

(ii) It is easy to check that PCAs and decidable morphisms form a category.

An equivalent definition of decidable morphisms can be given involving a similar criterion for the numerals in a PCA:

Proposition 2.4.16 *An applicative morphism $\gamma : A \to B$ is decidable iff there exists $f \in B$ such that for all $n \in N$ we have*

$$b \in \gamma(\overline{n}_A) \implies fb = \overline{n}_B.$$

PROOF Suppose $r \in B$ realizes γ. First suppose we have f as above. Let $a = \lambda^* x.\, \mathit{if}_A\ x\ \overline{0}\ \overline{1}$, take $a' \in \gamma(a)$, and let $d = \lambda^* x.\, \mathit{iszero}_B\, (f(ra'x))$. Then for $b \in \gamma(\mathit{true}_A)$ we have $ra'b \in \gamma(a\ \mathit{true}_A) = \gamma(\overline{0}_A)$ and so $f(ra'b) = \overline{0}_B$, hence $db = \mathit{true}_B$. Similarly, if $b \in \gamma(\mathit{false}_A)$ then $db = \mathit{false}_B$. Thus γ is decidable.

Conversely, suppose d is a decider for γ. We will construct an element f using the fixed point combinator Z of Proposition 1.1.13. First take $\mathit{iszero}' \in \gamma(\mathit{iszero}_A)$, and let $\mathit{Iszero} = \lambda^* x.\, d\,(r\ \mathit{iszero}'\ x)$. Note that

$$b \in \gamma(\overline{0}) \implies \mathit{Iszero}\ b = \mathit{true}_B, \qquad b \in \gamma(\overline{n+1}) \implies \mathit{Iszero}\ b = \mathit{false}_B.$$

Next, take $\mathit{pred}' \in \gamma(\mathit{pred})$ and let $\mathit{Pred} = r\ \mathit{pred}'$. Then we have

$$b \in \gamma(\overline{0}) \implies \mathit{Pred}\, b \in \gamma(\overline{0}), \qquad b \in \gamma(\overline{n+1}) \implies \mathit{Pred}\, b \in \gamma(\overline{n}).$$

Informally we wish to define f by the following recursive definition:

"fun $f\ x\ =$ if $(\mathit{Iszero}\ x)$ then 0 else $\mathit{succ}(f(\mathit{Pred}\ x))$."

To achieve this in combinatory logic, we define

$$\mathit{next} \;=\; \lambda^* gx.\ \mathit{succ}_B\,(g\,(\mathit{Pred}\ x)i),$$
$$F \;=\; \lambda^* gx.\ \mathit{if}_B\,(\mathit{Iszero}\ x)\,(k\,\overline{0})\,(\mathit{next}\ g), \qquad f \;=\; \lambda^* x.\,(ZF)xi.$$

We show by induction on n that if $b \in \gamma(\overline{n})$ then $fb = \overline{n}$. First, if $b \in \gamma(\overline{0})$ then we have

$$\begin{aligned} fb &\simeq (ZF)bi \\ &\simeq F(ZF)bi \\ &\simeq \mathit{if}\,(\mathit{Iszero}\ b)\,(k\,\overline{0})\,(\mathit{next}\,(ZF))\,i \\ &\simeq \mathit{if}\ \mathit{true}\ (k\,\overline{0})\,(\mathit{next}\,(ZF))\,i \\ &\simeq k\ \overline{0}\ i \qquad \text{(since } \mathit{next}\,(ZF)\!\downarrow) \\ &\simeq \overline{0}. \end{aligned}$$

Next, if $b \in \gamma(\overline{n+1})$ and the result holds for n, then we have

$$\begin{aligned} fb &\simeq \mathit{if\,false}\,(k\,\overline{0})\,(\mathit{next}\,(ZF))\,i \\ &\simeq \mathit{next}\,(ZF)\,i \qquad (\text{since } k\,\overline{0}\downarrow) \\ &\simeq \mathit{succ}\,((ZF)(\mathit{Pred}\,b)i) \\ &\simeq \mathit{succ}\,(f(\mathit{Pred}\,b)) \\ &\simeq \mathit{succ}\,\overline{n} \qquad (\text{by induction hypothesis}) \\ &\simeq \overline{n+1}. \qquad \Box \end{aligned}$$

Remark 2.4.17 Again, there is always an element $g \in B$ such that $g\overline{n}_B \in \gamma(\overline{n}_A)$ for all n. So γ is decidable if the numerals in B are "equivalent" to the image under γ of the numerals in A.

One application of the above proposition is the following interesting property of the PCA K_1.

Proposition 2.4.18 *K_1 is initial in the category of PCAs and decidable applicative morphisms modulo $\sim$.*

PROOF Given any PCA, define $\kappa : K_1 \longrightarrow\!\!\!\!\!\!+\; A$ by $\kappa(n) = \{\overline{n}_A\}$. To see that κ is an applicative morphism, let $\varphi : \mathrm{N} \times \mathrm{N} \rightharpoonup \mathrm{N}$ be the partial recursive function defined by $\varphi(m_1, m_2) \simeq \{m_1\}(m_2)$. Then by Theorem 1.1.16 there exists $e \in B$ such that $e\,\overline{m_1}\,\overline{m_2} = \overline{n}$ whenever $\varphi(m_1, m_2) = n$. Clearly e is a realizer for κ. Moreover κ is decidable by Proposition 2.4.16, since if $a \in \kappa(\overline{n}_A)$ then $ia = \overline{n}_B$. For uniqueness, suppose $\kappa' : K_1 \longrightarrow\!\!\!\!\!\!+\; A$ is any other decidable morphism. By Proposition 2.4.16 there exists $f \in A$ such that if $a \in \kappa'(n)$ then $fa \in \kappa(n)$; and by Remark 2.4.17 there exists $g \in A$ such that if $a \in \kappa(n)$ then $ga \in \kappa'(n)$. Thus f, g realize $\kappa' \preceq \kappa \preceq \kappa'$ respectively, so $\kappa' \sim \kappa$. $\Box$

The functors arising from decidable morphisms enjoy many good properties, and can be characterized in a variety of ways. We first consider functors between categories of assemblies:

Theorem 2.4.19 *The following are equivalent for an applicative morphism $\gamma : A \longrightarrow\!\!\!\!\!\!+\; B$:*

(i) γ *is decidable;*

(ii) $\gamma_* 2 \cong 2$, *where* $2 \cong 1+1$*;*

(iii) γ_* *preserves finite sums;*

(iv) γ_* *preserves the natural number object.*

PROOF (i) $\Leftrightarrow$ (ii): In both $\mathbf{Ass}(A)$ and $\mathbf{Ass}(B)$ we may take 2 to be the object given by

$$|2| = \{0,1\}, \qquad \|0\| = \{true\}, \qquad \|1\| = \{false\}.$$

So it is clear by Remark 2.4.15(i) that γ is decidable iff the canonical morphism $2_B \to \gamma_* 2_A$ has an inverse. And it is also clear that if there is any isomorphism $2 \to \gamma_* 2$ then the canonical morphism is one.

(i) $\Rightarrow$ (iii): Suppose r realizes γ, d decides γ and c is as in Remark 2.4.15(i). Clearly $\gamma_* 0 \cong 0$ whether or not γ is decidable. So suppose X, Y are objects of $\mathbf{Ass}(A)$. From the description of sums in Proposition 1.2.7, we see that if $x \in |X|$ then

$$\begin{aligned} \|(0,x) \in \gamma_* X + \gamma_* Y\| &= \{\, \langle true_B, b \rangle \mid \ b \in \|x\|_{\gamma_* X} \}, \\ \|(0,x) \in \gamma_*(X+Y)\| &= \gamma^+\{\, \langle true_A, a \rangle \mid \ a \in \|x\|_X \}; \end{aligned}$$

and similarly if $y \in |Y|$. Hence the canonical morphism $\gamma_* X + \gamma_* Y \to \gamma_*(X+Y)$ is tracked by $\lambda^* x.\ r\,(r\, pair'\,(c\,(fst\ x)))(snd\ x)$, where $pair' \in \gamma(pair_A)$; and its inverse is tracked by $\lambda^* x.\ \langle d\,(r\ fst'\ x),\ r\ snd'\ x \rangle$, where $fst' \in \gamma(fst_A)$, $snd' \in \gamma(snd_A)$.

(iii) $\Rightarrow$ (ii) is trivial.

(i) $\Leftrightarrow$ (iv): By Proposition 2.4.16 and the remark following it, it is clear that γ is decidable iff the canonical morphism $N_B \to \gamma_* N_A$ has an inverse. □

We can also give a similar result for functors between realizability toposes. First we recall a general result due to Peter Freyd:

Lemma 2.4.20 *Suppose* $\mathcal{C}$, $\mathcal{D}$ *are categories with finite limits, finite colimits and natural number object, and suppose* $F : \mathcal{C} \to \mathcal{D}$ *is exact and preserves finite sums and the NNO. Then* F *preserves coequalizers, and hence all finite colimits.*

PROOF An easy adaptation of the argument in [27, Section 5.5]. □

Theorem 2.4.21 *The following are equivalent for an applicative morphism $\gamma : A \longrightarrow B$:*

(i) γ is decidable;

(ii) $\hat{\gamma}$ preserves finite sums;

(iii) $\hat{\gamma}$ preserves the natural number object;

(iv) $\hat{\gamma}$ preserves finite colimits.

PROOF (i) $\Rightarrow$ (ii): Clearly $\hat{\gamma}0 \cong 0$ whether or not γ is decidable. Given objects (X,ρ) and (Y,σ) of $\mathbf{RT}(A)$ where $\rho : R \rightarrowtail X \times X$ and $\sigma : S \rightarrowtail Y \times Y$, let τ be the composite morphism

$$R + S \xrightarrow{\rho+\sigma} (X \times X) + (Y \times Y) \longrightarrow (X+Y) \times (X+Y).$$

It is straightforward to check that τ is an equivalence relation on $X+Y$, and that $(X+Y,\tau)$ is the sum of (X,ρ) and (Y,σ) in $\mathbf{RT}(A)$. Now by Remark 2.3.9 we have $\hat{\gamma}(X+Y,\tau) \cong (\gamma_*(X+Y), \gamma_*\tau)$. But if γ is decidable then γ_* preserves sums, and so $\gamma_*(X+Y) \cong \gamma_*X + \gamma_*Y$ and $\gamma_*\tau$ is isomorphic to the composite

$$\gamma_*R + \gamma_*S \longrightarrow (\gamma_*X \times \gamma_*X) + (\gamma_*Y \times \gamma_*Y) \longrightarrow (\gamma_*X + \gamma_*Y) \times (\gamma_*X + \gamma_*Y).$$

Thus $\hat{\gamma}(X+Y,\tau) \cong \hat{\gamma}(X,\rho) + \hat{\gamma}(Y,\sigma)$. It is easy to check that $\hat{\gamma}$ also preserves the inclusions modulo this isomorphism.

(ii) $\Rightarrow$ (i): Clearly $\hat{\gamma}$ preserves finite sums then so does γ_*; hence γ is decidable by Theorem 2.4.19.

(i) $\Leftrightarrow$ (iii): Clearly $\hat{\gamma}$ preserves the NNO iff γ_* does, so both implications follow by Theorem 2.4.19.

(iv) $\Rightarrow$ (ii) is trivial.

(i) $\Rightarrow$ (iv): Suppose γ is decidable. We already know that $\hat{\gamma}$ preserves finite sums and the NNO, so by the Lemma $\hat{\gamma}$ preserves finite colimits. $\Box$

Combining this with Proposition 2.4.18 we obtain an attractive property of the effective topos.

Corollary 2.4.22 *Let A be any PCA. Then there is (up to natural isomorphism) a unique functor $\mathcal{E}\!f\!f \to \mathbf{RT}(A)$ that preserves finite limits and colimits and commutes with ∇.* $\Box$

2.5 Adjoint pairs of morphisms

We close this chapter with a discussion of *adjoint pairs* of applicative morphisms. The notion of an adjoint pair makes sense in any 2-category, though we give the definition just for **PCA**.

Definition 2.5.1 (Adjoint pair) *An* adjoint pair $(\gamma \dashv \delta) : A \to B$ *consists of a pair of applicative morphisms* $\delta : A \longrightarrow\!\!\!\!\!+\,B$, $\gamma : B \longrightarrow\!\!\!\!\!+\,A$ *such that* $\mathrm{id}_B \preceq \delta\gamma$ *and* $\gamma\delta \preceq \mathrm{id}_A$. *We say that* γ *is* left adjoint *to* δ, *or that* δ *is* right adjoint *to* γ.

Note that the "triangular identities" for the adjunction hold automatically as **PCA** is preorder-enriched.

The category of PCAs and adjoint pairs seems to have much better properties than **PCA** itself, although we will not pursue these here. In the next chapter we will see that there are many interesting examples of adjoint pairs in **PCA**. In fact, we will meet many examples of pairs for which three out of the four possible applicative transformations exist:

Definition 2.5.2 (Inclusion, retraction) *Suppose* $(\gamma \dashv \delta) : A \to B$ *is an adjoint pair. We say* (γ, δ) *is an* (applicative) inclusion *if* $\gamma\delta \sim \mathrm{id}_A$, *and an* (applicative) retraction *if* $\delta\gamma \sim \mathrm{id}_B$.

We now obtain some simple but powerful facts about adjoint pairs:

Theorem 2.5.3 *Suppose* $\delta : A \longrightarrow\!\!\!\!\!+\,B$ *and* $\gamma : B \longrightarrow\!\!\!\!\!+\,A$ *are applicative morphisms.*

(i) If $\gamma\delta \preceq \mathrm{id}_A$, *then* δ *is discrete and* γ *is decidable;*

(ii) If $\gamma \dashv \delta$, *then* γ *is also projective.*

Proof Suppose r realizes γ and t realizes $\gamma\delta \preceq \mathrm{id}_A$.

(i) To see that δ is discrete, suppose $b \in \delta(a_1) \cap \delta(a_2)$, and take $b' \in \gamma(b)$. Then $b' \in \gamma\delta(a_1) \cap \gamma\delta(a_2)$, so $a_1 = tb' = a_2$. To see that γ is decidable, take $true' \in \delta(true_A)$, $false' \in \delta(false_A)$. Let $e = \lambda^* x.\ if_B\ x\ true'\ false'$ and take $e' \in \gamma(e)$. Now if $a \in \gamma(true_B)$ then $re'a \in \gamma(true') \subseteq \gamma\delta(true_A)$, so $t(re'a) = true_A$. Similarly if $a \in \gamma(false_B)$ then $t(re'a) = false_A$. So $\lambda^* a.\, t(re'a)$ is a decider for γ.

(ii) Suppose u realizes $\mathrm{id}_B \preceq \delta\gamma$ and take $u' \in \gamma(u)$. For any $b \in B$ we have $ub \in \delta\gamma(b)$, so $ub \in \delta(c_b)$ for some $c_b \in \gamma(b)$. Moreover c_b is unique since δ is discrete. Define a total relation $\gamma' : B \longrightarrow\!\!\!\!\!\!\rightarrow A$ by $\gamma'(b) = \{c_b\}$. Then since $c_b \in \gamma(b)$, we have that $\gamma' \preceq \gamma$ is realized by i (note that $\preceq$ can be defined for total relations just as for applicative morphisms). Also if $a \in \gamma(b)$ then $ru'a \in \gamma(ub) \subseteq \gamma\delta(c_b)$, so $t(ru'a) = c_b$. So $\lambda^* a.\, t(ru'a)$ realizes $\gamma \preceq \gamma'$. Thus $\gamma' \sim \gamma$, so clearly γ' is an applicative morphism. Moreover γ' is single-valued, so γ is projective. □

Corollary 2.5.4 *If $(\gamma \dashv \delta)$ is a retraction, then both δ and γ are discrete and decidable, and γ is projective.* □

One application of Theorem 2.5.3 is to give an improved characterization of "equivalence" between PCAs:

Corollary 2.5.5 *Suppose $(\gamma, \delta) : A \leftrightarrows B$ form an equivalence of PCAs. Then there is an equivalence $(\gamma', \delta') : A \leftrightarrows B$ where both δ and γ are single-valued (i.e. are* functions*).*

Proof Both δ and γ are projective by Theorem 2.5.3. □

Functions between PCAs are much more manageable to work with than general relations, and it is this criterion for equivalence between PCAs that we will use in practice in the next chapter. For convenience we summarize our characterizations of equivalence here:

Theorem 2.5.6 *The following are equivalent for two PCAs A, B:*

(i) $A \simeq B$;

(ii) There is an equivalence $A \simeq B$ consisting of single-valued morphisms;

(iii) $\mathbf{Mod}(A) \simeq \mathbf{Mod}(B)$;

(iv) $\mathbf{Ass}(A) \simeq \mathbf{Ass}(B)$;

(v) $\mathbf{RT}(A) \simeq \mathbf{RT}(B)$.

Proof Combine Corollaries 2.2.21, 2.3.8, 2.4.6 and 2.5.5. □

We now turn our attention to the functors arising from adjoint pairs. The basic result is the following:

Proposition 2.5.7 *If $\delta : A \twoheadrightarrow B$, $\gamma : B \twoheadrightarrow A$ are applicative morphisms, then*

$$\gamma \dashv \delta \quad \textit{iff} \quad \gamma_* \dashv \delta_* \quad \textit{iff} \quad \hat{\gamma} \dashv \hat{\delta}.$$

PROOF Trivial by Theorems 2.2.20 and 2.3.7. □

The main interest seems to lie with the functors $\hat{\delta}, \hat{\gamma}$. Here we import a standard notion from topos theory (cf. [45]):

Definition 2.5.8 (Geometric morphism) *Suppose $\mathcal{E}, \mathcal{F}$ are toposes. A* geometric morphism *$(H, G) : \mathcal{E} \to \mathcal{F}$ consists of a pair of functors $G : \mathcal{E} \to \mathcal{F}$, $H : \mathcal{F} \to \mathcal{E}$ such that $H \dashv G$ and H preserves finite limits. We say (H, G) is a* geometric inclusion *if the counit $HG \to \mathrm{id}$ is an isomorphism. We will also say (H, G) is a* geometric retraction *if the unit $\mathrm{id} \to GH$ is an isomorphism.*

Geometric inclusions $\mathcal{E} \to \mathcal{F}$ are especially interesting since they correspond to *sheaf subtoposes* of $\mathcal{F}$, which in turn are precisely equivalent to *topologies* on $\mathcal{F}$ (see Chapter 3 of [45]). The notion of geometric retraction is less intrinsically interesting—note that the definition is stronger than the (more standard) definition of *geometric surjection* (see [45, Definition 4.11]).

We will call a geometric morphism (H, G) *exact* if both H and G are exact functors. Note that since $H \dashv G$ we know that G preserves existing limits and H preserves existing colimits; hence H is automatically exact, and (H, G) is exact iff G preserves regular epis. (In fact every epi in a topos is regular, so it is enough to require that G preserves epis!) The following facts are now clear from Theorem 2.3.7:

Proposition 2.5.9 *If $\delta : A \twoheadrightarrow B$, $\gamma : B \twoheadrightarrow A$ are applicative morphisms, then*

(i) $(\gamma \dashv \delta)$ is an adjoint pair iff $(\hat{\gamma}, \hat{\delta})$ is an exact geometric morphism;

(ii) $(\gamma \dashv \delta)$ is an inclusion iff $(\hat{\gamma}, \hat{\delta})$ is an exact geometric inclusion;

(iii) $(\gamma \dashv \delta)$ is a retraction iff $(\hat{\gamma}, \hat{\delta})$ is an exact geometric retraction. □

Example 2.5.10 Suppose $A = \mathbf{1}$. Let $\delta : \mathbf{1} \longrightarrow\!\!\!\!\!+ B$, $\gamma : B \longrightarrow\!\!\!\!\!+ \mathbf{1}$ be the (essentially unique) morphisms given by Proposition 2.1.7. Then clearly $(\gamma \dashv \delta)$ is an applicative inclusion. Moreover, we have $\mathbf{Ass}(\mathbf{1}) \simeq \mathbf{RT}(\mathbf{1}) \simeq \mathbf{Set}$, and the functors δ_*, γ_* are (up to isomorphism) none other than ∇, Γ respectively. Thus $(\gamma \dashv \delta)$ gives rise to the geometric inclusion $(\Gamma', \nabla') : \mathbf{Set} \to \mathbf{RT}(B)$ between toposes (cf. Proposition 1.3.9).

We close with some special properties of the functors arising from inclusions and retractions.

Proposition 2.5.11 *(i) If $(\gamma \dashv \delta)$ is an inclusion then $\delta_*, \hat{\delta}$ are full and faithful and preserve exponentials.*

(ii) If $(\gamma \dashv \delta)$ is a retraction then $\hat{\delta}$ preserves finite colimits, and $\gamma_*, \hat{\gamma}$ are faithful and reflect isomorphisms.

Proof (i) For any adjoint pair of functors $H \dashv G$, where $G : \mathcal{C} \to \mathcal{D}$, if the counit is iso then G is full and faithful (see [57, Chapter 4.3]). If moreover both $\mathcal{C}$ and $\mathcal{D}$ are cartesian-closed and H preserves binary products, then for any object X of $\mathcal{C}$ we have natural isomorphisms

$$H(- \times GX) \;\cong\; H(-) \times HGX \;\cong\; H(-) \times X : \;\mathcal{D} \longrightarrow \mathcal{C}.$$

Hence by taking right adjoints we obtain $G((-)^X) \cong (G(-))^{GX}$.

(ii) The fact that $\hat{\delta}$ preserves finite colimits is immediate from Theorem 2.4.21 and Corollary 2.5.4. Moreover, for any adjoint pair $H \dashv G$, it is trivial to show that if the unit is iso then H is faithful and reflects isos. $\square$

Chapter 3

Examples of applicative morphisms

In this chapter we give some applications of the theory developed in Chapter 2. Our intention is to survey a variety of PCAs and to illustrate the diversity embraced by the concept of applicative morphism. We give two kinds of applications. Firstly, by constructing applicative morphisms we show the existence of certain functors between the categories. Secondly, by showing that two PCAs are *inequivalent* we are able to show the inequivalence of the corresponding categories. The spirit of this chapter is exploratory—it contains a somewhat loose assortment of the results we have found to date rather than a comprehensive survey of all known PCAs. We hope that more results in the same vein will be forthcoming in the future.

In Section 3.1 we consider K_1 and similar PCAs arising from the notion of relative computability. In Section 3.2 we discuss term models for combinatory logic and for various λ-calculi, and the morphisms between them. The highlight of this section is a proof that the call-by-name and call-by-value λ-calculi have inequivalent term models. In Section 3.3 we consider Scott's *graph model* $\mathcal{P}\omega$ and its submodel $\mathcal{P}\omega_{re}$. These two PCAs will play an important role in subsequent chapters. Among other things we discover a pleasing proof that $\mathcal{P}\omega_{re}$ is inequivalent to certain term models.

Many of the results in this chapter are (at present) of purely theoretical interest, and are not used in later chapters. The reader wanting only to follow the main line of the thesis should concentrate on Sections 3.2.1 and 3.3.

3.1 K_1 and Turing degrees

In this section we consider PCAs based on the notion of "computability relative to an oracle". The material here is not really new—our aim is simply to show how some known facts can be conveniently presented in terms of applicative morphisms. First we recall some definitions concerning relative computability—for more details see Chapter 9 of [17].

Given a subset $A \subseteq \mathrm{N}$, we write χ_A for the *characteristic function* of A:

$$\chi_A(n) \;=\; \begin{cases} 1 & \text{if } n \in A \\ 0 & \text{if } n \notin A. \end{cases}$$

We write $\deg(A)$ for the set of partial functions that are *recursive in* A (that is, the smallest set of partial functions $\mathrm{N}^k \rightharpoonup \mathrm{N}$ containing χ_A and the basic functions and closed under substitution, recursion and minimalization). We call $\deg(A)$ the *Turing degree* of A. If $A, B \subseteq \mathrm{N}$, we write $A \leq_T B$ if $\chi_A \in \deg(B)$; in this case we say A is *reducible* to B. Clearly $\deg(A) = \deg(B)$ iff $A \leq_T B$ and $B \leq_T A$.

We may think of $\deg(A)$ as the set of partial functions computable by a Turing machine equipped with an *oracle* for A (i.e. a device that magically computes χ_A). We can view such a device as a machine requiring two inputs: a number n supplied on the "input tape", and an oracle χ plugged into the "oracle socket". Let $M_0, M_1, \ldots$ be some fixed effective enumeration of such machines. We write $\{e\}^A(n)$ to denote the result (if any) returned by the machine M_e when supplied with the number n and the oracle χ_A. For each A we thus have an enumeration $\{0\}^A, \{1\}^A, \ldots$ of the partial A-recursive functions $\mathrm{N} \rightharpoonup \mathrm{N}$. We can now introduce the PCAs we wish to study.

Definition 3.1.1 *For any $A \subseteq \mathrm{N}$, let K_1^A be the partial applicative structure $(\mathrm{N}, \cdot)$, where application is defined by $x \cdot y \simeq \{x\}^A(y)$.*

Proposition 3.1.2 *K_1^A is a PCA.*

Proof (Sketch.) Since the enumeration $M_0, M_1, \ldots$ is "effective", one can show that there are $k, s \in \mathrm{N}$ (independent of A) such that for all $x, y, z \in \mathrm{N}$ we have

$$k \cdot x \cdot y \;=\; x, \qquad s \cdot x \cdot y\downarrow, \qquad s \cdot x \cdot y \cdot z \;\simeq\; (x \cdot z) \cdot (y \cdot z). \;\Box$$

Remark 3.1.3 In fact k, s may be chosen such that for any x, y the values of $\{k\}^A(x)$ and $\{\{s\}^A(x)\}^A(y)$ are independent of A. (Intuitively, these values can be computed without "consulting the oracle".)

It is clear from the definitions that any partial A-recursive function is represented by an element of K_1^A in the sense of Theorem 1.1.16. This observation is useful in clearing up a small technical point:

Proposition 3.1.4 *Suppose $M_0, M_1, \ldots$ and $M'_0, M'_1, \ldots$ are two effective enumerations of Turing machines as above, giving rise to PCAs K_1^A, ${K'}_1^A$ for some A. Then $K_1^A \simeq {K'}_1^A$.*

Proof Write $\cdot$, $\cdot'$ for the application operations in K_1^A, ${K'}_1^A$ respectively. The partial function $(x, y) \mapsto x \cdot y$ is clearly A-recursive, and so there exists $r \in {K'}_1^A$ such that $(r \cdot' x) \cdot' y \simeq x \cdot y$. Hence the identity relation $\gamma : K_1^A \longrightarrow\!\!\!\!\!+ \; {K'}_1^A$ is an applicative morphism realized by r. Similarly the identity relation $\delta : {K'}_1^A \longrightarrow\!\!\!\!\!+ \; K_1^A$ is an applicative morphism, and trivially $\delta\gamma = \mathrm{id}$, $\gamma\delta = \mathrm{id}$. Thus $K_1^A \simeq {K'}_1^A$. $\Box$

Exactly the same argument can be used to prove a similar result for K_1:

Proposition 3.1.5 *Suppose $N_0, N_1, \ldots$ and $N'_0, N'_1, \ldots$ are two effective enumerations of Turing machines (without oracle socket), giving rise to PCAs K_1, K'_1. Then $K_1 \simeq K'_1$. Moreover, if A is a recursive set then $K_1 \simeq K_1^A$.* $\Box$

It is easy to see when two PCAs of the form K_1^A are equivalent:

Proposition 3.1.6 *Suppose $A, B \subseteq \mathrm{N}$. Then $A \leq_T B$ iff there is a (necessarily essentially unique) decidable morphism $\delta : K_1^A \longrightarrow\!\!\!\!\!+ \; K_1^B$. Hence $K_1^A \simeq K_1^B$ iff $\deg(A) = \deg(B)$.*

PROOF Suppose $A \leq_T B$. Then application in K_1^A is B-recursive, and so as above the identity relation $\mathrm{id} : K_1^A \longrightarrow K_1^B$ is an applicative morphism. Clearly id is decidable. Conversely, suppose $\delta : K_1^A \longrightarrow K_1^B$ is any decidable morphism. Then by Proposition 2.4.16 there exists $r \in K_1^B$ such that if $m \in \delta(n)$ then $\{r\}^B(m) = n$; and by Remark 2.4.17 there exists t such that $\{t\}^B(n) \in \delta(n)$ for all n. Thus $\mathrm{id} : K_1^A \longrightarrow K_1^B$ is a decidable morphism and $\delta \sim \mathrm{id}$. Take q representing χ_A in K_1^A, u realizing id, and d a decider for id. Then $\lambda^* x.d(uqx)$ (interpreted in K_1^B) represents χ_A in K_1^B. Thus $\chi_A \in \deg(B)$ as required.

It clearly follows that if $K_1^A \simeq K_1^B$ then $\deg(A) = \deg(B)$. Conversely if $A \leq_T B \leq_T A$ then we have decidable morphisms $\delta : K_1^A \longrightarrow K_1^B$ and $\epsilon : K_1^B \longrightarrow K_1^A$. Now $\epsilon\delta$ is decidable, so $\epsilon\delta \sim \mathrm{id}$ by uniqueness. Similarly $\delta\epsilon \sim \mathrm{id}$, and so $K_1^A \simeq K_1^B$.
□

If $A \leq_T B$, the existence of a morphism $K_1^A \longrightarrow K_1^B$ is hardly surprising. What is more surprising is that there is a morphism in the other direction:

Proposition 3.1.7 *If $A \leq_T B$ then there is an applicative inclusion $(\delta \dashv \gamma)$: $K_1^B \to K_1^A$.*

PROOF Let $\delta = \mathrm{id}$; then δ is an applicative morphism by Proposition 3.1.6. Define the relation $\gamma : K_1^B \to K_1^A$ by $\gamma(n) = \{x \mid \{x\}^B(0) = n\}$, and write $\cdot^A$, $\cdot^B$ for the application operations in K_1^A, K_1^B respectively. If the combinator s is well-chosen as in Remark 3.1.3, then for any $x \in \gamma(m)$, $y \in \gamma(n)$ we have

$$(s \cdot^A x \cdot^A y) \cdot^B 0 \;\simeq\; s \cdot^B x \cdot^B y \cdot^B 0 \;\simeq\; (x \cdot^B 0) \cdot^B (y \cdot^B 0) \;\simeq\; \{m\}^B(n)$$

and so $s \cdot^A x \cdot^A y \in \gamma(\{m\}^B(n))$. Thus γ is realized by s.

Clearly $\mathrm{id} \preceq \delta\gamma$ is realized by k and $\delta\gamma \preceq \mathrm{id}$ is realized by $\lambda^* x.x0$ (interpreted in K_1^B). Moreover, if k is well-chosen as in Remark 3.1.3 then k realizes $\mathrm{id} \preceq \gamma\delta$. So δ, γ constitute an inclusion $K_1^B \to K_1^A$. □

We thus have an alternative proof for an already known result: for any A there is a geometric inclusion $\mathcal{E}\!f\!f^A \to \mathcal{E}\!f\!f$ (where $\mathcal{E}\!f\!f^A$ denotes $\mathbf{RT}(K_1^A)$), and indeed the lattice of Turing degrees embeds into the lattice of *topologies* on $\mathcal{E}\!f\!f$. This result appears in [70] (= Section 2.8 of [71]), where an explicit description of the topology induced by A is also given.

We close this section by using applicative morphisms to obtain a proof of another known result: the effective topos is not equivalent to $\mathbf{RT}(C)$ for any *total* combinatory algebra C (see [46]). The original proof depended on finding a categorical property of $\mathcal{E}\!f\!f$ not shared by C; our proof is very similar except that we work entirely with the PCAs. Recall that a PCA C has *decidable equality* if there exists $q \in C$ such that for all $x, y \in C$ we have

$$q\ x\ y \;=\; \begin{cases} \mathit{true} & \text{if } x = y \\ \mathit{false} & \text{if } x \neq y. \end{cases}$$

Theorem 3.1.8 *Suppose B, C are PCAs and $B \simeq C$. If B has decidable equality then so does C.*

Proof Suppose $\gamma : B \longrightarrow\!\!\!\!\!\!\!\!\!\!+\;\; C$ and $\delta : C \longrightarrow\!\!\!\!\!\!\!\!\!\!+\;\; B$ form an equivalence, where both γ, δ are single-valued. Let $q \in B$ be a decider for equality as above. Since δ is discrete, we have

$$q\ (\delta x)\ (\delta y) \;=\; \begin{cases} \mathit{true} & \text{if } x = y \\ \mathit{false} & \text{if } x \neq y. \end{cases}$$

for all $x, y \in C$. Now suppose that r realizes γ, d decides γ and t realizes $\mathrm{id} \preceq \gamma\delta$. Then it is easy to see that the element $p = \lambda^* xy.\ d(q'\ (tx)\ (ty))$ decides equality in C, where $q' = \gamma q$. $\square$

Corollary 3.1.9 *No PCA K_1^A is equivalent to a total combinatory algebra.*

Proof First note that K_1^A has decidable equality. However, suppose C is total and p decides equality in C. Then $Y(p\ \mathit{false})\downarrow$; let $v = Y(p\ \mathit{false})$ and note that $v = p\ \mathit{false}\ v$. But if $v = \mathit{false}$ then $p\ \mathit{false}\ v = \mathit{true}$; and if $v \neq \mathit{false}$ then $p\ \mathit{false}\ v = \mathit{false}$. So we obtain a contradiction. Thus $K_1^A \not\simeq C$. $\square$

3.2 Term models

We now consider applicative morphisms arising in connection with term models for combinatory logic and for various λ-calculi. We will presuppose some acquaintance with the theory of the untyped λ-calculus—see [8] for background. We will also make frequent use of the following facts about applicative morphisms arising from algebraic homomorphisms of PCAs:

Proposition 3.2.1 *Suppose $f : A \to B$ is a function between PCAs such that $f(aa') = f(a) \cdot f(a')$ whenever $aa'\downarrow$. Then*

(i) $f : A \longrightarrow\!\!\!\!\!\!\!\!+\;\; B$ is a projective applicative morphism;

(ii) if $f : A \to B$ is injective then f is discrete;

(iii) if $f : A \to B$ is surjective and moreover $aa'\downarrow$ whenever $f(a) \cdot f(a')\downarrow$, then f has a right adjoint g, and $(f \dashv g) : B \to A$ is an applicative inclusion.

PROOF (i) The element $\lambda^* xy.xy$ realizes f (see Example 2.1.3(ii)), and f is projective since it is single-valued. (ii) Trivial. (iii) Define $g : B \longrightarrow\!\!\!\!\!\!\!\!+\;\; A$ by $g(b) = \{a \mid f(a) = b\}$. Then g is total since f is surjective, and moreover g is realized by $\lambda^* xy.xy$. Clearly $fg = \mathrm{id}_A$, and $\mathrm{id}_B \preceq gf$ is realized by i. $\Box$

3.2.1 Combinatory logic and λ-calculus

We start by considering term models for combinatory logic. For definiteness we concentrate mostly on *closed* term models, though most of our results have obvious counterparts for open term models. Recall that the set $\mathcal{C}^0$ of closed terms of combinatory logic is defined inductively by:

$$K, S \in \mathcal{C}^0, \qquad p, q \in \mathcal{C}^0 \Longrightarrow pq \in \mathcal{C}^0.$$

When writing down elements of $\mathcal{C}^0$ we adopt the convention that juxtaposition is left-associative. A *congruence* on $\mathcal{C}^0$ is an equivalence relation $\sim$ such that

$$p \sim q \implies rp \sim rq \;\wedge\; pr \sim qr$$

for all $p, q, r \in \mathcal{C}^0$. A congruence $\sim$ is called a *CL-theory* if it is consistent (i.e. not every pair of terms is related) and contains all equivalences of the forms

$$Kpq \sim p, \qquad Spqr \sim (pr)(qr).$$

We will write CL to denote the least CL-theory.

If T is a CL-theory then the set $\mathcal{C}^0/T$ of equivalence classes has an applicative structure induced by that of $\mathcal{C}^0$, and is a total combinatory algebra (take k, s to be the equivalence classes of K, S respectively). We call $\mathcal{C}^0/CL$ the *closed term model for combinatory logic.* We now obtain our first result:

Proposition 3.2.2 *If T, T' are CL-theories and $T \subseteq T'$ then there is an applicative inclusion $\mathcal{C}^0/T' \to \mathcal{C}^0/T$. Moreover, this inclusion is an equivalence iff $T = T'$. (Thus the lattice of CL-theories embeds into the lattice of topologies on* $\mathbf{RT}(\mathcal{C}^0/CL)$*.)*

Proof Clearly the quotient map $\mathcal{C}^0/T \to \mathcal{C}^0/T'$ is surjective and preserves application, hence by Proposition 3.2.1 we have an inclusion $(\delta \dashv \gamma) : \mathcal{C}^0/T' \to \mathcal{C}^0/T$. Now suppose this inclusion is an equivalence. Given $p, q \in \mathcal{C}^0$ such that $T' \vdash p = q$, let $[p], [q]$ denote the equivalence classes of p, q respectively in $\mathcal{C}^0/T$. Then $\gamma([p]) = \gamma([q])$, so $[p] = [q]$ since γ is discrete. Hence $T \vdash p = q$; thus $T = T'$.
$\square$

Remark 3.2.3 It seems reasonable to conjecture that if T, T' are distinct CL-theories then the PCAs $\mathcal{C}^0/T$, $\mathcal{C}^0/T'$ are inequivalent. However, so far we have been unable to prove any inequivalences at all between PCAs of the form $\mathcal{C}^0/T$!

An important class of CL-theories is the class of λ-*theories*. Let Λ^0/β denote the set of closed terms of the untyped λ-calculus modulo β-equality; recall from Section 7.3 of [8] that for some CL-theory β we have $\Lambda^0/\beta \cong \mathcal{C}^0/\beta$. By a λ-*theory* we will mean an equivalence relation T on Λ^0/β such that

$$M =_T N \implies PM =_T PN \ \wedge \ MP =_T NP,$$

or equivalently a CL-theory T such that $\beta \subseteq T$. (It is easy to show that this agrees with the notion of λ-theory in [8, Chapter 4].) We will use the same letter to denote both the CL-theory and the equivalence relation on Λ^0/β. The following can now be seen as a consequence of the previous proposition:

Proposition 3.2.4 *If T, T' are λ-theories and $T \subseteq T'$ then there is an applicative inclusion $\Lambda^0/T' \to \Lambda^0/T$. This inclusion is an equivalence iff $T = T'$.* $\Box$

Recall that a λ-theory T is called *sensible* if it equates all unsolvable terms (i.e. terms with no head normal form), The following are some important λ-theories:

> $\mathcal{K}$, the smallest sensible λ-theory;
> $\mathcal{B}$, the theory that equates terms iff they have the same Böhm tree;
> $\mathcal{K}^*$, the theory that equates terms iff their Böhm trees are η-equivalent
> (in fact this gives us the unique maximal sensible theory).

We use these symbols to denote both the λ-theories and the corresponding CL-theories. Thus we have the inclusions

$$CL \ \subset\ \beta\ \subset\ \mathcal{K}\ \subset\ \mathcal{B}\ \subset\ \mathcal{K}^*.$$

Proposition 3.2.5 *Suppose T, T' are CL-theories and $\beta \subseteq T \subseteq T' \subseteq \mathcal{B}$. Then the canonical applicative morphism $\mathcal{C}^0/T' \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}^0/T$ is decidable.*

Proof If $T' \vdash M = true$ then $\mathcal{B} \vdash M = true$; hence $M =_\beta true$ since $true$ has a β-normal form, and so $T \vdash M = true$. Similarly for *false*. Thus i is a decider for the morphism $\mathcal{C}^0/T' \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}^0/T$. $\Box$

For T any CL-theory, we know already that $\mathcal{C}^0/T \not\simeq K_1$ since $\mathcal{C}^0/T$ is total. However, it is natural to ask what morphisms do exist between K_1 and $\mathcal{C}^0/T$.

Theorem 3.2.6 *For any T, there is an adjoint pair $(\kappa \dashv \nu) : \mathcal{C}^0/T \to K_1$. Moreover, if $T \subseteq \mathcal{K}^*$ then this pair is an applicative retraction.*

Proof Let κ be the essentially unique decidable morphism $K_1 \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}^0/T$ as in Proposition 2.4.18. Let $\ulcorner - \urcorner$ be some effective Gödel-numbering $\mathcal{C}^0 \to \mathrm{N}$, and define $\nu : \mathcal{C}^0/T \longrightarrow\!\!\!\!\!\!+\ K_1$ by $\nu([p]) = \{\ulcorner q \urcorner \mid T \vdash p = q\}$. Take $r \in K_1$ such that $r \cdot \ulcorner p \urcorner \cdot \ulcorner q \urcorner = \ulcorner pq \urcorner$; then r realizes ν. Let $\overline{n}$ denote the standard CL-term representing a natural number n (see Proposition 1.1.11); clearly the function $n \mapsto \ulcorner \overline{n} \urcorner$ is recursive and $\ulcorner \overline{n} \urcorner \in \nu\kappa(n)$, so we have id $\preceq \nu\kappa$.

To see that $\kappa\nu \preceq$ id, it suffices construct a CL-term E (called an *enumerator* for $\mathcal{C}^0$) such that $CL \vdash E\overline{\ulcorner p \urcorner} = p$ for all $p \in \mathcal{C}^0$. First, by Theorem 1.1.16 there exist elements *isK*, *isS*, *left*, *right* $\in \mathcal{C}^0$ such that for all $p, q \in \mathcal{C}^0$ we have

$$isK\ \overline{\ulcorner p \urcorner} = \begin{cases} true & \text{if } p \equiv K \\ false & \text{otherwise,} \end{cases} \qquad isS\ \overline{\ulcorner p \urcorner} = \begin{cases} true & \text{if } x \equiv S \\ false & \text{otherwise,} \end{cases}$$

$$left\ \overline{\ulcorner pq \urcorner} = \overline{\ulcorner p \urcorner}, \qquad right\ \overline{\ulcorner pq \urcorner} = \overline{\ulcorner q \urcorner}.$$

Informally, we wish to define E as follows:

$$\begin{aligned} \text{“fun } E\ x = {} & \text{if } (isK\ x) \text{ then } K \\ & \text{else if } (isS\ x) \text{ then } S \\ & \text{else } (E(left\ x))(E(right\ x)).\text{”} \end{aligned}$$

More formally, let

$$E = Z(\lambda^* gx.\ if(isK\ x)\,(K)\,(if(isS\ x)\,(S)\,((g(left\ x))(g(right\ x))))).$$

It is straightforward to show by structural induction that $CL \vdash E\overline{\ulcorner p \urcorner} = p$.

Finally, if $p \in CL$ then let p_Λ denote the λ-term obtained via the translation:

$$K_\Lambda = \lambda xy.x, \qquad S_\Lambda = \lambda xyz.(xz)(yz), \qquad (pq)_\Lambda = p_\Lambda q_\Lambda.$$

If $\ulcorner\!\ulcorner - \urcorner\!\urcorner$ is an effective Gödel-numbering of λ-terms then clearly there is a recursive function h such that $h(\ulcorner p \urcorner) = \ulcorner\!\ulcorner p_\Lambda \urcorner\!\urcorner$. Moreover, there is a partial recursive function j such that if $M \precsim_\eta \overline{n}$ then $j(\ulcorner\!\ulcorner M \urcorner\!\urcorner) = n$ (Informally, j computes the leftmost branch of the Böhm tree of M—see 10.2 of [8].) But $M \precsim_\eta \overline{n}$ iff $\mathcal{K}^* \vdash M = \overline{n}$ (see Theorem 16.2.7 of [8]). Hence if $T \subseteq K^*$ and $T \vdash p = \overline{n}$ then $p_\Lambda \precsim_\eta \overline{n}$ and so $jh(\ulcorner p \urcorner) = n$. Thus $\nu\kappa \preceq$ id, so $(\kappa \dashv \nu)$ is an applicative retraction. $\square$

3.2.2 Open and closed term models

Before continuing our survey of different λ-calculi we digress briefly to touch on the relationship between *open* and *closed* term models. Recall that Λ denotes the set of λ-terms over a countably infinite set of variables. We write $\Lambda^0(x)$ for the set of λ-terms M such that $\mathrm{FV}(M) \subseteq \{x\}$.

Proposition 3.2.7 *For any λ-theory T, there is a retraction $\Lambda^0(x)/T \to \Lambda^0/T$.*

Proof Let γ be the applicative morphism determined by the inclusion $\Lambda^0/T \hookrightarrow \Lambda^0(x)/T$, and let $\delta : \Lambda^0(x)/T \longrightarrow\!\!\!\!\!\!\rightarrow \Lambda^0/T$ be the applicative morphism given by $\delta(M) = \{\lambda x.M\}$ (this is realized by $\lambda yzx.(yx)(zx)$). Then $\mathrm{id} \preceq \delta\gamma$ is realized by $\lambda yx.y$; $\delta\gamma \preceq \mathrm{id}$ is realized by $\lambda y.yi$; and $\gamma\delta \preceq \mathrm{id}$ is realized by $\lambda y.yx$. $\Box$

This retraction will not be an equivalence in general: e.g. if $T = \beta$ and M realizes $\mathrm{id} \preceq \gamma\delta$ then $Mx =_\beta \lambda x.x$ so $MN =_\beta \lambda x.x$ for all N, and so $\lambda x.N =_\beta \lambda x.x$, a contradiction. We would conjecture that for any theory T the PCAs Λ^0/T, $\Lambda^0(x)/T$ are in fact inequivalent. However, so far we have only managed to prove the following:

Theorem 3.2.8 *For any theory T generated by equations between closed terms, the PCAs Λ^0/T, Λ/T are inequivalent.*

Proof Recall that $\mathbf{k}, \mathbf{s}$ generate the whole of Λ^0/T under application. Suppose $\gamma : \Lambda^0/T \longrightarrow\!\!\!\!\!\!\rightarrow \Lambda/T$ and $\delta : \Lambda/T \longrightarrow\!\!\!\!\!\!\rightarrow \Lambda^0/T$ form an equivalence, and take R realizing γ, U realizing $\gamma\delta \preceq \mathrm{id}$, $K \in \gamma(\mathbf{k})$ and $S \in \gamma(\mathbf{s})$. Given any $N \in \Lambda/T$, take $N' \in \delta(N)$; then N' is generated by $\mathbf{k}, \mathbf{s}$ in Λ^0/T. Thus some $N'' \in \gamma(N')$ is generated by R, K, S in Λ/T. But $T \vdash UN'' = N$. Thus R, K, S and U generate the whole of Λ^0/T. This is impossible since R, K, S, U between them contain only finitely many free variables. $\Box$

3.2.3 Lazy λ-calculus

In this paragraph we briefly consider the *lazy λ-calculus*, a variant of the standard untyped λ-calculus. This system has been studied extensively by Abramsky and

Ong (see e.g. [2,4]); it was conceived as a prototypical lazy functional language in which terms are evaluated to *weak head normal form* rather than head normal form as in the classical theory.

The set of terms of the lazy λ-calculus is just the usual set Λ of untyped λ-terms, but the fundamental notion of equality between terms is a restricted form of β-equality which does not allow the contraction of redexes under abstractions. More formally, let ℓ be the least equivalence relation on Λ (or Λ^0) such that

$$(\lambda x.M)N \ =_\ell \ M[N/x]; \qquad M =_\ell N \implies MP =_\ell NP \ \wedge \ PM =_\ell PN$$

for all $M, N, P \in \Lambda$. Then the quotient Λ^0/ℓ is clearly a total PCA (with k, s as before)—we call it the *closed term model for the lazy λ-calculus*. In general, a congruence T on Λ^0 is called a *lazy λ-theory* if T is consistent and contains ℓ. So just as before we have

Proposition 3.2.9 *If T, T' are lazy λ-theories and $T \subseteq T'$ then there is an applicative inclusion $\Lambda^0/T' \to \Lambda^0/T$. This inclusion is an equivalence iff $T = T'$.* $\square$

Remark 3.2.10 Note that the usual β-equality on Λ^0 provides an example of a lazy λ-theory. Thus any standard λ-theory may be regarded as a lazy λ-theory containing β. In particular we have an applicative inclusion between term models $\Lambda^0/\beta \to \Lambda^0/\ell$.

Thus the term model Λ^0/β may be regarded as a quotient either of $\mathcal{C}^0/CL$ or of Λ^0/ℓ. However, neither of $\mathcal{C}^0/CL$, Λ^0/ℓ is (canonically) a quotient of the other: on the one hand, the term $\lambda x.x$ is not ℓ-equivalent to any λ-term built from $\mathbf{k}, \mathbf{s}$ using juxtaposition; and on the other hand, the combinator S is not CL-equivalent to any combinator in the image of the standard translation $\Lambda^0 \to \mathcal{C}^0$.

Again, we would conjecture that distinct lazy λ-theories never give rise to equivalent PCAs, and that if T is a CL-theory and T' is a lazy λ-theory then $\mathcal{C}^0/T$, Λ^0/T' are equivalent iff they are canonically isomorphic.

3.2.4 Call-by-value λ-calculus

We now turn to consider another variant of the λ-calculus: the *call-by-value* λ-calculus studied by Plotkin [78]. This system is also called the λ_V-calculus, and by contrast the standard theory is sometimes called the λ_N-calculus or *call-by-name* λ-calculus.

The λ_V calculus may be described as follows. We say that a λ-term M is a *value* if it is a variable, a constant or an abstraction (i.e. if it is not of the form $M_1 M_2$). We write $\mathcal{V} \subset \Lambda$ for the set of values, and $\mathcal{V}^0$ for $\mathcal{V} \cap \Lambda^0$. We now define the reduction relation $\rightarrow_V$ to be the smallest reflexive-transitive relation such that

$$(\lambda x.M)N \;\rightarrow_V\; M[N/x] \quad \text{whenever } N \in \mathcal{V};$$
$$M \rightarrow_V N \implies MP \rightarrow_V NP \;\wedge\; PM \rightarrow_V PN \;\wedge\; \lambda x.M \rightarrow_V \lambda x.N$$

for all $M, N, P \in \Lambda$. Let V denote the equivalence relation generated by $\rightarrow_V$. We can endow $\mathcal{V}^0/V$ with a partial applicative structure by declaring that

$$[M] \cdot [N] \;=\; [P] \quad \text{iff} \quad MN \;=_V\; P$$

for all $M, N, P \in \mathcal{V}^0$. (It is easy to see that this does define a partial binary operation on $\mathcal{V}^0/V$.) We will often abuse our notion and write M instead of $[M]$. Note that $M \cdot N\!\downarrow$ iff there is a value P such that $M \rightarrow_V P$ and $N \rightarrow_V P$, since the Church-Rosser property holds for $\rightarrow_V$ (see Section 4 of [78]).

Proposition 3.2.11 *$(\mathcal{V}^0/V, \cdot)$ is a PCA.*

Proof First note that for $M, N, P \in \mathcal{V}^0$ we have

$$\mathbf{k} \cdot M \cdot N \;\simeq\; (\lambda y.M) \cdot N \;\simeq\; M;$$
$$\mathbf{s} \cdot M \cdot N \;\simeq\; (\lambda yz.Mz(yz)) \;\simeq\; (\lambda z.Mz(Nz))\downarrow.$$

Now if $(M \cdot P) \cdot (N \cdot P)\!\downarrow$ then $(M \cdot P) \cdot (N \cdot P) = (MP)(NP)$; but $(\mathbf{s} \cdot M \cdot N)P = (MP)(NP)$ and so $\mathbf{s} \cdot M \cdot N \cdot P = (M \cdot P) \cdot (N \cdot P)$. Finally, if $\mathbf{s} \cdot M \cdot N \cdot P\!\downarrow$ then $(MP)(NP) \rightarrow_v Q$ for some value Q. Suppose UV is the last term in the reduction sequence that is not a value. Then U, V are values and $M \cdot P = U,\ N \cdot P = V$. Thus $(M \cdot P) \cdot (N \cdot P)\!\downarrow$. $\square$

We say that an equivalence relation T on $\mathcal{V}^0$ is a *λ_V-theory* if T is consistent and contains V, and moreover T is a congruence in the following sense:

$$M =_T N \;\wedge\; M \cdot P\!\downarrow \;\Longrightarrow\; M \cdot P =_T N \cdot P;$$
$$M =_T N \;\wedge\; P \cdot M\!\downarrow \;\Longrightarrow\; P \cdot M =_T P \cdot N.$$

We call T a *sensible* λ_V-theory if T identifies all terms M such that $M \cdot N\!\uparrow$ for all $N \in \mathcal{V}^0$. If T is a λ_V-theory then clearly $\mathcal{V}^0/T$ is a PCA, and as before we have

Proposition 3.2.12 *If T, T' are λ_V-theories and $T \subseteq T'$ then there is an applicative inclusion $\Lambda^0/T' \to \Lambda^0/T$. This inclusion is an equivalence iff $T = T'$.* $\square$

Remark 3.2.13 Although $\mathcal{V}^0/T$ is not a total combinatory algebra, it is easy to see that it does not have decidable equality (and hence $\mathcal{V}^0/T \not\simeq K_1^A$). For if $\mathcal{V}^0/T$ has decidable equality then there exists $p \in \mathcal{V}^0/T$ such that for all $q \in \mathcal{V}^0/T$ we have

$$p \cdot q \;=\; \begin{cases} \mathit{true} & \text{if } T \vdash q = \bot \\ \mathit{false} & \text{otherwise} \end{cases}$$

(where $\bot = \lambda y.(\lambda x.xx)(\lambda x.xx)$). Now take r such that $r \cdot \mathit{true} = i$ and $r \cdot \mathit{false}\!\uparrow$, and let $t = \lambda^* x.r(px)$. Then we have $t \cdot q\!\downarrow$ iff $T \vdash q = \bot$. But by the context lemma for the λ_V-calculus, if $t \cdot \bot\!\downarrow$ then $t \cdot q\!\downarrow$ for all $q \in \mathcal{V}^0/T$, so we have a contradiction.

Remark 3.2.14 It is also possible to formulate a "lazy call-by-value λ-calculus", where in the definition of the reduction relation $\to_V$ we omit the clause

$$M \to_V N \;\Longrightarrow\; \lambda x.M \to_V \lambda x.N.$$

This system has been studied in [18]. From the point of view of applicative morphisms, the relationship between this theory and the λ_V-calculus is analogous to that between the lazy and standard λ-calculi.

The rest of this section is devoted to investigating the relationship between the call-by-name and call-by-value term models. Firstly, there is a rather trivial morphism from the call-by-value to the call-by-name model:

Proposition 3.2.15 *Suppose T is a λ_V-theory and U is a λ_N-theory. Then the inclusion $\mathcal{V}^0 \hookrightarrow \Lambda^0$ gives rise to an applicative morphism $\iota : \mathcal{V}^0/T \multimap \Lambda^0/U$. Moreover, this morphism is decidable and projective if $T \subseteq U$, but never discrete.*

Proof The relation ι is defined as follows:

$$\iota([M]) \;=\; \{[N] \in \Lambda^0/U \mid T \vdash N = M\}.$$

This relation is well-defined and total, and is an applicative morphism realized by $\lambda^* xy.xy$, since if $M_1, M_2 \in \mathcal{V}^0$, $M_1 \cdot M_2 \downarrow$ and $T \vdash N_1 = M_1, N_2 = M_2$ then $T \vdash N_1 N_2 = M_1 \cdot M_2$. If $T \subseteq U$ then clearly $\iota([M]) = \{[M]\}$, so ι is projective; moreover $\iota(\mathit{true}) = \{\mathit{true}\}$ and $\iota(\mathit{false}) = \{\mathit{false}\}$ so ι is decidable. However, if ι is discrete then for all $M, N \in \mathcal{V}^0$ with $M =_\beta N$ we must have $T \vdash M = N$. In particular, if $M = \lambda xy.y$ and $N = \lambda x.(\lambda zy.y)(\bot i)$ then $T \vdash M = N$. But $M \cdot i \downarrow$ and $N \cdot i \uparrow$, so we have a contradiction. □

More interestingly, applicative morphisms arise from the simulations of call-by-value in call-by-name λ-calculus and vice versa given by Plotkin [78]. In that paper, a syntactic translation $\mathcal{V} \to \Lambda$, $M \mapsto \overline{M}$ was defined inductively as follows:

$$\overline{x} \;=\; \lambda z.zx, \qquad \overline{\lambda x.M} \;=\; \lambda z.z(\lambda x.\overline{M}), \qquad \overline{MN} \;=\; \lambda z.\overline{M}(\lambda a.\overline{N}(\lambda b.abz)).$$

Also a translation $\Lambda \to \mathcal{V}$, $M \mapsto \underline{M}$ was defined by:

$$\underline{x} \;=\; x, \qquad \underline{\lambda x.M} \;=\; \lambda z.z(\lambda x.\underline{M}), \qquad \underline{MN} \;=\; \lambda z.\underline{M}(\lambda a.a\underline{N}z).$$

Clearly these translations restrict to functions $\mathcal{V}^0 \to \Lambda^0$ and $\Lambda^0 \to \mathcal{V}^0$ respectively. It is not hard to show that these functions give rise to applicative morphisms:

Proposition 3.2.16 *Suppose T is a λ_V-theory and U is a λ_N-theory. Then there are applicative morphisms $\gamma : \mathcal{V}^0/T \multimap \Lambda^0/U$ and $\delta : \Lambda^0/U \multimap \mathcal{V}^0/T$ defined by*

$$\gamma([M]) = \{[\overline{N}] \in \Lambda^0/U \mid T \vdash N = M\},$$
$$\delta([M]) = \{[\underline{N}] \in \mathcal{V}^0/T \mid U \vdash N = M\}.$$

Proof The relations γ, δ are clearly well-defined and total. But γ is realized by $R = \lambda xyz.x(\lambda a.y(\lambda b.abz))$, since if $M_1, M_2 \in \mathcal{V}^0$, $M_1 \cdot M_2 \downarrow$ and $T \vdash N_1 = M_1$, $T \vdash N_2 = M_2$ then $R\,\overline{N_1}\,\overline{N_2} =_\beta \overline{N_1 N_2}$ and $T \vdash N_1 N_2 = M_1 \cdot M_2$, so $R\,\overline{N_1}\,\overline{N_2} \in \gamma([M_1 M_2])$. A similar argument proves that δ is realized by $\lambda xyz.x(\lambda a.ayz)$. □

The morphisms γ, δ turn out to be particularly well-behaved in the case of the term models $\mathcal{V}^0/V$ and Λ^0/β:

Proposition 3.2.17 *Suppose $T = V$ and $U = \beta$ in the above proposition. Then both γ and δ are discrete, projective and decidable.*

Proof It is proved in [78] that for $M, N \in \mathcal{V}^0$ we have $v \vdash M = N$ iff $\beta \vdash \overline{M} = \overline{N}$. Thus $\gamma([M]) = \{[\overline{M}]\}$, γ is discrete and projective. To see that γ is decidable, note that

$$\overline{\mathit{true}} = \lambda z.z(\lambda xw.w(\lambda yv.vx)), \qquad \overline{\mathit{false}} = \lambda z.z(\lambda xw.w(\lambda yv.vy)).$$

Take $D = \lambda t.t(\lambda u.u\ \mathit{true}\ (\lambda q.q\ \mathit{false}\ i))$; then it is easily checked that

$$\beta \vdash D\ \overline{\mathit{true}} = \mathit{true}, \qquad \beta \vdash D\ \overline{\mathit{false}} = \mathit{false}.$$

Thus D is a decider for γ.

The argument for δ is similar. We know from [78] that for $M, N \in \Lambda^0$ we have $\beta \vdash M = N$ iff $v \vdash \underline{M} = \underline{N}$; hence δ is discrete and projective. Moreover it is routine to verify that the term $\lambda t.t(\lambda u.u\ \mathit{true}\ (\lambda q.q\ \mathit{false}))$ is a decider for δ. $\square$

It would be interesting to know whether there are other "matching" pairs of theories T, U for which the corresponding morphisms γ, δ are similarly well-behaved.

3.2.5 Inequivalence of λ_N and λ_V

We conclude this section by showing that the PCAs $\mathcal{V}^0/T$, Λ^0/U are inequivalent, at least when T, U are sensible theories. This result is particularly striking since it gives rise to a previously unknown inequivalence between realizability toposes. (In fact, we do not know of any purely categorical way of showing that $\mathbf{RT}(\mathcal{V}^0/T)$ and $\mathbf{RT}(\Lambda^0/U)$ are inequivalent.)

We first fix on some notation. Write I, Ω to denote the equivalence classes in Λ^0/U of the terms $\lambda x.x$, $(\lambda x.xx)(\lambda x.xx)$ respectively, and write $\top, \bot$ for the equivalence classes in $\mathcal{V}^0/T$ of YK, $\lambda y.(\lambda x.xx)(\lambda x.xx)$ respectively. Note that $M \in \Lambda^0$ is unsolvable iff $U \vdash M = \Omega$; and $N \in \mathcal{V}^0$ satisfies $\forall P.N \cdot P \uparrow$ iff $T \vdash N = \bot$. We define the *observational preorder* $\lesssim$ on Λ^0/U by

$$M_1 \lesssim M_2 \quad \text{iff} \quad \forall P \in \Lambda^0.\ (PM_1 \text{ solvable}) \implies (PM_2 \text{ solvable})$$

(note that $\lesssim$ is well-defined on U-equivalence classes). We also define the preorder $\lesssim$ on $\mathcal{V}^0/T$ by

$$N_1 \lesssim N_2 \quad \text{iff} \quad \forall P \in \mathcal{V}^0.\ P \cdot N_1{\downarrow} \implies P \cdot N_2{\downarrow}$$

(this relation is similarly well-defined). It is easy to see that in both cases we have $M \lesssim N \implies C[M] \lesssim C[N]$ for any context C. In both cases we write $M \approx N$ to mean $M \lesssim N \wedge N \lesssim M$. Note that $\Omega \lesssim M$ for all $M \in \Lambda^0/U$, and $\bot \lesssim N \lesssim \top$ for all $N \in \mathcal{V}^0/T$.

The rest of this section is devoted to the proof of the following:

Theorem 3.2.18 *Suppose T is a sensible λ_V-theory, U a sensible λ_V-theory. Then $\mathcal{V}^0/T \not\simeq \Lambda^0/U$.*

To prove this, we suppose that we have morphisms

$$\gamma :\ \mathcal{V}^0/T \longrightarrow \Lambda^0/U, \qquad \delta : \Lambda^0/U \longrightarrow \mathcal{V}^0/T$$

forming an equivalence, and head for a contradiction. We assume without loss of generality that both γ, δ are single-valued relations, and for notational convenience regard them as *functions* between PCAs. The proof is easy once we have established that γ is monotone with respect to $\lesssim$. We show this by the following series of lemmas.

Lemma 3.2.19 *$\delta I \not\approx \delta\Omega$ and $\gamma\top \not\approx \gamma\bot$.*

Proof Suppose $\delta I \approx \delta\Omega$. Let $M = \lambda x.x\ \mathit{true}$ in Λ^0/U; note that $MI = \mathit{true}$ and $M\Omega = \Omega$ in Λ^0/U. So there exists $M' \in \mathcal{V}^0/T$ such that $M' \cdot \delta I = \delta\ \mathit{true}$ and $M' \cdot \delta\Omega = \delta\Omega$; thus $\delta\ \mathit{true} \approx \delta\Omega$. Similarly $\delta\ \mathit{false} \approx \delta\Omega$, so $\delta\ \mathit{true} \approx \delta\ \mathit{false}$. But δ is decidable, so $\mathit{true} \approx \mathit{false}$ in $\mathcal{V}^0/T$—a contradiction.

Similarly, suppose $\gamma\top \approx \gamma\bot$. Let $N = \lambda xz.(\lambda y.true)(xi)$ in $\mathcal{V}^0/T$; note that $N \cdot \top = \lambda z.true$ but $N \cdot \bot = \lambda z.(\lambda y.true)(\bot i)$, hence $N \cdot \bot = \bot$ as $\bot i$ has no value. So as before we have $\gamma\ true \approx \gamma\bot$ and similarly $\gamma\ false \approx \gamma\bot$. But γ is decidable, so $true \approx false$ in Λ^0/T—a contradiction. □

Lemma 3.2.20 $\gamma\top \not\lesssim \gamma\bot$.

Proof Let $P = \lambda xyw.((\lambda z.i)(xi))((\lambda z.\top)(yi))$ in $\mathcal{V}^0/T$. Then clearly $P \cdot \top \cdot \top = \lambda w.\top = \top$ in $\mathcal{V}^0/T$. Also $P \cdot \top \cdot \bot = \lambda w.(\lambda z.\top)(\bot i) = \bot$ in $\mathcal{V}^0/T$, and similarly $P \cdot \bot \cdot \top = \bot$. So there exists $P' \in \Lambda^0/U$ such that

$$P'(\gamma\top)(\gamma\top) \;=\; \gamma\top, \qquad P'(\gamma\top)(\gamma\bot) \;=\; P'(\gamma\bot)(\gamma\top) \;=\; \gamma\bot \quad (\text{in } \Lambda^0/U).$$

Suppose $\gamma\top \lesssim \gamma\bot$. We already know that $\gamma\top \not\approx \gamma\bot$; the idea now is that the existence of P' contradicts Berry's stability theorem on the non-existence of "parallel" functions in the λ_N-calculus. Unfortunately, the form of this theorem given in [8] concerns stability with respect to Böhm tree inclusion rather than our relation $\lesssim$, and so a little hacking is required before we can invoke the theorem.

Note that according to Theorem 19.2.9(i) of [8] we have $\mathrm{BT}(\gamma\top)\ {}^{\eta}\!\subseteq^{\eta} \mathrm{BT}(\gamma\bot)$, and so by Exercise 10.6.7(ii) of [8] there exist $G, H \in \Lambda^0/T$ such that

$$\mathrm{BT}(\gamma\top) \leq_\eta \mathrm{BT}(G) \subseteq \mathrm{BT}(H) \geq_\eta \mathrm{BT}(\gamma\bot).$$

Now $\mathrm{BT}(\gamma\top) =_\eta \mathrm{BT}(G)$, and so $\gamma\top \approx G$ by Theorem 19.2.9; similarly $\gamma\bot \approx H$. Thus $G \not\approx H$. Note also that we have

$$P'GG \;\approx\; G, \qquad P'GH \;\approx\; P'HG \;\approx\; H.$$

Take Q such that QG is unsolvable but QH is solvable. Suppose $Q(P'GH)$ and $Q(P'HG)$ have principal head normal forms

$$Q(P'GH) \;=\; \lambda x_1 \ldots x_m.x_i R_1 \ldots R_p, \qquad Q(P'HG) \;=\; \lambda y_1 \ldots y_n.y_j S_1 \ldots S_q.$$

Then $i = j$ and $p - m = q - n$ (see Theorem 10.2.31 of [8]). Assume $m \geq n$ without loss of generality, and let $J = \lambda zx_1 \ldots x_n.zx_1 \ldots x_n$. Let $M_1 = J(Q(P'GH))$ and $M_2 = J(Q(P'HG))$; note that the Böhm trees of M_1, M_2 intersect non-trivially (in particular $\mathrm{BT}(M_1) \cap \mathrm{BT}(M_2) \supseteq \mathrm{BT}(\lambda x_1 \ldots x_m.x_i\Omega_1 \ldots \Omega_p)$).

Now let $\mathcal{X} = \{\lambda x.xGH, \lambda x.xHG\}$. In the notation of Section 14.4 of [8], clearly $\uparrow \mathcal{X}$ and $\sqcap\mathcal{X} \approx \lambda x.xGG$. Let $C[M] \equiv J(Q(P'(\mathit{fst}\ M)(\mathit{snd}\ M)))$. By Theorem 14.4.10 of [8] we have

$$\begin{aligned}\mathrm{BT}(C[\lambda x.xGG]) &= \mathrm{BT}(C[\lambda x.xGH]) \cap \mathrm{BT}(C[\lambda x.xHG]) \\ &\supseteq \mathrm{BT}(\lambda x_1 \ldots x_m.x_i\Omega_1 \ldots \Omega_p).\end{aligned}$$

But $C[\lambda x.xGG] \approx J(QG)$ which is unsolvable—a contradiction. $\square$

Lemma 3.2.21 *$\delta\Omega \lesssim \delta I$.*

Proof Suppose there exists $N \in \mathcal{V}^0/T$ such that $N \cdot \delta\Omega\downarrow$ but $N \cdot \delta I\uparrow$. Consider $P = \lambda zw.(\lambda x.\top)(Nz)$; note that $P \cdot \delta\Omega = \top$ and $P \cdot \delta I = \bot$. So there exists $P' \in \Lambda^0/U$ such that $P'(\gamma\delta\Omega) = \gamma\top$ and $P'(\gamma\delta I) = \gamma\bot$. Hence, since $\mathrm{id} \preceq \gamma\delta$, there exists $Q \in \Lambda^0/U$ such that $Q\Omega = \gamma\top$ and $QI = \gamma\bot$. But $\Omega \lesssim I$ and so $\gamma\top \lesssim \gamma\bot$, contradicting Lemma 3.2.20. $\square$

Lemma 3.2.22 *δ reflects $\lesssim$, i.e. if $\delta M_1 \lesssim \delta M_2$ then $M_1 \lesssim M_2$.*

Proof Suppose $M_1, M_2 \in \Lambda^0/U$ are such that $\delta M_1 \lesssim \delta M_2$ but $M_1 \not\lesssim M_2$. Then there exists $P \in \Lambda^0/U$ such that $PM_1 = I$, $PM_2 = \Omega$. Thus there exists $P' \in \mathcal{V}^0/T$ such that $P' \cdot \delta M_1 = \delta I$ and $P' \cdot \delta M_2 = \delta\Omega$. Next note that, since $\delta\Omega \lesssim \delta I$ but $\delta\Omega \not\approx \delta I$, there exists $Q \in \mathcal{V}^0/T$ such that $Q \cdot \delta I\downarrow$ but $Q \cdot \delta\Omega\uparrow$. Take $R = \lambda xw.\top(Q(P'x))$; then $R \cdot \delta M_1 = \top$ and $R \cdot \delta M_2 = \bot$. But $\delta M_1 \lesssim \delta M_2$ and so $\top \lesssim \bot$, a contradiction. $\square$

Lemma 3.2.23 *γ preserves $\lesssim$.*

Proof Suppose $N_1, N_2 \in \mathcal{V}^0/T$ and $N_1 \lesssim N_2$. Then $\delta\gamma N_1 \lesssim \delta\gamma N_2$ since $\mathrm{id} \preceq \delta\gamma$; hence $\gamma N_1 \lesssim \gamma N_2$ by the last lemma. $\square$

We can now prove Theorem 3.2.18. Observe that *true*, *false* $\lesssim \top$ in $\mathcal{V}^0/T$; hence $\gamma(\mathit{true})$, $\gamma(\mathit{false}) \lesssim \gamma\top$ in Λ^0/U. Suppose $Q \in \Lambda^0/U$ is a decider for γ; then we have *true*, *false* $\lesssim Q(\gamma\top)$ in Λ^0/U. This is a contradiction since *true*, *false* have no upper bound with respect to $\lesssim$ (since clearly there can be no $M \in \Lambda^0/U$ such that $BT(\mathit{true})$, $\mathrm{BT}(\mathit{false})\ {}^{\eta}\!\subseteq^{\eta} \mathrm{BT}(M)$). The proof of the theorem is now complete. $\square$

3.3 The Scott graph model

There is a large class of PCAs based on the idea of "continuous function application". Examples are Scott's graph model $\mathcal{P}\omega$ ([89]), Kleene's second model K_2 ([50]), and lattice-theoretic and CPO-models such as Scott's D_∞ ([88]). In this section we concentrate on the graph models $\mathcal{P}\omega$ and $\mathcal{P}\omega_{re}$, though we anticipate that it will be possible to obtain results in the same spirit for many other PCAs.

3.3.1 The PCAs $\mathcal{P}\omega$ and $\mathcal{P}\omega_{re}$

We start by explaining the construction of Scott's graph model. Let ω denote the set of natural numbers, and let $\mathcal{P}\omega$ be its powerset. The idea is to give $\mathcal{P}\omega$ the structure of a (total) combinatory algebra by "encoding" functions $\mathcal{P}\omega \to \mathcal{P}\omega$ as subsets of ω. Of course it will not be possible to encode *all* such functions in this way, since card $(\mathcal{P}\omega \to \mathcal{P}\omega) >$ card $(\mathcal{P}\omega)$, so the idea is to encode just the *continuous* functions. Recall that a function $f : \mathcal{P}X \to \mathcal{P}X$ is continuous if f preserves inclusion and $f(\bigcup A_i) = \bigcup fA_i$ whenever $A_0 \subseteq A_1 \subseteq \ldots \subseteq \mathcal{P}\omega$.

The crucial observation is that such a function is completely determined by its action on *finite* subsets of ω, since any $A \in \mathcal{P}\omega$ can be expressed as the union of a chain of finite subsets. Suppose that $(-;-) : \omega \times \omega \to \omega$ is some (injective) "coding" function for pairs of natural numbers, and that $[-, \ldots, -] : \mathcal{P}_{fin}\omega \to \omega$ is a coding function for finite subsets of ω. Given $f : \mathcal{P}\omega \to \mathcal{P}\omega$ a continuous function, the action of f on finite sets can be captured by the set

$$\text{Graph } f \;=\; \{\, ([x_1, \ldots, x_n]; y) \;\mid\; y \in f\{x_1, \ldots, x_n\}\} \;\subseteq\; \omega.$$

To see now that f is indeed completely determined by Graph f, note that for any $B \in \mathcal{P}\omega$ we have $f(B) = \{\, y \mid \exists x_1, \ldots, x_n \in B.\ ([x_1, \ldots, x_n]; y) \in \text{Graph } f\}$. This motivates the following definition:

Definition 3.3.1 *Let $(\mathcal{P}\omega, \cdot)$ be the total applicative structure given by*

$$A \cdot B \;=\; \{\, y \mid \exists x_1, \ldots, x_n \in B.\ ([x_1, \ldots, x_n]; y) \in A\}.$$

Note that $(\mathrm{Graph}\ f) \cdot B = f(B)$, but not all $A \in \mathcal{P}\omega$ are of the form Graph f. However, for any A the function $B \mapsto A \cdot B$ is a continuous function $\mathcal{P}\omega \to \mathcal{P}\omega$. In fact we have

Proposition 3.3.2 *The application operation* $\cdot : \mathcal{P}\omega \times \mathcal{P}\omega \to \mathcal{P}\omega$ *is continuous, and* $\mathcal{P}\omega$ *equipped with this application is a PCA.*

Proof See e.g. Section 18.1 of [8]. $\Box$

Technically, the definition of the PCA $\mathcal{P}\omega$ depends on particular choices of encoding functions $(-;-)$ and $[-,\ldots,-]$; and in fact different choices can give rise to non-isomorphic PCAs (see [87]). However, we are really interested in PCAs up to *equivalence* rather than isomorphism, and so the following fact is reassuring:

Proposition 3.3.3 *Suppose* $(-;-)_1, (-;-)_2$ *are two codings for pairs;* $[\ldots]_1, [\ldots]_2$ *are two codings for finite sets; and* $\cdot_1, \cdot_2$ *are the resulting application operations. Then the PCAs* $(\mathcal{P}\omega, \cdot_1)$ *and* $(\mathcal{P}\omega, \cdot_2)$ *are equivalent.*

Proof Define both $\gamma, \delta : \mathcal{P}\omega \longrightarrow\!\!\!\!\!\!+\ \mathcal{P}\omega$ to be the identity relation. Clearly $\gamma\delta = \mathrm{id}$ and $\delta\gamma = \mathrm{id}$, so we just need to show that $\gamma : (\mathcal{P}\omega, \cdot_1) \longrightarrow\!\!\!\!\!\!+\ (\mathcal{P}\omega, \cdot_2)$ and $\delta : (\mathcal{P}\omega, \cdot_2) \longrightarrow\!\!\!\!\!\!+\ (\mathcal{P}\omega, \cdot_1)$ are applicative morphisms. Let $R \in \mathcal{P}\omega$ be the set

$$\{\ ([x_1, \ldots, x_m]_2; ([y_1, \ldots, y_n]_2; z)_2)_2 \ \mid\ z \in \{x_1, \ldots, x_m\} \cdot_1 \{y_1, \ldots, y_n\}\ \}.$$

Suppose $A, B \in \mathcal{P}_{fin}\omega$. It is easy to see that $R \cdot_2 A \cdot_2 B \supseteq A \cdot_1 B$. Conversely, if $z \in R \cdot_2 A \cdot_2 B$ then there exist $x_1, \ldots, x_m \in A$ and $y_1, \ldots, y_n \in B$ such that $([x_1, \ldots, x_m]_2; ([y_1, \ldots, y_n]_2; z)_2)_2 \in R$. So $z \in \{x_1, \ldots, x_m\} \cdot_1 \{y_1, \ldots, y_n\}$, whence $z \in A \cdot_1 B$ since $\cdot_1$ is monotone. Thus $R \cdot_2 A \cdot_2 B = A \cdot_1 B$ for finite A, B, and hence for all A, B since $\cdot_1$ and $\cdot_2$ are continuous. Thus γ is an applicative morphism realized by R. Similarly δ is an applicative morphism. $\Box$

Hence the categories $\mathbf{Mod}(\mathcal{P}\omega)$, $\mathbf{Ass}(\mathcal{P}\omega)$, $\mathbf{RT}(\mathcal{P}\omega)$ are (up to equivalence) independent of the choice of coding functions.

The PCA $\mathcal{P}\omega$ is of limited usefulness in the study of computation since its definition has no "recursion-theoretic" content. However, we can define a PCA

$\mathcal{P}\omega_{re}$ which is a kind of recursive analogue of $\mathcal{P}\omega$. ($\mathcal{P}\omega_{re}$ is called the *r.e. graph model.*) This PCA will play an important role in later chapters.

To define $\mathcal{P}\omega_{re}$, we need to consider *recursive* coding functions. We already know what it means for a pairing function $(-;-)$ to be recursive; we say that a coding $[\ldots] : \mathcal{P}_{fin}\omega \to \omega$ is recursive if there is a partial recursive binary function ψ such that for all $x_1, \ldots, x_n, y \in \omega$ we have

$$\psi([x_1, \ldots, x_n], y) \;=\; [x_1, \ldots, x_n, y].$$

It follows from this that the image of the function $[\ldots]$ is an r.e. set, and that other natural operations and predicates on finite sets are recursive. For example, there are partial recursive functions *member, size* such that for all $x_1, \ldots, x_n, y$ we have

$$\begin{aligned} \mathit{member}\,(y, [x_1, \ldots, x_n]) \;&=\; \begin{cases} 0 & \text{if } y = x_i \text{ for some } i \\ 1 & \text{otherwise,} \end{cases} \\ \mathit{size}\,([x_1, \ldots, x_n]) \;&=\; \text{card}\,\{x_1, \ldots, x_n\}. \end{aligned}$$

Now suppose $\cdot : \mathcal{P}\omega \times \mathcal{P}\omega \to \mathcal{P}\omega$ is the application operation obtained from the recursive codings $(-;-)$ and $[\ldots]$, and let $\mathcal{P}\omega_{re}$ denote the set of r.e. subsets of ω. It is easy to check that if $A, B \in \mathcal{P}\omega_{re}$ then $A \cdot B \in \mathcal{P}\omega_{re}$; hence we obtain a total applicative structure $(\mathcal{P}\omega_{re}, \cdot)$. Moreover, the elements $k, s \in \mathcal{P}\omega$ may be chosen to be r.e. sets, and so $\mathcal{P}\omega_{re}$ is a PCA. As with $\mathcal{P}\omega$, it turns out that the particular choice of codings is not important:

Proposition 3.3.4 *Suppose $(-;-)_1, (-;-)_2$ are two recursive codings for pairs; $[\ldots]_1, [\ldots]_2$ are two recursive codings for finite sets; and $\cdot_1, \cdot_2$ are the resulting application functions. Then the PCAs $(\mathcal{P}\omega_{re}, \cdot_1)$ and $(\mathcal{P}\omega_{re}, \cdot_2)$ are equivalent.*

Proof It suffices to check that the realizer R for γ given in the proof of Proposition 3.3.3 is an r.e. set. But this is clear since there is a partial recursive ternary function φ such that for all $x_1, \ldots, x_m, y_1, \ldots, y_n, z$ we have

$$\varphi\,([x_1, \ldots, x_m]_2, [y_1, \ldots, y_n]_2, z) \;=\; \begin{cases} 0 & \text{if } z \in \{x_1, \ldots, x_m\} \cdot_1 \{y_1, \ldots, y_n\} \\ 1 & \text{otherwise.} \end{cases} \qquad \square$$

Remark 3.3.5 Note that (for a fixed choice of recursive encodings) the inclusion $\mathcal{P}\omega_{re} \hookrightarrow \mathcal{P}\omega$ preserves application, and hence gives rise to a discrete and projective applicative morphism.

3.3.2 Applicative morphisms involving $\mathcal{P}\omega_{re}$

We now consider the relationships between $\mathcal{P}\omega_{re}$ and the PCAs K_1 and Λ^0/U, where U is a λ-theory. In particular we show that there are an applicative retraction $\mathcal{P}\omega_{re} \to K_1$ and an applicative morphism $\Lambda^0/U \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{P}\omega_{re}$, but that $\mathcal{P}\omega_{re}$ and Λ^0/U are inequivalent. (Note that $\mathcal{P}\omega_{re}$ and K_1 are inequivalent by Corollary 3.1.9.)

First, note that we can represent the numerals in $\mathcal{P}\omega_{re}$ by taking $\overline{n} = \{n\}$. To show that this representation is equivalent to the "canonical" representation given in Definition 1.1.11, it suffices by Proposition 1.1.15 to show the following:

Proposition 3.3.6 *There exist elements succ, rec* $\in \mathcal{P}\omega_{re}$ *such that*

$$succ\ \overline{n} = \overline{n+1}, \qquad rec\ A\ B\ \overline{0} = A, \qquad rec\ A\ B\ \overline{n+1} \simeq B\,\overline{n}(rec\ A\ B\ \overline{n}).$$

Proof Take $succ = \{([n]; n+1) \mid n \in \omega\}$; clearly $succ$ is an r.e. set. For rec, note first that if $A = \{x_1, \ldots, x_m\}$ and $B = \{y_1, \ldots, y_n\}$ then for each n the set

$$R(A, B, n) \ \equiv\ B \cdot \overline{n} \cdot (B \cdot \overline{n-1} \cdot (\ldots (B \cdot \overline{0} \cdot A) \ldots))$$

is finite; and indeed if $A = \{x_1, \ldots, x_m\}$, $B = \{y_1, \ldots, y_p\}$ and $R(A, B, n) = \{z_1, \ldots, z_q\}$ then $[\vec{z}]$ can be computed recursively from $[\vec{x}]$, $[\vec{y}]$ and n. Now take

$$rec = \{\ ([\vec{x}]; ([\vec{y}]; ([n]; z)))\ \mid\ \vec{x}, \vec{y}, n \in \omega;\ z \in R(\{x_1, \ldots, x_m\}, \{y_1, \ldots, y_p\}, n)\ \}$$

Then rec is an r.e. set. Moreover the two equations for rec are clearly satisfied when A, B are finite, and hence for all A, B by continuity of application. $\Box$

Note in passing that this argument also shows that $\overline{0}, \overline{1}, \ldots$ are a good choice of numerals for $\mathcal{P}\omega$. From this it follows immediately that the applicative morphism $\mathcal{P}\omega_{re} \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{P}\omega$ mentioned in Remark 3.3.5 is decidable.

We now consider the relationship between $\mathcal{P}\omega_{re}$ and K_1.

Proposition 3.3.7 *There is an applicative retraction* $(\delta \dashv \gamma) : \mathcal{P}\omega_{re} \to K_1$.

PROOF Let $\delta : K_1 \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{P}\omega_{re}$ be defined by $\delta(n) = \{\overline{n}\}$; by Proposition 2.4.18 we know that δ is a decidable applicative morphism. Now define $\gamma : \mathcal{P}\omega_{re} \longrightarrow\!\!\!\!\!\!\mapsto K_1$ by $\gamma(A) = \{e \in \mathrm{N} \mid \mathrm{Im}\ \{e\} = A\}$. To see that γ is an applicative morphism, note that the predicates

$$([x_1, \ldots, x_n]; y) \in \mathrm{Im}\ \{e\}, \qquad \{x_1, \ldots, x_n\} \subseteq \mathrm{Im}\ \{e'\}$$

are both semi-decidable (regarded as predicates in $[\vec{x}], y, e, e'$). Hence, since we can recursively enumerate the natural numbers of the form $[x_1, \ldots, x_n]$, the predicate

$$P(y, e, e') \ \equiv\ \exists \vec{x}.\ ([\vec{x}]; y) \in \mathrm{Im}\ \{e\} \wedge x_1, \ldots, x_n \in \mathrm{Im}\ \{e'\}$$

is semi-decidable. Hence we may construct a total recursive function $f : \mathrm{N} \times \mathrm{N} \to \mathrm{N}$ such that for all y, e, e' we have

$$\{f(e, e')\}(y) = y \ \text{ if } P(y, e, e'), \qquad \{f(e, e')\}(y)\!\uparrow \ \text{ otherwise.}$$

Now note that if $A, B \in \mathcal{P}\omega_{re}$ and $e \in \gamma(A)$, $e' \in \gamma(B)$ then $f(e, e') \in \gamma(A \cdot B)$. So from an index for f we may easily obtain a realizer for γ.

Next note that for any $n \in \mathrm{N}$ we have $\mathrm{Im}\ \{\{k\}(n)\} = \overline{n}$; hence k realizes $\mathrm{id} \preceq \gamma\delta$. Conversely, if $\mathrm{Im}\ \{e\} = \overline{n}$ then we may recursively recover n from e by computing $\{e\}(0), \{e\}(1), \ldots$ "in parallel". (More formally, using Kleene's notation we may define $n(e) = U(\mu y.\exists m.T(e, m, y))$.) Hence $\gamma\delta \preceq \mathrm{id}$. Finally, to see that $\delta\gamma \preceq \mathrm{id}$ note that if $A \in \mathcal{P}\omega_{re}$ then $\delta\gamma(A) = \{\overline{e} \mid \mathrm{Im}\ \{e\} = A\}$. Take $R = \{([e]; x) \mid x \in \mathrm{Im}\ \{e\}\}$; clearly R is an r.e. set, and if $\overline{e} \in \delta\gamma(A)$ then $R \cdot \overline{e} = A$. $\square$

Remark 3.3.8 It seems worth observing that an alternative "presentation" of the morphism γ is possible. Define $\gamma'(A) = \{e \in \mathrm{N} \mid \ \mathrm{Dom}\ \{e\} = A\}$; then it is easy to show that γ' is also an applicative morphism and that $\gamma' \sim \gamma$.

We now turn our attention to the relationship between $\mathcal{P}\omega_{re}$ and Λ^0/U. Clearly we now have morphisms in both directions factoring through K_1; however, a more natural morphism to consider is that given by the interpretation of λ-terms in

$\mathcal{P}\omega_{re}$. If Γ is a partial function from variables to elements of $\mathcal{P}\omega_{re}$, the interpretation $[\![M]\!]_\Gamma$ of a term M in the environment Γ is defined inductively as follows:

$$\begin{array}{rcl} [\![x]\!]_\Gamma & = & \Gamma(x) \qquad (x \in \mathrm{Dom}\ \Gamma); \\ [\![MN]\!]_\Gamma & = & [\![M]\!]_\Gamma \cdot [\![N]\!]_\Gamma; \\ [\![\lambda x.M]\!]_\Gamma & = & \{([y_1, \ldots, y_n]; z) \mid z \in [\![M]\!]_{\Gamma(x \mapsto \{y_1, \ldots, y_n\})}\}. \end{array}$$

If M is closed then $[\![M]\!]_\Gamma$ is independent of Γ, so we may write just $[\![M]\!]$.

Proposition 3.3.9 *For U any λ-theory, let $\epsilon_U : \Lambda^0/U \longrightarrow\!\!\!\!\!\!\!\;+\;\; \mathcal{P}\omega_{re}$ be defined by $\epsilon_U(M) = \{[\![N]\!] \mid U \vdash N = M\}$. Then*

(i) ϵ_U is an applicative morphism;

(ii) ϵ_U is discrete iff $U \supseteq \mathcal{B}$;

(iii) ϵ_U is projective if $U \subseteq \mathcal{B}$.

Proof (i) Clearly ϵ_U is realized by $\lambda^* xy.xy$ in $\mathcal{P}\omega_{re}$.

(ii),(iii) follow from the fact that $[\![M]\!] = [\![N]\!]$ iff $\mathrm{BT}(M) = \mathrm{BT}(N)$ (cf. [8, Theorem 19.1.19]). $\square$

Finally we give a proof that $\mathcal{P}\omega_{re} \not\simeq \Lambda^0/U$, at least when U is a *semi-sensible* theory, i.e. U does not equate a solvable and an unsolvable term. (Note that any sensible theory is semi-sensible.) The proof relies on the fact that computations in Λ^0/U are "sequential", whereas $\mathcal{P}\omega_{re}$ allows "parallelism"—it thus provides some insight into the computational nature of the difference between the two PCAs.

Theorem 3.3.10 *Suppose U is a semi-sensible λ-theory. Then the PCAs $\mathcal{P}\omega_{re}$ and Λ^0/U are inequivalent.*

Proof Suppose $\gamma : \Lambda^0/U \longrightarrow\!\!\!\!\!\!\!\;+\;\; \mathcal{P}\omega_{re}$ and $\delta : \mathcal{P}\omega_{re} \longrightarrow\!\!\!\!\!\!\!\;+\;\; \Lambda^0/U$ are single-valued morphisms forming an equivalence. As in Section 3.2.5, let I, Ω denote the equivalence classes in Λ^0/U of $\lambda x.x$, $(\lambda x.xx)(\lambda x.xx)$ respectively. Let $A \subset \omega$ be the set

$$\{([x]; ([\,]; x)) \mid x \in \gamma I\} \;\cup\; \{([\,]; ([x]; x)) \mid x \in \gamma I\} \;\cup\; \{([y]; ([y]; y)) \mid y \in \gamma\Omega\}.$$

Clearly A is r.e. Moreover, it is easy to check that, whatever γI and $\gamma\Omega$ are, we have

$$A \cdot \gamma I \cdot \gamma\Omega \;=\; A \cdot \gamma\Omega \cdot \gamma I \;=\; \gamma I, \qquad A \cdot \gamma\Omega \cdot \gamma\Omega \;=\; \gamma\Omega.$$

Now suppose N realizes δ, P realizes $\mathrm{id} \preceq \delta\gamma$ and Q realizes $\delta\gamma \preceq \mathrm{id}$. Let $M = \lambda xy.Q(N(N(\delta A)(Px))(Py))$. Then in Λ^0/U we have

$$MI\Omega \;=\; M\Omega I \;=\; I; \qquad M\Omega\Omega = \Omega.$$

Now U is semi-sensible so $U \subseteq \mathcal{K}^*$ (see [8, Section 17.1]), so these equalities hold also in $\Lambda^0/\mathcal{K}^*$. By the argument used in the proof of Lemma 3.2.20 it is now easy to construct M' such that in $\Lambda^0/\mathcal{B}$ we have

$$M'I\Omega \;=\; M'\Omega I \;=\; I; \qquad M'\Omega\Omega = \Omega.$$

But this contradicts Berry's sequentiality theorem for the untyped λ-calculus (see [8, Section 14.4]). $\Box$

Remark 3.3.11 In fact this argument shows that there do not exist morphisms $\gamma : \Lambda^0/U \longrightarrow\!\!\!\!\!+\, \mathcal{P}\omega_{re}$ and $\delta : \mathcal{P}\omega_{re} \longrightarrow\!\!\!\!\!+\, \Lambda^0/U$ where γ is projective and $\delta\gamma \sim \mathrm{id}$.

From the point of view of programming language semantics, the above proof provides a useful hint. It suggests that Λ^0/U might yield a good category of domains for modelling *sequential* programming languages (such as PCF), whereas $\mathcal{P}\omega_{re}$ might be useful for modelling languages with parallelism. (This idea appears already in the work of Phoa [72], who showed that there is no morphism in $\mathbf{RT}(\Lambda^0/\mathcal{B})$ corresponding to "parallel or".) We will explore this idea in more detail in Chapter 7.

Chapter 4

Dominances

In the previous chapters we introduced categories such as $\mathbf{Mod}(A)$ and examined them from a pure-mathematical standpoint. In the following chapters we wish to show how these categories can be used to give semantics for programming languages such as PCF. An important aspect of these languages is that they allow *partial functions* to be defined, and permit definitions of functions by *recursion*. In this chapter we introduce the categorical structure that we need in order to model partiality; in the next chapter we will consider recursion.

A clean categorical account of partial maps is provided by Rosolini's theory of *dominances* (see [85]). A dominance in a category $\mathcal{C}$ is essentially a small piece of extra structure on $\mathcal{C}$ that determines a notion of "partial map" in $\mathcal{C}$. The most familiar example is probably the *r.e. subobject classifier* Σ in the effective topos. Phoa [71] has considered dominances in other realizability toposes, in particular $\mathbf{RT}(\mathcal{P}\omega_{re})$ and $\mathbf{RT}(\Lambda^0/\mathcal{B})$; Amadio [6] has also implicitly used a notion of partiality arising from a dominance in $\mathbf{Mod}(D)$ where D is a domain-theoretic model of the λ-calculus. Our aim here is to give a uniform account of these (and other) examples in the setting of an arbitrary realizability model.

The chapter is structured as follows: In Section 4.1 we review Rosolini's theory of dominances and partial maps—the material here is not original. In Section 4.2 we consider dominances in $\mathbf{RT}(A)$, and give an explicit description of the lift functor in this setting. In Section 4.3 we introduce the notion of a *divergence*, an attractive and computationally motivated way of specifying a divergence in

$\mathbf{RT}(A)$ via some extra structure on A. We show how all the above examples of dominances, and many others, arise from divergences.

4.1 Dominances and partial maps

We start with a brief summary of the theory of dominances and partial maps. All the ideas in this section appear in [85] or in [40], though our presentation of them is tailored to suit our purposes. We omit many of the easy proofs.

4.1.1 Classes of admissible monos

Our motivation is to give an account of partial maps in categories such as $\mathbf{Mod}(A)$. As we saw in Chapter 1, $\mathbf{Mod}(A)$ can be regarded as a category in which "all morphisms are computable", and we would like if possible to maintain this intuition when considering *partial* maps.

In the category of sets, a partial function $X \rightharpoonup Y$ is given by a subset $X' \subseteq X$ and a *total* function $X' \to Y$. So a first approximation to the definition of partial map in a general category $\mathcal{C}$ might be the following:

Definition 4.1.1 (Partial morphism) *A* partial morphism $X \rightharpoonup Y$ *in* $\mathcal{C}$ *is given by a pair* (m, f)*, where* $m : X' \rightarrowtail X$ *and* $f : X' \to Y$*. We regard* (m, f) *and* (m', f') *as giving the* same *partial morphism* $X \rightharpoonup Y$ *if there is an isomorphism* $\mathrm{dom}\ m \cong \mathrm{dom}\ m'$ *that identifies* m *with* m' *and* f *with* f'*.*

However, there may be many such partial morphisms that are not intuitively "computable". For instance, in $\mathbf{Mod}(K_1)$ we could take a partial morphism $(m, f) : N \rightharpoonup N$, where $\mathrm{dom}\ m$ corresponded to some *non-r.e.* subset of N. Thus we would like some way of cutting down the class of partial morphisms. One way of doing this is to specify a class $\mathcal{M}$ of subobjects (i.e. a class of monos closed under isomorphisms) giving the domains of the partial morphisms we wish to regard as "admissible".

Definition 4.1.2 (Class of admissible monos) *For any class $\mathcal{M}$ of subobjects in $\mathcal{C}$, let $\mathcal{M}(X,Y)$ denote the set of partial morphisms $(m,f) : X \rightharpoonup Y$ such that $m \in \mathcal{M}$.*

(i) A class $\mathcal{M}$ is representable *if for any object Y there is a morphism $\eta_Y : Y \to Y_\perp$ such that for every X we have a bijective correspondence $\mathcal{M}(X,Y) \cong \mathcal{C}(X,Y_\perp)$, where (m,f) corresponds to g iff the square*

$$\begin{tikzcd} X' \arrow[r, "f"'] \arrow[d, "m"'] & Y \arrow[d, "\eta_Y"] \\ X \arrow[r, "g"] & Y_\perp \end{tikzcd}$$

is a pullback. In this situation we say that (m,f) is a partial map *(relative to $\mathcal{M}$), and that it is* classified *by g.*

(ii) A class $\mathcal{M}$ of subobjects is a class of admissible monos *if it is representable and is closed under composition.*

The definition of representable class is inspired by the tradition in domain theory of representing partial maps by total maps into a "lifted" object. Note at once the following trivial consequences:

Proposition 4.1.3 *Suppose $\mathcal{M}$ is a representable class in $\mathcal{C}$. Then*

(i) $\mathcal{M}$ contains all isomorphisms;

(ii) for all objects Y we have $\eta_Y \in \mathcal{M}$;

(iii) for any Y, the object $Y_\perp$ and morphism η_Y are determined up to isomorphism by their defining property;

(iv) pullbacks of monos in $\mathcal{M}$ exist and are monos in $\mathcal{M}$;

(v) the mapping $Y \mapsto Y_\perp$ canonically extends to a functor $\perp : \mathcal{C} \to \mathcal{C}$ (called the lift functor *associated with $\mathcal{M}$), and the morphisms η_Y form a natural transformation $\eta : \mathrm{id} \to \perp$.* $\square$

In the definition of class of admissible monos, the extra requirement that $\mathcal{M}$ is closed under composition is added to ensure that the partial maps associated with $\mathcal{M}$ form a category. It also has the consequence that the lift functor is *monadic*: for any X the mono $\eta_{X_\perp} \circ \eta_X$ is admissible, and so we obtain a morphism $\mu_X : X_{\perp\perp} \to X_\perp$ classifying the partial map $(\eta_{X_\perp} \circ \eta_X, \mathrm{id}_X)$. It is easy to verify that these form a natural transformation $\mu : \perp^2 \to \perp$, and that $(\perp, \eta, \mu)$ is a monad on $\mathcal{C}$ (we call it the *lift monad* associated with $\mathcal{M}$). Finally, note that any class $\mathcal{M}$ of admissible monos is closed under *binary products*.

In general a representable class $\mathcal{M}$ is a rather large object, so it is natural to ask whether $\mathcal{M}$ can be determined by a smaller piece of structure on $\mathcal{C}$. It is easy to see that this is indeed the case:

Proposition 4.1.4 *Suppose $\mathcal{C}$ has a terminal object, and $\mathcal{M}$ is a representable class of subobjects in $\mathcal{C}$. Then $m \in \mathcal{M}$ iff m arises as a pullback of η_1.*

Proof Suppose $(m : X \to Y) \in \mathcal{M}$, and let u denote the unique morphism $X \to 1$. Then by Definition 4.1.2(i) there is a unique morphism g such that the diagram

$$\begin{tikzcd} X \arrow[r, "u"'] \arrow[d, "m"'] & 1 \arrow[d, "\eta_1"] \\ Y \arrow[r, "g"] & 1_\perp \end{tikzcd}$$

is a pullback. Conversely, if such a morphism g exists then $m \in \mathcal{M}$, since $\eta_1 \in \mathcal{M}$ and $\mathcal{M}$ is closed under pullbacks. □

We can thus regard η_1 as the "generic" subobject in $\mathcal{M}$. In recognition of its special role we write Σ for the object $1_\perp$, and $\top : 1 \to \Sigma$ for the morphism η_1. If m and g are such that the above square is a pullback then we say that g *classifies* the mono m.

4.1.2 Pre-dominances and dominances

We have seen how a representable class of subobjects is completely determined by the morphism η_1. We now ask whether we can characterize the morphisms in $\mathcal{C}$ that arise as η_1 for some representable class, or indeed for some class of admissible monos. We will give a precise answer to these questions in the case that $\mathcal{C}$ is a *locally cartesian-closed category*. We first recall the definition:

Definition 4.1.5 *A category $\mathcal{C}$ is* locally cartesian-closed *if it has finite limits and, for every $f : X \to Y$ in $\mathcal{C}$, the pullback functor $f^* : \mathcal{C}/Y \to \mathcal{C}/X$ has a right adjoint Π_f.*

We first give a simple characterization of the representable classes in a locally cartesian-closed category $\mathcal{C}$:

Definition 4.1.6 (Pre-dominance) *Let $\top : 1 \rightarrowtail \Sigma$ be an arbitrary mono in $\mathcal{C}$. We say $(\Sigma, \top)$ is a* pre-dominance *in $\mathcal{C}$ if, for every pair $f, g : X \rightrightarrows \Sigma$ in $\mathcal{C}$, $f^*(\top) \cong g^*(\top)$ implies $f = g$.*

Proposition 4.1.7 *For any pre-dominance $(\Sigma, \top)$, the class*

$$\mathcal{M}(\top) \;=\; \{m \mid m \textit{ is a pullback of } \top\}$$

is representable. Moreover, the mapping $\top \mapsto \mathcal{M}(\top)$ gives a bijection between (isomorphism classes of) pre-dominances and representable classes of subobjects in $\mathcal{C}$.

Proof Assume $(\Sigma, \top)$ is a pre-dominance. For each $Y \in \mathcal{C}$ define $Y_\perp$ via $(Y_\perp \to \Sigma) = \Pi_\top(Y \to 1)$, and let $\eta_Y : Y \to Y_\perp$ be the morphism corresponding under the adjunction $\top^* \dashv \Pi_\top$ to

$$\mathrm{id} : \;\top^*(Y \to 1 \xrightarrow{\top} \Sigma) \;\longrightarrow\; (Y \to 1).$$

Some routine diagram-chasing is required to check that $Y_\perp$ indeed classifies the partial maps $X \rightharpoonup Y$ in $\mathcal{P}$; moreover we clearly have $\eta_1 = \top$. Conversely, if $\mathcal{M}$ is any representable class it is easy to see from the definition that $(1_\perp, \eta_1)$ is a pre-dominance. Finally, by Proposition 4.1.4 we have $\mathcal{M}(\eta_1) = \mathcal{M}$. $\Box$

Thus if $(\Sigma, \top)$ is a pre-dominance then we obtain a lift functor $\perp$ and a natural transformation η by Proposition 4.1.3(v). We say m represents a Σ-*subobject* if $m \in \mathcal{M}(\top)$. We can now characterize precisely the pre-dominances in $\mathcal{C}$ that give rise to classes of admissible monos:

Definition 4.1.8 (Dominance) *Suppose $(\Sigma, \top)$ is a pre-dominance. We say $(\Sigma, \top)$ is a* dominance *if the composite morphism $\eta_\Sigma \circ \top : 1 \to \Sigma_\perp$ is a Σ-subobject (i.e. "the generic composition of Σ-subobjects is a Σ-subobject").*

Proposition 4.1.9 *Suppose $(\Sigma, \top)$ is a pre-dominance, and $\mathcal{M}(\top)$ is defined as above. Then $\mathcal{M}(\top)$ is a class of admissible monos iff $(\Sigma, \top)$ is a dominance. Thus we have a bijection between (isomorphism classes of) dominances and classes of admissible monos in $\mathcal{C}$.*

Proof Note that $\top, \eta_\Sigma$ are Σ-subobjects, and so if $\mathcal{M}(\top)$ is a class of admissible monos then $\eta_\Sigma \top$ is a Σ-subobject. Conversely, suppose $\eta_\Sigma \circ \top$ is a Σ-subobject; we need to show that if $m : X \to Y,\ p : Y \to Z$ are both Σ-subobjects then so is pm. But by Definition 4.1.2(i) we obtain a diagram

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "m"'] & 1 \arrow[r] \arrow[d, "\top"] & 1 \arrow[dd, "\top"] \\
Y \arrow[r] \arrow[d, "p"'] & \Sigma \arrow[d, "\eta_\Sigma"] & \\
Z \arrow[r] & \Sigma_\perp \arrow[r] & \Sigma
\end{tikzcd}$$

in which all three rectangles are pullbacks; hence the outer square is a pullback, and so $pm \in \mathcal{M}(\top)$. $\square$

We will often abuse language and say "Σ is a (pre-)dominance" when the morphism $\top$ is evident. By our earlier remarks, a dominance Σ in $\mathcal{C}$ thus gives rise to a lift monad $(\perp, \eta, \mu)$ on $\mathcal{C}$. Note that for any class of admissible monos the object $1_\perp$ is a dominance even if $\mathcal{C}$ is not locally cartesian-closed.

We will sometimes have occasion to refer to the following natural *order* relation on (pre-)dominances in $\mathcal{C}$: if $(\Sigma_1, \top_1)$ and $(\Sigma_2, \top_2)$ are pre-dominances, we take $\Sigma_1 \preceq \Sigma_2$ iff $\top_1$ is a Σ_2-subobject. Clearly $\preceq$ is a preorder, and if $\Sigma_1 \preceq \Sigma_2 \preceq \Sigma_1$ then $\Sigma_1 \cong \Sigma_2$. We also have the following simple facts:

Proposition 4.1.10 *Suppose Σ_1, Σ_2 are pre-dominances in $\mathcal{C}$, and $\mathcal{M}_1, \mathcal{M}_2$ are the corresponding representable classes. Then*

(i) $\Sigma_1 \preceq \Sigma_2$ iff $\mathcal{M}_1 \subseteq \mathcal{M}_2$.

(ii) If $\Sigma_1 \preceq \Sigma_2$ then the morphism $h : \Sigma_1 \to \Sigma_2$ classifying $\top_1$ is mono.

Proof (i) If $\Sigma_1 \preceq \Sigma_2$ then clearly all pullbacks of $\top_1$ are pullbacks of $\top_2$. Conversely if $\mathcal{M}_1 \subseteq \mathcal{M}_2$ then $\top_1, \in \mathcal{M}_2$ so $\top_1$ is a Σ_2-subobject.

(ii) Suppose $hf = hg$ for $f, g : X \rightrightarrows \Sigma_1$. Then hf, hg classify the same Σ_2-subobject of X, whence f, g classify the same Σ_1-subobject of X, so $f = g$. $\Box$

We conclude this section with some simple examples of dominances. Many more examples will be given in Section 4.3.

Examples 4.1.11 (i) For any $\mathcal{C}$, there is a dominance given by $\Sigma = 1$—we call this the *trivial* dominance. This is the least dominance on $\mathcal{C}$ with respect to the ordering $\preceq$: the admissible monos are precisely the isomorphisms, so the partial maps we obtain are just the total ones.

(ii) At the other extreme, suppose $\mathcal{C}$ is a topos, Ω its subobject classifier (see e.g. [45, Chapter 1]). Then the morphism $true : 1 \to \Omega$ gives a dominance. This is the greatest dominance on $\mathcal{C}$: all monos are admissible, and all partial morphisms are partial maps.

(iii) Suppose $\mathcal{C}$ has binary sums. Let $\Sigma = 1 + 1$ and take $\top$ to be the left inclusion; it is easy to verify that this is a dominance. A mono $X \rightarrowtail Y$ is a Σ-subobject if it is "classified" by a morphism $Y \to 1 + 1$; hence this dominance is sometimes known as the *decidable subobject classifier.*

(iv) The motivating example: Let $\mathcal{C} = \mathbf{Mod}(K_1)$, and define Σ by

$$|\Sigma| = \{\top, \bot\}, \qquad \|\top\| = \{n \mid \{n\}(0)\downarrow\}, \qquad \|\bot\| = \{n \mid \{n\}(0)\uparrow\}.$$

Let $\top : 1 \to \Sigma$ be the morphism that picks out $\top$. We will show in Example 4.2.9(ii) below that $(\Sigma, \top)$ is a dominance. Moreover, the Σ-subobjects of N correspond precisely to the r.e. subsets of N, and the Σ-partial maps $N \rightharpoonup N$ in $\mathcal{C}$ are just the partial recursive functions. Thus in this case it appears that we have successfully captured our intuitive idea of a "computable partial map".

4.2 Dominances in realizability models

We now turn to discuss dominances within realizability toposes. Our aim is to give a concrete characterization of dominances in terms of realizers, and to give an explicit description of the lift functor arising from a pre-dominance.

In fact we will restrict attention to dominances $(\Sigma, \top)$ where Σ is an *assembly*. One reason for doing this is obviously that such dominances are easy to work with. Another reason is that assemblies are the objects which we can understand informally as "datatypes": recall from Chapter 1 that if X is an assembly then $|X|$ represents the set of values of a datatype and $\|x\|_X$ represents the set of "machine representations" of the value x. We certainly want the object $1_\perp$ to be a "datatype" (corresponding e.g. to the type of partial functions from the unit type to itself)—in other words Σ should be an assembly. Note that we are thus excluding the dominance given in Example 4.1.11(ii), since the subobject classifier Ω in $\mathbf{RT}(A)$ is not an assembly.

First, recall that $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$ and $\mathbf{RT}(A)$ are all locally cartesian-closed (see e.g. [74]); moreover, the inclusions $\mathbf{Mod}(A) \overset{J}{\hookrightarrow} \mathbf{Ass}(A) \overset{I}{\hookrightarrow} \mathbf{RT}(A)$ preserve the locally cartesian-closed structure (that is, they preserve finite limits and commute with Π-functors up to natural isomorphism). Hence we obtain the following pleasant facts:

Proposition 4.2.1 *(i) Suppose given* $\top : 1 \to \Sigma$ *in* $\mathbf{Mod}(A)$*. Then* $(\Sigma, \top)$ *is a dominance in* $\mathbf{Mod}(A)$ *iff* $(J\Sigma, J\top)$ *is a dominance in* $\mathbf{Ass}(A)$*.*

(ii) Suppose given $\top : 1 \to \Sigma$ *in* $\mathbf{Ass}(A)$*. Then* $(\Sigma, \top)$ *is a dominance in* $\mathbf{Ass}(A)$ *iff* $(I\Sigma, I\top)$ *is a dominance in* $\mathbf{RT}(A)$*.*

Proof (i) First note that $(\Sigma, \top)$ is a pre-dominance iff $(J\Sigma, J\top)$ is a pre-dominance, as J is full and faithful and preserves pullbacks. Let $\perp : \mathbf{Mod}(A) \to \mathbf{Mod}(A)$ and $\perp' : \mathbf{Ass}(A) \to \mathbf{Ass}(A)$ be the lift functors obtained from Σ and $J\Sigma$ respectively, and let $\eta : \mathrm{id} \to \perp$ and $\eta' : \mathrm{id} \to \perp'$ be the corresponding natural transformations. Note that $J \circ \Pi_\top \cong \Pi_{J\top} \circ J$, and so by the constructions of $\perp$

and η in the proof of Proposition 4.1.7(ii) we see that $J \circ \bot \cong \bot' \circ J$ and $J\eta \cong \eta'_J$. Hence in particular we have $J(\Sigma_\bot) = (J\Sigma)_{\bot'}$ and $J(\eta_\Sigma \circ \top) = \eta'_{J\Sigma} \circ J\top$. But J is full and preserves and reflects pullbacks, so $\eta_\Sigma \circ \top$ is a pullback of $\top$ iff $\eta'_{J\Sigma} \circ J\top$ is a pullback of $J\top$. Thus $(\Sigma, \top)$ is a dominance iff $(J\Sigma, J\top)$ is a dominance. The proof of (ii) is precisely similar. $\Box$

Remark 4.2.2 In [72] Phoa gives a short proof that a certain object is a dominance in $\mathbf{Ass}(\Lambda^0/\mathcal{B})$, and then a much longer (concrete) proof that it is in fact a dominance in $\mathbf{RT}(\Lambda^0/\mathcal{B})$. The above proposition shows that the latter can be bypassed, for quite general abstract reasons.

We now go about characterizing the dominances in $\mathbf{Ass}(A)$. First note that if $(\Sigma, \top)$ is a *pre-dominance* in $\mathbf{Ass}(A)$ then card $|\Sigma| \leq 2$, since otherwise we can find distinct morphisms $f, g : 1 \rightrightarrows \Sigma$ with $f^*\top = g^*\top = (0 \rightarrowtail 1)$. Moreover, if card $|\Sigma| < 2$ then Σ is the trivial dominance. So for any non-trivial pre-dominance we have card $|\Sigma| = 2$. This suggests the following definition.

Definition 4.2.3 *A* pre-dominance *on a PCA A consists of a pair (T, U) of non-empty subsets of A. To each pre-dominance (T, U) we associate an object $\Sigma \in$ $\mathbf{Ass}(A)$ and a morphism $\top : 1 \to \Sigma$ as follows:*

$$|\Sigma| = \{\top, \bot\}, \qquad \|\top\| = T, \qquad \|\bot\| = U, \qquad \top(*) = \top.$$

We write $(T, U) \preceq (T', U')$ if there exists $r \in A$ such that $r(T) \subseteq T'$ and $r(U) \subseteq U'$ (i.e. for all $a \in T$ we have $ra \in T'$ etc.). We also write $(T, U) \sim (T', U')$ if $(T, U) \preceq (T', U') \preceq (T, U)$.

The letters T, U here are chosen to stand for "top" and "undefined". It is easy to see that in this definition the pair $(\Sigma, \top)$ is always a pre-dominance, since for any subobject $X' \rightarrowtail X$ in $\mathbf{Ass}(A)$ there is a unique set-theoretic function $|X| \to |\Sigma|$ classifying m. Moreover, the above construction clearly induces an equivalence of preorders between pre-dominances on A and non-trivial pre-dominances in $\mathbf{Ass}(A)$.

Next note that if (T, U) is a pre-dominance and $T \cap U = \emptyset$ then the associated object Σ is a modest set; otherwise Σ is a uniform assembly and so $\Sigma \cong \nabla 2$ (see Section 1.2.3). In the latter case we call Σ the *uniform* pre-dominance; note that it is the largest pre-dominance in $\mathbf{Ass}(A)$ with respect to $\preceq$. However, this pre-dominance is rather boring since it yields too many Σ-subobjects: e.g. for any X and any subset $|X'| \subseteq |X|$, the evident regular mono $X' \rightarrowtail X$ gives a Σ-subobject of X.

We now ask: when does a pre-dominance (T, U) give rise to a *dominance* in $\mathbf{Ass}(A)$? In order to answer this question, it is convenient first to give an explicit description of the lift functor associated with (T, U). We need some notation: if $P, Q \subseteq A$ then we write

$$\begin{aligned} P \wedge Q &= \{\langle a, b\rangle \mid a \in P,\ b \in Q\}, \\ P \Rightarrow Q &= \{a \mid \forall b \in P.\ ab \in Q\}. \end{aligned}$$

We also write I for the set $\{i\}$.

Proposition 4.2.4 *Suppose (T, U) is a pre-dominance on A, Σ the corresponding pre-dominance in $\mathbf{Ass}(A)$, and $\perp$ the associated lift functor. Then for any X, the object $X_\perp$ and morphism η_X are given as follows (up to isomorphism):*

$$\begin{aligned} |X_\perp| &= |X| \sqcup \{\perp\}, & \eta_X(x) &= x, \\ \|x \in X_\perp\| &= T \wedge (I \Rightarrow \|x \in X\|), & \|\perp \in X_\perp\| &= U \wedge A. \end{aligned}$$

Proof First note that η_X as defined above is tracked by $\lambda^* x.\langle a, kx\rangle$, where $a \in T$. It suffices to check that the object $X_\perp$ defined above does indeed classify the partial maps into X. So suppose $(m, f) : Y \rightharpoonup X$ is a partial map, where $m : Y' \rightarrowtail Y$ is classified by $e : Y \to \Sigma$. Then m is a regular mono, so without loss of generality we can assume that $m : |Y'| \rightarrowtail |Y|$ is a set-theoretic inclusion, and that $\|y \in Y'\| = \|y \in Y\|$ for all $y \in |Y'|$. Define $g : |Y| \to |X_\perp|$ by

$$g(y) = \begin{cases} f(y) & \text{if } y \in |Y'| \\ \perp & \text{otherwise.} \end{cases}$$

Suppose f, e are tracked by t, u respectively; we claim $g : Y \to X_\perp$ is tracked by $r = \lambda^* a.\,\langle ua, \lambda^* w.ta\rangle$. For if $y \in Y'$ and $a \in \|y\|_Y$ then $e(y) = \top$ and $a \in \|y\|_{Y'}$;

hence $ua \in T$ and $ta \in \|fy\|_X$, and so

$$ra \;=\; \langle ua, \lambda^* w.ta\rangle \;\;\in\;\; T \wedge (I \Rightarrow \|fy\|_X).$$

On the other hand if $y \notin Y'$ and $a \in \|y\|_Y$ then $e(y) = \perp$ so $ua \in U$; but $\lambda^* w.ta\downarrow$ and so $ra \in U \wedge A$.

We now need to check that g is the unique morphism such that

$$\begin{tikzcd}
Y' \arrow[r, "f"'] \arrow[d, "m"'] & X \arrow[d, "\eta_X"] \\
Y \arrow[r, "g"] & X_\perp
\end{tikzcd}$$

is a pullback. But given any $f' : Z \to X$ and $m' : Z \to Y$ such that $\eta_X f' = gm'$, clearly there is a unique function $h : |Z| \to |Y'|$ such that $f' = fh$ and $m' = mh$; and if v tracks m' then v also tracks h. To see that g is unique, just note that $g : |Y| \to |X_\perp|$ is the unique function such that the underlying square in **Set** is a pullback. $\square$

Remarks 4.2.5 (i) The action of the lift functor on morphisms of $\mathbf{Ass}(A)$ is obvious: if $f : X \to Y$, then $f_\perp : X_\perp \to Y_\perp$ is given by $f_\perp(x) = f(x)$, $f_\perp(\perp) = \perp$. If f is tracked by t then $f_\perp$ is tracked by $\lambda^* a.\, \langle \mathit{fst}\ a,\ \lambda^* z.\, t((\mathit{snd}\ a)i)\rangle$.

(ii) The choice of the set I is arbitrary; any non-empty set would do. The role of I is simply to "delay the evaluation" of a realizer for $x \in X$. It is easy to check that if A is *total* then the following simpler description of $X_\perp$ suffices:

$$|X_\perp| \;=\; |X| \sqcup \{\perp\}, \qquad \|x \in X_\perp\| \;=\; T \wedge \|x \in X\|, \qquad \|\perp \in X_\perp\| \;=\; U \wedge A.$$

We are now in a position to characterize the pre-dominances on A that give rise to dominances.

Definition 4.2.6 *A pre-dominance* (T, U) *on* A *is called a* dominance *if the corresponding pre-dominance* Σ *in* $\mathbf{Ass}(A)$ *is a dominance.*

Proposition 4.2.7 *A pre-dominance (T, U) is a dominance iff for some $r \in A$ we have*

$$r(T \wedge (I \Rightarrow T)) \subseteq T, \qquad r(T \wedge (I \Rightarrow U)) \subseteq U, \qquad r(U \wedge A) \subseteq U.$$

Proof By Proposition 4.2.4 we obtain the following description of $\Sigma_\perp$:

$$|\Sigma_\perp| = \{\top, M, \perp\},$$
$$\|\top\| = T \wedge (I \Rightarrow T), \ \ \|M\| = T \wedge (I \Rightarrow U), \ \ \|\perp\| = U \wedge A.$$

Now the morphism $\eta_\Sigma \top : 1 \to \Sigma_\perp$ is given by $\eta_\Sigma \top(*) = \top$, and so a morphism $\mu : \Sigma_\perp \to \Sigma$ is a classifier for $\eta_\Sigma \top$ iff it satisfies $\mu(\top) = \top$, $\mu(M) = \perp$, $\mu(\perp) = \perp$. But a realizer for such a morphism is precisely an element r satisfying the above conditions. □

Remarks 4.2.8 (i) The above characterization of dominances is rather unsightly, and is too close to the definition of dominance in **Ass**(A) to be of much interest in itself. But we will use it to give a much more illuminating (though less general) way of specifying dominances in the next section.

(ii) If A is total, it is easy to see that the above condition can be replaced by a slightly simpler one: there exists $q \in A$ such that

$$q(T \wedge T) \subseteq T, \qquad q(T \wedge U) \subseteq U, \qquad q(U \wedge A) \subseteq U.$$

Examples 4.2.9 (i) The uniform pre-dominance $\nabla 2$ mentioned above is clearly a dominance: in Proposition 4.2.7 we may take $r = ka$ where $a \in T \cap U$.

(ii) We can now verify that the r.e. subobject classifier Σ in **Mod**(K_1) is indeed a dominance (see Example 4.1.11(iv)). Note that the object Σ arises from the pre-dominance (T, U), where $T = \{n \mid \{n\}(0)\downarrow\}$, $U = \{n \mid \{n\}(0)\uparrow\}$. Let

$$r = \lambda^* a. \lambda^* z.\, k(\mathit{fst}\ a\ 0)(\mathit{snd}\ a\ i\ 0).$$

It is easy to see that r satisfies the three conditions of Proposition 4.2.7, and so Σ is a dominance.

It is natural to ask how (pre-)dominances on PCAs interact with the applicative morphisms introduced in Chapter 2. The relevant notion of morphism between PCAs-with-pre-dominance seems to be the following:

Definition 4.2.10 *Suppose (A, T, U) and (B, T', U') are PCAs equipped with pre-dominances. A* dominical morphism *from (A, T, U) to (B, T', U') is an applicative morphism $\gamma : A \twoheadrightarrow B$ such that $(\gamma^+ T, \gamma^+ U) \sim (T', U')$.*

Proposition 4.2.11 *Suppose (A, T, U), (B, T', U') are PCAs equipped with pre-dominances, and suppose $\Sigma, \Sigma', \top, \top', \bot, \bot', \eta, \eta'$ have the obvious meanings. Then the following are equivalent for an applicative morphism $\gamma : A \twoheadrightarrow B$:*

(i) γ is a dominical morphism;

(ii) $\gamma_*(\Sigma) \cong \Sigma'$, via an isomorphism identifying $\gamma_*(\top)$ with $\top'$;

(iii) (If B is total.) $\gamma_* \circ \bot \cong \bot' \circ \gamma_* : \mathbf{Ass}(A) \to \mathbf{Ass}(B)$, via a natural isomorphism identifying $\gamma_* \eta$ with η'_{γ_*}.

Proof (i) $\Leftrightarrow$ (ii): Immediate from the definition of Σ and Σ'.

(iii) $\Rightarrow$ (ii): Trivial, since $\Sigma = 1_\bot$, $\top = \eta_1$ etc. and $\gamma_*(1) \cong 1$.

(i) $\Rightarrow$ (iii): Suppose B is total. Then for any $X \in \mathbf{Ass}(A)$ the objects $Y = \gamma_*(X_\bot)$ and $Z = (\gamma_* X)_{\bot'}$ may be described explicitly as follows:

$$\begin{array}{rclrcl} & & |Y| \;=\; |Z| \;=\; |X| \sqcup \{\bot\}, & & & \\ \|x \in Y\| & = & \gamma^+(T \wedge (I \Rightarrow \|x\|_X)), & \|\bot \in Y\| & = & \gamma^+(U \wedge A), \\ \|x \in Z\| & = & T' \wedge \gamma^+ \|x\|_X, & \|\bot \in Z\| & = & U' \wedge B. \end{array}$$

Suppose r realizes γ, t realizes $(\gamma^+ T, \gamma^+ U) \preceq (T', U')$ and u realizes $(T', U') \preceq (\gamma^+ T, \gamma^+ U)$. Then $\mathrm{id} : |Y| \to |Z|$ is tracked by $\lambda^* b.\, \langle t(r\ \mathit{fst}'\ b), r(r\ \mathit{snd}'\ b) i' \rangle$ where $\mathit{fst}' \in \gamma(\mathit{fst})$, $\mathit{snd}' \in \gamma(\mathit{snd})$, $i' \in \gamma(i)$; moreover $\mathrm{id} : |Z| \to |Y|$ is tracked by $\lambda^* b.\, r(r\ \mathit{pair}'\ (u(\mathit{fst}\ b)))(r\ k'(\mathit{snd}\ b))$ where $\mathit{pair}' \in \gamma(\mathit{pair})$, $k' \in \gamma(k)$. Thus $\gamma_*(X_\bot) \cong (\gamma_* X)_{\bot'}$. It is easy to see that this isomorphism is natural in X and identifies $\gamma_* \eta$ with η'_{γ_*}. $\square$

Curiously, the implication (i) $\Rightarrow$ (iii) in the above proposition seems to fail if B is not total. The problem is that we cannot construct an element $v \in B$ such that $vb\downarrow$ for all $b \in B$ and $v(I \Rightarrow \gamma^+ P) \subseteq \gamma^+(P)$ for all $P \subseteq A$.

4.3 Divergences

In Proposition 4.2.7 we showed that dominances in $\mathbf{Ass}(A)$ correspond to pairs of subsets $T, U \subseteq A$ satisfying certain properties. This characterization of dominances was not very interesting as it was too close to the definition of a dominance. In this section we offer a simpler and more "computational" way of looking at dominances as extra structure on a PCA, and show that it gives rise to many natural examples of dominances. The idea was inspired by a suggestion of Gordon Plotkin; a similar idea appears in [4, Section 3] (also under the name "divergence"!) in the context of models for the lazy λ-calculus.

Let us return to our analogy of "PCAs as abstract machines". To specify a dominance in $\mathbf{RT}(A)$, we want to specify a notion of "partial computation" in A itself. If A is non-total, then of course undefined applications in A give a perfectly good notion of non-terminating computations. But the existence of interesting total combinatory algebras prompts us to seek a more general notion. In general, it ought to be enough to specify which elements of A (if any) correspond to "infinite" or "diverging" computations. This leads us to the following definition.

Definition 4.3.1 (Divergence) *A* divergence *on a PCA A is a subset $D \subseteq A$ such that*

(i) if A is total then $D \neq \emptyset$;

(ii) if $a \in D$ and $ab{\downarrow}$ then $ab \in D$.

The first condition ensures that diverging computations are possible: this condition perhaps looks more natural when rephrased as $\exists\, a, b.\ ab{\downarrow} \Rightarrow\ ab \in D$. The second condition says that if a represents a diverging computation then ab cannot represent a converging computation—a divergence is thus a kind of "weak right ideal". Both these conditions seem to be natural in terms of the "abstract machine" viewpoint. We now show that every divergence gives rise to a dominance:

Definition 4.3.2 *Given any set $D \subseteq A$, we define sets T_D, U_D by*

$$T_D = \{a \mid ai = i\}, \qquad U_D = \{a \mid ai{\downarrow} \Rightarrow ai \in D\}.$$

Proposition 4.3.3 *If D is a divergence then (T_D, U_D) is a dominance on A.*

PROOF First note that $ki \in T_D$ so $T_D \neq \emptyset$, and if a, b are such that $ab{\downarrow} \Rightarrow ab \in D$ then $\lambda^* x.ab \in U_D$ so $U_D \neq \emptyset$. Thus (T_D, U_D) is a pre-dominance. Now let $r = \lambda^* xz.(\mathit{fst}\, x\, i)(\mathit{snd}\, x\, i\, i)$. We check that r satisfies the three conditions of Proposition 4.2.7. First, if $a \in T_D$ and $b \in (I \Rightarrow T_D)$ then $ai = bii = i$ so $r\langle a, b\rangle = \lambda^* z.(ai)(bii) \in T_D$; thus $r(T_D \wedge (I \Rightarrow T_D)) \subseteq T_D$. Next, if $a \in T_D$ and $b \in (I \Rightarrow U_D)$ then $ai = i$ and $bii{\downarrow} \Rightarrow bii \in D$, so $\lambda^* z.(ai)(bii) \in U_D$; thus $r(T_D \wedge (I \Rightarrow U_D)) \subseteq U_D$. Finally, if $a \in U_D$ and $b \in A$ then $ai{\downarrow} \Rightarrow ai \in D$; hence $(ai)(bii){\downarrow} \Rightarrow (ai)(bii) \in D$. Thus $\lambda^* z.(ai)(bii) \in U_D$, and so $r(U_D \wedge A) \subseteq U_D$. $\Box$

Remark 4.3.4 If A is total, we could more simply define $T_D = \{i\}$, $U_D = D$. It is easy to see that this gives a dominance equivalent to that given by the above definition.

We now show how various known dominances arise from divergences:

Examples 4.3.5 (i) The set A itself is trivially a divergence on A; in fact, if D is a divergence on A then $D = A$ iff $i \in D$. Hence $ki \in T_A \cap U_A$, and so (T_A, U_A) is the uniform dominance on A, giving rise to the dominance $\nabla 2$ in $\mathbf{RT}(A)$ (see Example 4.2.9(i) above).

(ii) If A is non-total, clearly $\emptyset$ is a divergence on A—this corresponds to the "natural" notion of diverging computation in the PCA. The corresponding dominance is easily seen to be equivalent to that defined by

$$T = \{a \mid ai{\downarrow}\}, \qquad U = \{a \mid ai{\uparrow}\}.$$

In the case $A = K_1$, this is in turn equivalent to the r.e. dominance Σ (see Example 4.2.9(ii) above). Thus Σ is in some sense the natural dominance associated with K_1.

(iii) Let $A = \Lambda^0/S$ where S is any semi-sensible λ-theory, and let D be the set of (equivalence classes of) *unsolvable* terms, i.e. terms with no head normal form. Clearly D is a divergence, since if M is unsolvable then so is MN. In the case $S = \mathcal{B}$, by Remark 4.3.4 we see that the associated dominance is given by $T_D = \{i\}, U_D = \{\Omega\}$. This corresponds to the dominance on $\mathbf{RT}(\Lambda^0/\mathcal{B})$ considered by Phoa in [71,72].

(iv) Let A be the graph model $\mathcal{P}\omega$ (or its r.e. submodel), and take $D = \{\emptyset\}$. From the definition of application in A it is clear that $\emptyset \cdot X = \emptyset$ for any $X \in A$, so D is a divergence. The corresponding dominance is easily seen to be equivalent to that defined by $T_D = \{\{0\}\}$, $U_D = \{\emptyset\}$—this gives rise to the dominance on $\mathbf{RT}(A)$ considered by Phoa in [71,73].

(v) The following situation is considered by Amadio in [6]. Let A be a directed-complete partial order such that the partial function space $(A \rightharpoonup A)$ is a retract of A. Then A has the structure of a PCA. In [6], a *partial morphism* $X \rightharpoonup Y$ in $\mathbf{Mod}(A)$ is (essentially) defined to be a partial function $f : |X| \to |Y|$ such that there exists $r \in A$ satisfying

$$\forall x \in |X|. \forall a \in \|x\|, (f(x)\downarrow \wedge ra \in \|f(x)\|) \vee (f(x)\uparrow \wedge ra\uparrow).$$

It is easy to see that these correspond precisely to the partial maps for the dominance arising from the empty divergence on A. This gives us a more categorical way of understanding this notion of partial morphism.

In fact the notion of divergence seems to embrace all known examples of dominances that are interesting from the point of view of semantics. However, not all dominances on PCAs arise from divergences. For example, consider the dominance 2 in $\mathbf{Mod}(K_1)$ (cf. Example 4.1.11(iii)). First note that this corresponds to the dominance on K_1 given by $T = \{true\}$, $U = \{false\}$. Suppose this is equivalent to (T_D, U_D) for some divergence D; then there exists r such that $rT_D \subseteq T$ and $rU_D \subseteq U$. Define $t = \lambda^* x.r(\lambda^* y.ki(xi))$; then for all a we have $ta = true$ if $ai\downarrow$ and $ta = false$ if $ai\uparrow$, contradicting the undecidability of the halting problem. Thus this dominance does not arise from a divergence.

We can recast some of the definitions and results from Section 4.2 in terms of divergences—this will make it simpler to perform calculations for particular models. First we note that the order relation between dominances can be expressed at the level of divergences. If D_1, D_2 are divergences on A, we write $D_1 \preceq D_2$ if there exists $r \in A$ such that

$$ri \;=\; i, \qquad a \in D_1 \;\Rightarrow\; (ra\!\downarrow \Rightarrow\; ra \in D_2).$$

We also write $D_1 \sim D_2$ if $D_1 \preceq D_2 \preceq D_1$.

Proposition 4.3.6 *Suppose D_1, D_2 are divergences on A. Then*

(i) $D_1 \preceq D_2$ iff $(T_{D_1}, U_{D_1}) \preceq (T_{D_2}, U_{D_2})$;

(ii) $D_1 \sim D_2$ iff $(T_{D_1}, U_{D_1}) \sim (T_{D_2}, U_{D_2})$.

Proof (i) If r is a realizer for $D_1 \preceq D_2$, take $t = \lambda^* ab.r(ab)$; then $tT_{D_1} \subseteq T_{D_2}$ and $tU_{D_1} \subseteq U_{D_2}$. Conversely, if t realizes $(T_{D_1}, U_{D_1}) \preceq (T_{D_2}, U_{D_2})$, take $r = \lambda^* a.t(ka)i$. Then $ki \in T_{D_1}$ so $t(ki) \in T_{D_2}$ so $ri = i$; and if $a \in D_1$ then $ka \in U_{D_1}$ so $t(ka) \in U_{D_2}$, and so if $ra\!\downarrow$ then $ra \in D_2$. Thus $D_1 \preceq D_2$. (ii) is immediate. □

Note that A and $\emptyset$ (if A is non-total) are respectively the greatest and least divergences with respect to $\preceq$. Next we show how the lift functor can be described directly in terms of a divergence:

Proposition 4.3.7 *Suppose D is a divergence on A, $\perp$ the lift functor on* $\mathbf{Ass}(A)$ *arising from the corresponding dominance. Then for any assembly X, the object $X_\perp$ is given as follows (up to isomorphism):*

$$\begin{aligned} \|x \in X_\perp\| &= \{a \mid \mathit{fst}(ai) = i,\ \mathit{snd}(ai) \in \|x \in X\|\}, \\ \|\perp \in X_\perp\| &= \{a \mid \mathit{fst}(ai)\!\downarrow \Rightarrow \mathit{fst}(ai) \in D\}. \end{aligned}$$

Proof Write $X'_\perp$ for the assembly defined as in Proposition 4.2.4 (where $T = T_D$, $U = U_D$). Explicitly we have

$$\begin{aligned} \|x \in X'_\perp\| &= \{\langle b, c\rangle \mid bi = i,\ ci \in \|x \in X\|\}, \\ \|\perp \in X'_\perp\| &= \{\langle b, c\rangle \mid bi\!\downarrow \Rightarrow\ bi \in D\}. \end{aligned}$$

Let $r = \lambda^* a.\, \langle \lambda^* w.\mathit{fst}(ai), \lambda^* w.\mathit{snd}(ai) \rangle$; then for all $z \in |X_\perp|$ we have $r \|z\|_{X_\perp} \subseteq \|z\|_{X'_\perp}$. Conversely, let $t = \lambda^* aw.\, \langle (\mathit{fst}\, a)i, (\mathit{snd}\, a)i \rangle$; then for all $z \in |X_\perp|$ we have $t \|z\|_{X'_\perp} \subseteq \|z\|_{X_\perp}$. Hence $X_\perp \cong X'_\perp$. $\square$

Remark 4.3.8 A much simpler description of $X_\perp$ may be given if A is total:

$$\|x \in X_\perp\| \;=\; I \wedge \|x \in X\|, \qquad \|\perp \in X_\perp\| \;=\; D \wedge A.$$

It is easy to check that this is isomorphic to the object $X_\perp$ defined above.

We conclude this chapter with a few more examples of divergences. Some of these seem very natural and could yield useful categories of domains, while others are quite pathological (they might be useful as counterexamples). Our intention is not to be comprehensive but merely to indicate the generality of the concept.

Examples 4.3.9 (i) Let A be the term model Λ^0/ℓ for the lazy λ-calculus (see Paragraph 3.2.3). Let $D \subseteq A$ be the set of (ℓ-classes of) terms with no head normal form, and D' the set of terms with no *weak* head normal form (i.e. terms whose head reduction sequence contains no term of the form $\lambda x.M$). Then D, D' are both right ideals and hence divergences, and clearly $D' \preceq D$. To see that $D \not\preceq D'$, let E be the canonical domain-theoretic model of the lazy λ-calculus considered in [4, Section 4]. Note that E has a bottom element $\perp_E$ and a top element $\top_E$; a term M denotes $\perp_E$ iff M has no whnf, and the unsolvable term Yk denotes $\top_E$. Thus if M realized $D \preceq D'$ we would have $E \models Mi = i$ and $E \models M(Yk) = \Omega$; but this is impossible since application in E is monotone.

(ii) Let A be the term model $\mathcal{V}^0/V$ be the call-by-value λ-calculus (see Paragraph 3.2.4), and let $D \subseteq V$ be the set of terms M such that $M \cdot N \uparrow$ for all $N \in \mathcal{V}^0$. Then D is trivially a divergence. Moreover it is easy to see that $D \sim \emptyset$, since $\emptyset \preceq D$ is realized by i and $D \preceq \emptyset$ is realized by $\lambda x.xi$.

(iii) For many PCAs with a "top" element, we can invert our usual intuition and take undefined to be represented by top rather than bottom. More precisely, let A be any PCA equipped with a preorder $\leq$ compatible with the application:

$$a \leq a' \;\wedge\; b \leq b' \;\wedge\; ab\!\downarrow \;\Rightarrow\; a'b'\!\downarrow \;\wedge\; ab \leq a'b',$$

and suppose there exists $\top \in A$ such that $a \leq \top$ for all $a \in A$. Let $D = \{a \in A \mid \top \leq a\}$. Then D is a divergence: given $a \in D$ and $b \in A$ we have $ab \geq \top b \geq (k\top)b = \top$, so $ab \in D$. We mention a few specific examples. If $A = \Lambda^0/\ell$ we may take $\leq$ to be the preorder induced by the order relation on the model E (see (i) above); in this case D is the set of *infinite-order unsolvables*, i.e. the set of terms M such that for all n the head reduction sequence of M contains a term $\lambda x_1 \ldots x_n.M'$ (see [4, Section 2]). If $A = \mathcal{V}^0/V$ we may take $\leq$ to be the observational preorder considered in Paragraph 3.2.5. Using the context lemma it can be shown that Yk is a top element for this preorder; we thus obtain the divergence

$$D = \{N \mid \forall P_1, \ldots, P_n.\ N \cdot P_1 \cdot \ldots \cdot P_n\!\downarrow\}.$$

If $A = \mathcal{P}\omega$ or $\mathcal{P}\omega_{re}$ we have the obvious order relation $\subseteq$; from this we obtain $D = \{\omega\}$. It is easy to show that none of these divergences are equivalent to any divergences considered above.

(iv) The union of two divergences on A is clearly a divergence. Using this fact we can construct bizarre examples such as the divergence $\{\emptyset, \omega\}$ on $\mathcal{P}\omega_{re}$.

(v) Suppose we have $a \in A$ such that $ab = a$ for all b; then clearly $\{a\}$ is a divergence. For instance, in any PCA at all we have $Zkb = Zk$ for all b, so $\{Zk\}$ is a divergence. A more artificial example: suppose $A = \mathcal{P}\omega$ or $\mathcal{P}\omega_{re}$, where the coding functions $(-;-)$ and $[-, \ldots, -]$ are chosen so that $([\,], 0) = 0$ (it is easy to see that we can construct recursive coding functions with this property). Then $\{\{0\}\}$ is a divergence on A.

Chapter 5

Completeness and fixed points

We now have a good theory of partial maps in our categories of interest. However, in order to interpret languages such as PCF we also need to be able to model *recursion*. In particular, we would like to be able to obtain fixed points of morphisms $X \to X$ for certain non-trivial objects X. Of course this would be impossible in a topos such as **Set**, but one of the beautiful features of realizability toposes is that very often we can do precisely this. In this chapter we develop some machinery for modelling recursion and fixed points in the general setting of a realizability topos with dominance. The material in this chapter represents joint work with Alex Simpson, to whom I am grateful for many of the ideas and for much valuable discussion.

Our work here may be regarded as a contribution to *synthetic domain theory* (cf. [40,97,81]). The basic idea is the same as in these papers. Starting with a topos, we identify a subcategory $\mathcal{D}$ whose objects can be thought of as "predomains". We then show—under certain axioms—that $\mathcal{D}$ contains all the objects we need for our denotational semantics, and hence that these objects have the fixed point property. Various candidates for the category $\mathcal{D}$ have been proposed, such as the *replete* objects [40,97] and the Σ-*CPOs* [81], the latter being an abstraction of the category of *extensional PERs* in $\mathbf{Mod}(K_1)$ [26]. Here we propose another alternative: the category of *well-complete* objects. One advantage of this category is that it allows us to give a uniform treatment of both "sequential" and "parallel" models of computation (cf. Theorem 3.3.10), whereas the papers cited above all make use of axioms that fail in “sequential” models such as $\mathbf{RT}(\Lambda^0/\mathcal{B})$.

Our approach also differs from that of [40,97,81] in two other respects. Firstly, it is less general—we work at the level of generality of a realizability topos rather than an arbitrary topos satisfying certain axioms. Nevertheless, we believe that the study of this class of concrete models can help guide the search for suitable axioms for a more abstract theory. Secondly, we have chosen to work “externally” with assemblies and realizers rather than exploit the internal logic of the topos, even where the latter might have been more elegant. We hope that by so doing we afford a firmer grasp of the situation to readers unversed in the mysteries of topos theory and constructive reasoning.

The chapter is structured as follows: In Section 5.1 we define objects $\omega, \overline{\omega}$ playing the role of a generic chain and its completion—these allow us to talk about ω-chains and limits in $\mathbf{RT}(A)$. Using these notions we define our category $\mathcal{D}$ of well-complete objects. In Section 5.2 we introduce a simple “Completeness Axiom” saying that the object 2 is well-complete. We show that this single axiom suffices to develop all the properties of $\mathcal{D}$ that we require. In Section 5.3 we show that many of the examples of dominances introduced in Chapter 4 do indeed satisfy the Completeness Axiom; in particular, we see that the models considered by Phoa [71] and Amadio [6] can be recovered as instances of our general theory. In Section 5.4 we obtain the fixed point property for well-complete domains, and prepare the ground for our work in denotational semantics in the next chapter.

The material in Sections 5.1 and 5.2 in particular is joint work with Alex Simpson. It is hard to say precisely who was responsible for which of the ideas here, though I accept sole responsibility for the details of the proofs as they appear here. The material in Section 5.3 are mainly due to me. Section 5.4 essentially consists of an adaptation of ideas found e.g. in [71] and [28].

5.1 ω-chains and limits

In this section we construct an object ω playing the role of an infinite ascending series of points in classical domain theory, and an object $\overline{\omega}$ playing the role of its chain completion. The idea is that morphisms $c : \omega \to X$ can then be regarded as "ascending chains in X", and such a chain has a limit point iff c extends to a morphism $\overline{\omega} \to X$. We then define the category of well-complete objects—essentially the objects X such that $X_\perp$ is chain-complete in a suitable sense.

5.1.1 ω-chains in $\mathbf{Ass}(A)$

We work in the setting of an arbitrary PCA A equipped with a dominance (T, U). Our first goal is to construct an object ω that plays the role of an infinite ascending chain. By analogy with the classical situation we might expect such an object to satisfy $\omega \cong \omega_\perp$, and moreover to be in some sense the *initial* solution to this equation.

Suppose we visualize Σ as a poset with $\perp \leq \top$ (we will make this intuition precise in Section 5.4). Then one way of obtaining an infinite ascending series of points is to consider morphisms $p : N \to \Sigma$ such that $p(m) \geq p(m+1)$ for all m. Formally, for each n let us define $p_n : N \to \Sigma$ by

$$p_n(m) \;=\; \begin{cases} \top & \text{if } m < n \\ \perp & \text{if } m \geq n. \end{cases}$$

(it is easy to give realizers for the p_n). Then $p_0 \leq p_1 \leq \ldots$ in the pointwise order. This suggests the following definition:

Definition 5.1.1 *Let $\omega \in \mathbf{Mod}(A)$ be the object defined by*

$$|\omega| \;=\; \mathrm{N}, \qquad \|n \in \omega\| \;=\; \|p_n \in \Sigma^N\|$$

(This way of presenting the object ω first appeared in [40]; it is also used in [81]). To see that $\omega \cong \omega_\perp$, let $j : |\omega_\perp| \to |\omega|$ be the function defined by $j(\perp) = 0$,

$j(n) = n+1$. Then from the description of lifting given by Proposition 4.2.4 we see that j is tracked by $\lambda^* qm.\, \mathit{if}(\mathit{iszero}\, m)(\mathit{fst}\, q)(\mathit{snd}\, q\, i\, (\mathit{pred}\, m))$, and its inverse is tracked by $\lambda^* p.\, \langle p\,\overline{0},\ \lambda^* zm.p\,(\mathit{succ}\, m)\rangle$.

The following proposition gives the crucial property of ω:

Proposition 5.1.2 *The morphism $j : \omega_\perp \to \omega$ is the* initial algebra *for the lift functor in* $\mathbf{Ass}(A)$*: for any assembly X and morphism $\alpha : X_\perp \to X$ there is a unique morphism $f : \omega \to X$ such that the square*

$$\begin{tikzcd} \omega_\perp \arrow[r, "f_\perp"'] \arrow[d, "j"'] & X_\perp \arrow[d, "\alpha"] \\ \omega \arrow[r, "f"] & X \end{tikzcd}$$

commutes. Indeed, there is a morphism $X^{X_\perp} \to X^\omega$ that computes f given α.

Proof First note that if f makes the diagram commute then we have

$$f(0) \;=\; (\alpha f_\perp j^{-1})(0) \;=\; \alpha(\perp), \qquad f(n+1) \;=\; (\alpha f_\perp j^{-1})(n+1) \;=\; (\alpha \eta_X f)(n),$$

and so if f exists it is unique. Now define $f : |\omega| \to |X|$ recursively by $f(0) = \alpha(\perp)$, $f(n+1) = (\alpha\eta_X f)(n)$. To show that f is a morphism making the diagram commute, we just need to construct a realizer for f.

Suppose t tracks α and v tracks j^{-1}. Let

$$R \;=\; \lambda^* rp.\, t\,\langle p\,\overline{0},\ \lambda^* z.r(\mathit{snd}\,(vp)\, i)\rangle, \qquad r \;=\; ZR.$$

We claim that r tracks f. First observe that for all $p \in A$ we have

$$rp \;\simeq\; Rrp \;\simeq\; t\,\langle p\,\overline{0},\ \lambda^* z.r(\mathit{snd}\,(vp)\, i)\rangle.$$

We now show by induction on n that if $p \in \|n \in \omega\|$ then $rp \in \|f(n) \in X\|$. For the base case, if $p \in \|0 \in \omega\|$ then $p\,\overline{0} \in U$; hence $\langle p\,\overline{0},\ \lambda^* z.r(\mathit{snd}\,(vp)\, i)\rangle \in \|\perp\, \in X_\perp\|$, whence $rp \in \|f(0) \in X\|$. For the induction step, if $p \in \|n+1 \in \omega\|$ then $p\overline{0} \in T$ and $\mathit{snd}\,(vp)\, i \in \|n \in \omega\|$, so $r(\mathit{snd}\,(vp)\, i) \in \|f(n) \in X\|$ by the induction hypothesis. Hence $\langle p\,\overline{0},\ \lambda^* z.r(\mathit{snd}\,(vp)\, i)\rangle \in \|f(n) \in X_\perp\|$, so $rp \in \|f(n+1) \in X\|$. Thus r tracks f as required.

Finally, let $h : |X^{X_\perp}| \to |X^\omega|$ be the set-theoretic function that given a morphism $\alpha : X_\perp \to X$ returns the corresponding morphism f. Then it is immediate from the above proof that h is tracked by

$$\lambda^* t.\ Z(\lambda^* rp.\ t\,\langle p\,\overline{0},\ \lambda^* z.r(snd\,(vp)\,i)\rangle). \quad \square$$

Remark 5.1.3 (Warning for specialists.) In [40, Section 5] another way constructing a generic chain is suggested: let u be the unique morphism $0 \to 1$, and define ω' to be the internal colimit of the internal N-indexed diagram:

$$0 \xrightarrow{u} 1 \xrightarrow{u_\perp} \Sigma \xrightarrow{u_{\perp\perp}} \Sigma_2 \xrightarrow{u_{\perp\perp\perp}} \quad \cdots$$

However, Alex Simpson has pointed out that this object is *not* the same as our object ω. The object ω' can in fact be described concretely as follows:

$$|\omega'| \ = \ \mathrm{N}, \qquad \|n \in \omega'\| \ = \ \|n \in \omega\| \ \wedge \ \{\overline{m} \mid m \geq n\}.$$

It is easy to show that (in the case $A = K_1$, for example) the objects ω and ω' are not isomorphic: there is an obvious morphism $\omega' \to \omega$ but it has no inverse.

We can informally regard morphisms $\omega \to X$ as "ascending chains in X" (this will be made more precise later on). The above proposition thus shows how the iteration of a morphism $X_\perp \to X$ gives rise to an ascending chain in X. Just as in classical domain theory, we will be interested in finding limits for such chains. But in order to discuss these, we need another object $\overline{\omega}$ which we can regard as a "generic ascending chain with limit point".

The idea is to add to the object ω a new point called ∞. The obvious candidate for ∞ is to take the "pointwise limit" of the morphisms $p_0, p_1, \ldots$, namely the morphism $p_\infty : N \to \Sigma$ defined by $p_\infty(m) = \top$. We are thus led to the following definition:

Definition 5.1.4 *Let $\overline{\omega} \in \mathbf{Mod}(A)$ be the object defined by*

$$|\overline{\omega}| \ = \ \mathrm{N} \sqcup \{\infty\}, \qquad \|n \in \overline{\omega}\| \ = \ \|p_n \in \Sigma^N\|, \qquad \|\infty \in \overline{\omega}\| \ = \ \|p_\infty \in \Sigma^N\|.$$

Once again, it is easy to check that $\overline{\omega} \cong \overline{\omega}_\perp$—in fact we can use exactly the same realizers that we used above to show that $\omega \cong \omega_\perp$. We write h for the isomorphism $\overline{\omega} \to \overline{\omega}_\perp$. The object $\overline{\omega}$ enjoys a universal property dual to that of ω:

Proposition 5.1.5 *The morphism h is the* terminal coalgebra *for the lift functor in* $\mathbf{Ass}(A)$*: for any assembly Y and morphism $\beta : Y \to Y_\perp$ there is a unique morphism $g : Y \to \overline{\omega}$ such that the square*

$$\begin{tikzcd} Y_\perp \arrow[r, "g_\perp"'] & \overline{\omega}_\perp \\ Y \arrow[u, "\beta"] \arrow[r, "g"] & \overline{\omega} \arrow[u, "h"'] \end{tikzcd}$$

commutes. Indeed, there is a morphism $(Y_\perp)^Y \to \overline{\omega}^Y$ that computes g given β.

Proof The proof is very similar to that of Proposition 5.1.2. First note that g, if it exists, is uniquely determined by the requirements

$$\begin{array}{rcll} g(y) & = & n & \text{if } \beta^m y \neq \perp \text{ for } 1 \leq m \leq n \text{ and } \beta^{n+1} y = \perp, \\ g(y) & = & \infty & \text{if } \beta^m y \neq \perp \text{ for all } m. \end{array}$$

So let $g : |Y| \to |\overline{\omega}|$ be the function defined thus; we want a realizer for g. Suppose t tracks β. First define

$$u \;=\; \lambda^* a.\, rec(\lambda^* z.ta)(\lambda^* mbz.\, t(snd\,(bi)\,i)).$$

Note that if $a \in \|y \in Y\|$ and $\beta^m y \neq \perp$ for $m \leq n$ then we have $u\,a\,\overline{n}\,i \in \|\beta^{n+1} y \in Y_\perp\|$. Next take $v = \lambda^* anz.\, fst\,(u\,a\,\overline{n}\,i)$; then if y, a, n are as above then $v\,a\,\overline{n}\,i \in \|c(\beta^{n+1} y) \in \Sigma\|$, where $c : Y_\perp \to \Sigma$ classifies η_Y. Now suppose w is a realizer for the dominance (T, U) as in Proposition 4.2.7, and let

$$r \;=\; \lambda^* a.\, rec(v\,a\,\overline{0}\,i)\,(\lambda^* mx.\, w\langle x, va(succ\, m)\rangle).$$

We claim r realizes g. Suppose $a \in \|y \in Y\|$. First, if $\beta y = \perp$ then $r\,a\,\overline{0} \in U$, and so by induction $r\,a\,\overline{m} \in U$ for all m; hence $ra \in \|p_0\|$. Next, if $\beta^m y \neq \perp$ for all $m \leq n$ where $n \geq 1$, then $r\,a\,\overline{0} \in T$, and so by induction $r\,a\,\overline{m} \in T$ for $m < n$. In particular, if $\beta^m y \neq \perp$ for all m then $ra \in \|p_\infty\|$. Lastly, if also $\beta^{n+1} y = \perp$ for

$n \geq 1$ then $r\,a\,\overline{(n-1)} \in T$ and $v\,a\,\overline{n} \in (I \Rightarrow U)$, hence $r\,a\,\overline{n} \in U$ and so $r\,a\,\overline{m} \in U$ for $m \geq n$. Thus $ra \in \|p_n\|$.

Finally, the above construction of a realizer for g is uniform in t, and so there is indeed a morphism $(Y_\perp)^Y \to \overline{\omega}^Y$ that computes g given β. $\square$

We write θ to denote the obvious inclusion $\omega \rightarrowtail \overline{\omega}$ tracked by i. This morphism will play a very important role in the next paragraph.

5.1.2 Well-complete objects

We now turn to the problem of finding a subcategory $\mathcal{D}$ of $\mathbf{Ass}(A)$ that will serve as a good "category of predomains". We would like this category to contain all the objects that correspond to types of PCF, such as $N_\perp$. We would also like it to have good closure properties, e.g. to be closed under lifting, products, sums, exponentials and so on. Finally, as in classical domain theory, we would like all our "predomains" to possess limits of ascending chains. We start by considering this last requirement.

Recall that we are regarding morphisms $\omega \to X$ as ascending chains in X, and morphisms $\overline{\omega} \to X$ as ascending chains with limit point. This suggests that we define an object X to be "chain-complete" if every morphism $\omega \to X$ extends to a unique morphism $\overline{\omega} \to X$. In fact, it turns out to be more fruitful to consider a slightly stronger condition—experts will recognize it as saying that X is chain-complete in an "internal" sense:

Definition 5.1.6 (Complete object) *An object $X \in \mathbf{Ass}(A)$ is* complete *if the map $X^\theta : X^{\overline{\omega}} \to X^\omega$ is an isomorphism. Given a morphism $c : \omega \to X$, we write $\overline{c}$ for the unique morphism $\overline{\omega} \to X$ extending c.*

Note at once that *any* morphism $f : X \to Y$ between complete objects preserves limits of ω-chains: given $c : \omega \to X$ the morphisms $\overline{fc}$, $f\overline{c}$ both extend fc, so $f(\overline{c}\infty) = \overline{fc}(\infty)$. This fact is expressed by the slogan "all maps are continuous".

It might seem that the (full sub)category of complete objects is itself a good choice for a category of predomains: for instance, it is easy to see that it is closed under finite products and finite sums and is an exponential ideal. Unfortunately, however, it seems hard to find simple and natural axioms from which we can prove that the complete objects are closed under lifting. We overcome this difficulty by choosing a slightly different definition for our category of predomains:

Definition 5.1.7 (Well-complete object) *An object X is* well-complete *if $X_\perp$ is complete. We write $\mathcal{D}$ for the full subcategory of* $\mathbf{Ass}(A)$ *consisting of well-complete objects.*

(We will see in Proposition 5.2.1 below that in all cases of interest every well-complete object is complete.)

Proposition 5.1.8 *(i) The complete objects are closed under* retracts*: given $X \xrightarrow{f} Y \xrightarrow{g} X$ such that $gf = \mathrm{id}_X$, if Y is complete then so is X.*

(ii) $\mathcal{D}$ is closed under retracts.

(iii) The complete objects are closed under binary products.

(iv) $\mathcal{D}$ is closed under binary products.

Proof (i) Suppose X, Y, f, g are as above, and Y is complete. Then the composite

$$X^\omega \xrightarrow{f^\omega} Y^\omega \xrightarrow{(Y^\theta)^{-1}} Y^{\overline{\omega}} \xrightarrow{g^{\overline{\omega}}} X^{\overline{\omega}}$$

clearly gives us an inverse to X^θ, so X is complete.

(ii) Suppose Y is well-complete and X is a retract of Y. Then $Y_\perp$ is complete and $X_\perp$ is a retract of $Y_\perp$, so $X_\perp$ is complete by (i). Thus X is well-complete.

(iii) Suppose X, Y are complete. Then the morphism $(X^\theta)^{-1} \times (Y^\theta)^{-1}$ is clearly an inverse to $(X \times Y)^\theta$, so $X \times Y$ is complete.

(iv) Suppose X, Y are well-complete. By (ii) and (iii), it suffices to show that $(X \times Y)_\perp$ is a retract of $X_\perp \times Y_\perp$. Let $p_X : (X \times Y)_\perp \to X_\perp$ classify the partial morphism $(\eta_{X\times Y}, \pi_X) : (X \times Y)_\perp \rightharpoonup X$, and let p_Y classify $(\eta_{X\times Y}, \pi_Y)$. Also let $q : X_\perp \times Y_\perp \to (X \times Y)_\perp$ classify the partial morphism $(\eta_X \times \eta_Y, \mathrm{id}_{X\times Y})$. By considering the action of these morphisms on the underlying sets, it is easy to see that $q\langle p_X, p_Y\rangle = \mathrm{id}$. □

5.2 The completeness axiom

In order to obtain further properties of our category $\mathcal{D}$, we will adopt a *Completeness Axiom*—this will simply state that the object 2 is well-complete. In Section 5.3 we will give several examples of models satisfying this axiom. However, it is also useful to see how far we can travel with the weaker axiom "1 is well-complete", and so we introduce this axiom first, adopting the stronger one only when it becomes necessary. (The phrase "Completeness Axiom" used without qualification will always refer to the stronger form). It turns out that under the Completeness Axiom the category $\mathcal{D}$ has very good closure properties indeed, and compares favourably with the categories of predomains proposed in [40,97,26,81].

5.2.1 The weak completeness axiom

We first postulate the following axiom. We can regard it as a condition on a PCA-with-dominance.

Axiom 1: 1 is well-complete.

Note that this is the same as the Axiom 9 of [40], which says that Σ is complete (or equivalently that θ is Σ-*equable*). A straightforward consequence of Axiom 1 is the following reassuring fact:

Proposition 5.2.1 *Any well-complete object is complete.*

Proof Suppose X is well-complete. Given $c : \omega \to X$, let $d : \overline{\omega} \to X_\perp$ be the unique morphism extending $\eta_X c$. We claim $d(\infty) \neq \perp$. For if $g : X_\perp \to \Sigma$ is the classifier of η_X, then $gd : \overline{\omega} \to \Sigma$ extends the constant morphism $\top : \omega \to \Sigma$, whence $gd = \top : \overline{\omega} \to \Sigma$ using Axiom 1. Thus we have $d(y) \in |X|$ for all $y \in |\overline{\omega}|$; hence if t tracks d then $\lambda^* z.snd(tz)i$ tracks a morphism $\overline{c} : \overline{\omega} \to X$ extending c. Moreover $\overline{c}$ is the unique such morphism since η_X is mono. Finally, the function $|X^\omega| \to |X^{\overline{\omega}}|$ that computes $\overline{c}$ from c is tracked by $\lambda^* r.\lambda^* z.snd(q(\lambda^* x.\langle v, k(rx)\rangle)z)i$, where q tracks $((X_\perp)^\theta)^{-1}$ and $v \in T$. □

Another easy consequence of Axiom 1 is the following closure property:

Proposition 5.2.2 *$\mathcal{D}$ is closed under lifting.*

PROOF Suppose X is well-complete; we wish to show that $X_{\perp\perp}$ is complete. By Axiom 1 and Proposition 5.1.8(iii) we know that $X_\perp \times \Sigma$ is complete, and so by 5.1.8(i) it suffices to show that $X_{\perp\perp}$ is a retract of $X_\perp \times \Sigma$. Let $g : X_{\perp\perp} \to \Sigma$ classify $\eta_{X_\perp}$ and let $f : X_\perp \times \Sigma \to X_{\perp\perp}$ classify the partial map $(\langle \mathrm{id}_{X_\perp}, \top\rangle, \mathrm{id}_{X_\perp})$. Then it is easy to see by considering elements of the underlying sets that $f\langle \mu_X, g\rangle = \mathrm{id}_{X_{\perp\perp}}$. □

It follows immediately from the above two propositions that an object of the form $X_\perp$ is complete iff it is well-complete.

Using Axiom 1 we may also obtain the following interesting fact. It can be seen as an abstract version of the "undecidability of the halting problem".

Proposition 5.2.3 *There is no morphism $H : \Sigma \to \Sigma$ such that $H(\top) = \perp$ and $H(\perp) = \top$.*

PROOF Suppose H is such a morphism, and consider the lift-functor algebra $H\mu_1 : \Sigma_\perp \to \Sigma$. Let c be the unique morphism such that the diagram

$$\begin{array}{ccc} \omega_\perp & \xrightarrow{c_\perp} & \Sigma_\perp \\ {\scriptstyle j}\downarrow & & \downarrow{\scriptstyle H\mu_1} \\ \omega & \xrightarrow{c} & \Sigma \end{array}$$

commutes, as in Proposition 5.1.2. By induction on n it is easy to see that $c(n) = \top$ iff n is even. Now let $d = Hc$ and $d' = cj\eta_\omega$. Then clearly $d(n) = \top$ iff n is odd, and similarly for d'; hence $d = d'$. Since all morphisms preserve limits of chains we have $\overline{d} = H\overline{c}$; but also $\overline{d'} = \overline{c}h^{-1}\eta_{\overline{\omega}}$ since clearly $h^{-1}\eta_{\overline{\omega}}\theta = \theta j \eta_\omega$. But $h^{-1}\eta_{\overline{\omega}}(\infty) = \infty$, so $\overline{c}(\infty) = \overline{d'}(\infty) = \overline{d}(\infty) = H\overline{c}(\infty)$, contradicting the definition of H. □

Corollary 5.2.4 *Given $f : \overline{\omega} \to \Sigma$, if $f(\infty) = \perp$ then $f(n) = \perp$ for all n.*

PROOF Suppose $f(\infty) = \perp$ and $f(n) = \top$. Let $t : 1 \to \overline{\omega}$ be the morphism that picks out ∞, and let $g = (h^{-1}\eta_{\overline{\omega}})^n \circ (h^{-1}t_\perp) : \Sigma \to \overline{\omega}$. It is easy to check that

$g(\top) = \infty$ and $g(\bot) = n$. But now $(fg)(\top) = \bot$ and $(fg)(\bot) = \top$, contradicting the above proposition. $\Box$

Remark 5.2.5 Proposition 5.2.3 is closely related to Axiom 6 of [40] and the Phoa principle of [97]. However, both of these express a slightly stronger property which does not hold in "sequential" models such as $\mathbf{RT}(\Lambda^0/\mathcal{B})$.

Next we show that the well-complete objects form an *exponential ideal*: if X is well-complete and Y is any assembly then X^Y is well-complete. It is easy to see that the complete objects form an exponential ideal, since if X is complete we have canonical isomorphisms $(X^Y)^{\overline{\omega}} \cong (X^{\overline{\omega}})^Y \cong (X^\omega)^Y \cong (X^Y)^\omega$. The result for the well-complete objects is harder; the proof we give involves some rather messy calculations with realizers.

Proposition 5.2.6 *$\mathcal{D}$ is an exponential ideal.*

Proof Suppose X is well-complete, Y any assembly. If $Y = 0$ then $X^Y \cong 1$ so X^Y is well-complete by Axiom 1. Otherwise, let $f : (X^Y)_\bot \to (X_\bot)^Y$ be the exponential transpose of the classifying morphism for the partial map

$$\begin{tikzcd} X^Y \times Y \arrow[r, "\mathrm{ev}"'] \arrow[d, "\eta_{X^Y} \times \mathrm{id}_Y"'] & X \\ (X^Y)_\bot \times Y. & \end{tikzcd}$$

Given $c : \omega \to (X^Y)_\bot$ there is a unique $d : \overline{\omega} \to (X_\bot)^Y$ extending fc, since $(X_\bot)^Y$ is complete. We first check that $d(\infty)$ is in the set-theoretic image of f. Take any $y \in |Y|$; then either $d(\infty)(y) = \bot$ or $d(\infty)(y) \in |X|$. If $d(\infty)(y) = \bot$ then $d(n)(y) = \bot$ for all n by Corollary 5.2.4; thus $c(n) = \bot$ for all n and so $d(\infty) = f(\bot)$. Otherwise if $d(\infty)(y) \in X$ then by Axiom 1 we obtain $d(n)(y) \in X$ for some n; hence $c(n) \in |X^Y|$ and $d(n)(y') \in X$ for all y'. Using Corollary 5.2.4 and the evident map $X_\bot \to \Sigma$ we thus have $d(\infty)(y') \in X$ for all y'. Moreover, if r tracks $d(\infty) : Y \to X_\bot$ then clearly $\lambda^* a.\, snd(ra)i$ tracks a morphism $g : Y \to X$ such that $d(\infty) = f(g)$.

Let $Z \rightarrowtail (X_\perp)^Y$ be the subobject determined by

$$|Z| = \mathrm{Im}\, f, \qquad \|z \in Z\| = \|z \in (X_\perp)^Y\|.$$

Then the above shows that $d : \overline{\omega} \to (X_\perp)^Y$ factors through Z. But $f : (X^Y)_\perp \to Z$ is an isomorphism, since it is clearly a set-theoretic bijection and its inverse is tracked by $\lambda^* a.\, \langle fst(ab), \lambda^* w.snd(ab)i\rangle$ for any $b \in \|y\|$. Thus we obtain a morphism $\overline{c} = f^{-1}d : \overline{\omega} \to (X^Y)_\perp$ extending c. Moreover $\overline{c}$ is unique since d is unique. Finally, if t tracks f, u tracks f^{-1} and v tracks $(((X_\perp)^Y)^\theta)^{-1}$, then the function $(X^Y)_\perp^\omega \to (X^Y)_\perp^{\overline{\omega}}$ that computes $\overline{c}$ from c is tracked by $\lambda^* r.\lambda^* q.u(v(\lambda^* p.t(rp))q)$. Thus $(X^Y)_\perp^\theta$ is an isomorphism, and X^Y is well-complete. $\Box$

Remark 5.2.7 It seems that a more natural proof of this proposition can be obtained if one adopts a different (but equivalent) definition of well-completeness using the internal logic. The details may appear elsewhere.

5.2.2 The strong completeness axiom

As we have seen, under Axiom 1 the category $\mathcal{D}$ already has quite good closure properties. However, to make further headway we need to adopt the stronger form of our axiom.

Axiom 2: 2 is well-complete.

Note that Axiom 2 implies Axiom 1 by Proposition 5.1.8(ii) since 1 is a retract of 2. An easy consequence of this axiom is that the well-complete objects are closed under finite sums.

Proposition 5.2.8 *(i) $\mathcal{D}$ is closed under finite sums.*

(ii) $\mathcal{D}$ is closed under "smash sums": if $X_\perp, Y_\perp \in \mathcal{D}$ then $(X+Y)_\perp \in \mathcal{D}$.

Proof (i) It follows trivially from Axiom 1 that 0 is well-complete. Suppose X, Y are well-complete; we wish to show that $X+Y$ is well-complete. If $X \cong 0$ or $Y \cong 0$ then the result is trivial, so suppose we have morphisms $x : 1 \to X$, $y : 1 \to Y$. Since $\mathcal{D}$ is closed under finite products and retracts, it suffices to show that $X+Y$

is a retract of $2 \times X \times Y$. But it is easy to check that $2 \times X \times Y \cong (X \times Y) + (X \times Y)$, and the morphisms

$$X + Y \underset{\pi_X + \pi_Y}{\overset{\langle \mathrm{id}, y\rangle + \langle x, \mathrm{id}\rangle}{\rightleftarrows}} (X \times Y) + (X \times Y)$$

clearly compose to give id_{X+Y}.

(ii) Suppose $X_\perp, Y_\perp$ are well-complete; then X, Y are well-complete by Proposition 5.2.1. Thus $X + Y$ is well-complete by (i), and so $(X + Y)_\perp$ is well-complete by Proposition 5.2.2. $\Box$

The next result is rather surprising: if 2 is well-complete then so is the natural number object N. The proof we give is essentially due to Alex Simpson.

Lemma 5.2.9 *There exists a morphism* $min : \overline{\omega} \times \omega \to \omega$ *satisfying*

$$min\,(m, n) \;=\; \begin{cases} m & \text{if } m \leq n \\ n & \text{if } m > n; \end{cases} \qquad min\,(\infty, n) \;=\; n.$$

Proof The morphism min can in fact be defined abstractly using just the initial-algebra property of ω, but it is simpler to take advantage of the representation of $\omega, \overline{\omega}$ as subobjects of Σ^N. Let $\wedge : \Sigma \times \Sigma \to \Sigma$ classify the mono $\langle \top, \top \rangle : 1 \to \Sigma \times \Sigma$, and let m be the composite morphism $\Sigma^N \times \Sigma^N \;\cong\; (\Sigma \times \Sigma)^N \;\xrightarrow{\wedge^N}\; \Sigma^N$. Clearly m restricts to a morphism min with the required properties. $\Box$

Lemma 5.2.10 *There exists a morphism* $sup : (N_\perp)^\omega \to N_\perp$ *such that, for any chain* $c : \omega \to N_\perp$*,* $sup\ c$ *is the least upper bound of* c *with respect to the usual partial order on* $|N_\perp|$.

Proof It is convenient to work with the following presentation of the object 2:

$$|2| \;=\; \{tt, \mathit{ff}\}, \qquad \|tt\| \;=\; \{true\}, \qquad \|\mathit{ff}\| \;=\; \{false\}.$$

The strategy for computing $sup\ c$ is as follows: for each natural number n in turn we compose c with the evident morphism $=_n : N_\perp \to 2_\perp$, and so using Axiom 2 we test whether $sup\ c = n$. If we find an n such that this is true, we return n as the result. This process can only fail to terminate if $sup\ c = \perp$.

Formally, let $=: N \times N \to 2$ be the equality testing morphism (clearly this exists by Theorem 1.1.16); let $\epsilon : N \to (2_\perp)^{N_\perp}$ be the exponential transpose of the classifying morphism of the partial map $(\mathrm{id} \times \eta_N, =)$, and suppose q tracks ϵ. Suppose also t tracks $((2_\perp)^\theta)^{-1}$ and $v \in \|\infty \in \overline{\omega}\|$. Since every chain c is monotone with respect to the usual ordering it has a least upper bound, and so there is a well-defined function $sup : |(N_\perp)^\omega| \to |N_\perp|$. To construct a realizer for sup, first define

$$F \;=\; Z(\lambda^* frn.\ \mathit{if}\,(\mathit{snd}(t(\lambda^* p.qn(rp))v)i)\ n\ (fr(\mathit{succ}\ n))).$$

If r tracks c, $sup\ c = n$ and $m \leq n$, it is easy to show by induction on $n - m$ that $F\,r\,\overline{m} = \overline{n}$; so in particular $F\,r\,\overline{0} = \overline{n}$. Moreover, if $d = \mathit{fst}(t(\lambda^* p.q\overline{0}(rp))v)$ it is clear that $d \in T$ if $sup\ c \in N$ and $d \in U$ if $sup\ c = \perp$. Thus sup is tracked by the element $\lambda^* r.\ \langle\, \mathit{fst}(t(\lambda^* p.q\overline{0}(rp))v),\ \lambda^* w.F\,r\,\overline{0}\,\rangle$. $\Box$

Proposition 5.2.11 *N is well-complete.*

PROOF We can obtain a morphism $g : (N_\perp)^\omega \to (N_\perp)^{\overline{\omega}}$ as the composite

$$(N_\perp)^\omega \xrightarrow{(N_\perp)^{min}} (N_\perp)^{\overline{\omega}\times\omega} \;\cong\; (N_\perp^\omega)^{\overline{\omega}} \xrightarrow{sup^{\overline{\omega}}} (N_\perp)^{\overline{\omega}}$$

By construction it is clear that $g(N_\perp)^\theta = \mathrm{id}$. Moreover, if $d : \overline{\omega} \to N_\perp$ extends $c : \omega \to N_\perp$ then d is monotone and if $c(n) = \perp$ for all n then $d(\infty) = \perp$ by Axiom 1; hence $d = h(c)$. Thus $(N_\perp)^\theta g = \mathrm{id}$, so $(N_\perp)^\theta$ is an isomorphism. $\Box$

The following theorem summarizes the closure properties of $\mathcal{D}$:

Theorem 5.2.12 *If Axiom 2 holds, then $\mathcal{D}$ is closed under finite products, finite sums, exponentials, retracts and lifting, and contains the object N.* $\Box$

Remark 5.2.13 We believe that $\mathcal{D}$ is also an internally small-complete and reflective subcategory of $\mathbf{RT}(A)$ in the sense of [39], and that $\mathcal{D}$ has initial solutions of recursive domain equations. To obtain these results, it seems easiest to use a characterization of the well-complete objects involving the internal logic. However, the details are beyond the scope of this thesis.

It is natural to ask how our category $\mathcal{D}$ compares with other "categories of predomains" in $\mathbf{RT}(A)$. The main advantage of our category is that it has very good closure properties which hold "uniformly" for a wide range of realizability models, as we shall see in Section 5.3. Other categories of predomains, such as the replete objects and the Σ-CPOs, share many but not all of these closure properties—for instance, in certain "sequential" models the objects 2 and N are neither replete nor Σ-CPOs! (The reason for this may appear elsewhere.) One can argue that this does not matter from the point of view of denotational semantics, since for abstract reasons these categories will contain their own objects 2 and N, different from those in the topos. Nevertheless, to us it seems more natural to try to use the existing natural number object in the topos as the "predomain of natural numbers"; and in any case desirable it seems that the inclusion from the category of predomains to the topos should preserve as much structure as possible.

We should point out, however, that so far we have only considered well-complete objects in the context of *realizability* models. It would be interesting to try to abstract away from this setting to give a more axiomatic treatment. At present we have not considered whether the well-complete objects work well in models other than realizability models—this obviously has a bearing on whether well-completeness is really a good notion from the point of view of synthetic domain theory.

Of the three categories we have mentioned, the category of replete objects seems the most natural and mathematically compelling. Hyland and Taylor have found several different definitions yielding this category, which provides some evidence for its "canonical" status. The price paid is the lack of a good concrete description of the replete objects. For both the Σ-CPOs and the well-complete objects, our concrete grasp is rather better. It is possible to prove facts about the well-complete objects by direct calculation, as some of the above proofs show.

5.3 Models satisfying completeness

In this section we show that several of our examples of dominances do indeed satisfy the Completeness Axiom. Note at once that not all of them do: for example, for both the decidable subobject classifier 2 and the uniform dominance $\nabla 2$ there is a morphism $\Sigma \to \Sigma$ interchanging $\top$ and $\bot$, so these dominances cannot satisfy even the weak Completeness Axiom (see Proposition 5.2.3). However, it seems that many natural examples do satisfy the axiom, and so give rise to good categories of predomains.

We begin with the leading example, the r.e. dominance Σ in $\mathcal{E}\!f\!f$ (cf. Example 4.3.5(ii)). We require the following fact from classical recursion theory:

Theorem 5.3.1 (Myhill-Shepherdson) *Every morphism $F : (N_\bot)^N \to (N_\bot)^N$ in $\mathbf{Mod}(K_1)$ is monotone and continuous (i.e. preserves existing lubs of chains) with respect to the pointwise order on $(N_\bot)^N$.*

Proof A simple reformulation of the classical Myhill-Shepherdson Theorem as stated e.g. in [17, Chapter 10]. □

Proposition 5.3.2 *The r.e. dominance on K_1 satisfies the Completeness Axiom.*

Proof We wish to show that $(2_\bot)^\theta : 2_\bot^{\overline{\omega}} \to 2_\bot^\omega$ is an isomorphism. We first construct a morphism $sup : 2_\bot^\omega \to 2_\bot$. Since every morphism $c : \omega \to 2_\bot$ is monotone with respect to the Σ-order, we may define a function sup by taking $sup\ c$ to be the least upper bound of the image of c. To obtain a realizer for sup, let r be such that if t is any realizer for any $c : \omega \to 2_\bot$ then

$$rt \ = \ n \ \Rightarrow \ c(n) \neq \bot, \qquad rt\!\uparrow \ \Rightarrow \ c(n) = \bot \text{ for all } n$$

(here we use the ability to perform "parallel" computations in K_1). Suppose also v tracks the morphism $N \to \omega$ given by $n \mapsto n$. It is now easy to check that sup is tracked by $\lambda^* t.\, \langle \lambda^* z.rt,\ \lambda^* z.snd(t(v(rt)))\, i\rangle$.

We can now obtain a morphism $2_\perp^\omega \to 2_\perp^{\overline{\omega}}$ as in the proof of Proposition 5.2.11. Let g be the composite

$$(2_\perp)^\omega \xrightarrow{(2_\perp)^{min}} (2_\perp)^{\overline{\omega}\times\omega} \quad\cong\quad (2_\perp^\omega)^{\overline{\omega}} \xrightarrow{sup^{\overline{\omega}}} (2_\perp)^{\overline{\omega}};$$

from the construction of g it is easy to see that $g(2_\perp)^\theta = \mathrm{id}$. To see that $(2_\perp)^\theta g = \mathrm{id}$, it suffices to show that for any $d : \overline{\omega} \to 2_\perp$ we have $d(\infty) = sup(d\theta)$. But this follows easily from the Myhill-Shepherdson Theorem, since $\overline{\omega}, 2_\perp$ can be exhibited as retracts of $(N_\perp)^N$. $\Box$

In all the other examples we shall consider, the PCA comes with a natural order structure and the proof of Axiom 2 boils down to the fact that application in the PCA is continuous. We next consider the model introduced as Example 4.3.5(iii), a mild generalization of the model considered in [72]. Let S be a semi-sensible λ-theory and let $D \subseteq \Lambda^0/S$ be the divergence consisting of unsolvable terms; let $\omega, \overline{\omega}$ be the objects defined as in Paragraph 5.1.1, (taking $T = \{i\}$, $U = D$). Our first task is to find some more convenient objects that are isomorphic to $\omega, \overline{\omega}$:

Lemma 5.3.3 *Let $\omega', \overline{\omega}'$ be defined by*

$$\begin{aligned} \|n \in \omega'\| \;=\; \|n \in \overline{\omega}'\| \;&=\; \{\lambda f.f^n B \mid B \text{ unsolvable}\}, \\ \|\infty \in \overline{\omega}'\| \;&=\; \{P \mid \mathrm{BT}(P) = \mathrm{BT}(Y)\}. \end{aligned}$$

and let $\theta' : \omega' \rightarrowtail \overline{\omega}'$ be tracked by i. Then there are isomorphisms $\omega \cong \omega'$, $\overline{\omega} \cong \overline{\omega}'$ identifying θ with θ'.

Proof Let $M = \lambda pm.\, \mathit{if}(\mathit{iszero}\ m)\, i\, (p(\mathit{pred}\ m))$. Then M tracks the morphism $e : \overline{\omega} \to \overline{\omega}$ given by $e(n) = n+1$, $e(\infty) = \infty$, and it is easy to check that $\lambda x.xM$ tracks the evident map $\overline{\omega}' \to \overline{\omega}$. Conversely, let $N = Y(\lambda q.\lambda npf.\,(pnf)\,(q(\mathit{succ}\ n)pf))$. Then for any n we have

$$N\,\overline{0} \;=\; \lambda^* pf.\,(p\,\overline{0}\,f)\cdots(p\,\overline{n}\,f)(N\,\overline{(n+1)}\,p\,f).$$

Thus if $P \in \|p_n\|$ then $N\,\overline{0}\,P = \lambda^* f.f^n B$ where B is unsolvable. Moreover if $P \in \|p_\infty\|$ then it is easy to see that $\mathrm{BT}(N\,\overline{0}\,P) = \mathrm{BT}(Y)$. Thus $N\,\overline{0}$ tracks the evident map $\overline{\omega} \to \overline{\omega}'$. Hence we have an isomorphism $\overline{\omega} \cong \overline{\omega}'$ which clearly restricts to an isomorphism $\omega \cong \omega'$. $\Box$

In the case $S = \mathcal{B}$ the objects ω', ϖ' are precisely the ones considered in [72]. Using standard facts about Böhm trees we now easily obtain:

Proposition 5.3.4 *The above divergence D on Λ^0/S gives a dominance satisfying the Completeness Axiom.*

Proof To see that $(2_\perp)^{\theta'}$ is an isomorphism, we simply use the fact that $\mathrm{BT}(Y) = \bigcup \mathrm{BT}(\lambda f.f^n\Omega)$, and that application preserves unions of chains with respect to Böhm tree inclusion (see [8, Section 14.3]). Note that the object $2_\perp$ can be given by:

$$\|tt\| \;=\; \{\lambda xy.x\}, \qquad \|ff\| \;=\; \{\lambda xy.y\}, \qquad \|\bot\| \;=\; \{\Omega\}.$$

Thus if Q tracks a morphism $c : \omega' \to 2_\perp$ and $P \in \|\infty \in \varpi'\|$ then $\mathrm{BT}(QP) = \mathrm{BT}(QY) = \bigcup \mathrm{BT}(Q(\lambda f.f^n\Omega))$ where $\lambda f.f^n\Omega \in \|n \in \omega'\|$; hence $QP \in \|z \in 2_\perp\|$ where z is the least upper bound of $c(0), c(1), \ldots$ in the usual ordering. Hence Q tracks c iff Q tracks its unique continuous extension $\overline{c} : \varpi' \to 2_\perp$. So $(2_\perp)^{\theta'}$ has an inverse tracked by i. $\square$

A very similar argument can be used to show a corresponding result for the lazy λ-calculus. Let S be any lazy λ-theory that is *semi-sensible* in the sense that if $S \vdash M = N$ and M has a whnf then so does N; and let $D \subset \Lambda^0/S$ be the divergence consisting of terms with no weak head normal form (cf. Example 4.3.9(i)). To show that the corresponding dominance satisfies the Completeness Axiom, we make use of *Longo trees* (see [53]). Recall that if M has no whnf then $\mathrm{LT}(M) = \bot$; if M has principal whnf $\lambda x.M'$ then $\mathrm{LT}(M) = \lambda x.\,\mathrm{LT}(M')$; and if M has principal whnf $yM_1 \ldots M_n$ then $\mathrm{LT}(M)$ is

$$\begin{array}{ccc} & y & \\ \swarrow & & \searrow \\ \mathrm{LT}(M_1) & \cdots & \mathrm{LT}(M_n). \end{array}$$

Proposition 5.3.5 *The divergence D on Λ^0/S gives a dominance satisfying the Completeness Axiom.*

Proof Let ω, ϖ be defined as in Paragraph 5.1.1, and define ω', ϖ' by

$$\|n \in \omega'\| \;=\; \|n \in \varpi'\| \;=\; \{M \mid \mathrm{LT}(M) = \mathrm{LT}(\lambda f.f^n\Omega)\},$$
$$\|\infty \in \varpi'\| \;=\; \{M \mid \mathrm{LT}(M) = \mathrm{LT}(Y)\}.$$

Then using exactly the same realizers $\lambda x.xM$ and $N\overline{0}$ as in Lemma 5.3.3, we have isomorphisms $\omega \cong \omega'$ and $\varpi \cong \varpi'$ identifying θ with the inclusion $\theta' : \omega' \rightarrowtail \varpi'$. Next note that $\mathrm{LT}(Y) = \bigcup \mathrm{LT}(\lambda f.f^n\Omega)$ and that application preserves unions of chains with respect to Longo tree inclusion (see [53]). Hence M realizes a morphism $\omega' \to 2_\perp$ iff M realizes its unique extension $\varpi' \to 2_\perp$. Thus $(2_\perp)^{\theta'}$ is an isomorphism. □

Remark 5.3.6 It seems that a similar result could be obtained for the call-by-value λ-calculus, possibly by using an appropriate notion of tree based on call-by-value reduction to weak head normal form. However, this would involve proving that application in $\mathcal{V}^0$ was continuous with respect to the tree topology, which might require some effort.

Finally we consider a couple of continuous models, i.e. PCAs with a non-trivial CPO-structure for which application is continuous. Given that the Completeness Axiom expresses a kind of "continuity" principle, we would expect it to be easy to show that it holds for such models. In the case of Phoa's dominance on the Scott graph model (see Example 4.3.5(iv)) this is indeed so:

Proposition 5.3.7 *Let $A = \mathcal{P}\omega$ or $\mathcal{P}\omega_{re}$. Then the divergence $\{\emptyset\}$ on A gives a dominance satisfying the Completeness Axiom.*

Proof The corresponding dominance may be taken to be that given by $T = \{\{0\}\}$, $U = \{\emptyset\}$, and the object 2 to be that given by $\|tt\| = \{\{0\}\}$, $\|f\!f\| = \{\{1\}\}$, $\|\perp\| = \{\emptyset\}$. For convenience we take the numerals to be given by $\overline{n} = \{n\}$. Let ω, ϖ be defined as in Paragraph 5.1.1. Given any $X \in \|\infty \in \varpi\|$, define X_n to be the set $X - \{([m]; 0) \mid m \geq n\}$; then $X_n \in \|n \in \varpi\|$ and $X = \bigcup X_n$. Thus if Z tracks a morphism $\omega \to 2_\perp$ then $Z \cdot X = \bigcup Z \cdot X_n$ so Z tracks its unique extension $\varpi \to 2_\perp$. Hence $(2_\perp)^{\theta'}$ is an isomorphism, its inverse being tracked by i. □

Somewhat surprisingly, the corresponding result for the model considered by Amadio (see Example 4.3.5(iv)) is much less trivial. The difficulty is that if x is a realizer for ∞ there is no reason to suppose that x is the lub of a chain $x_0, x_1, \ldots$, where $x_n \in \|n\|$. The following proof shows how this problem may be overcome in this particular case.

Proposition 5.3.8 *Let A be a DCPO such that there are continuous maps $\jmath : A \to (A \rightharpoonup A)$ and $\imath : (A \rightharpoonup A) \to A$ such that $\jmath\imath = \mathrm{id}$. Then the dominance arising from the empty divergence on A satisfies the Completeness Axiom.*

PROOF For each n, let $B_n \subseteq A$ be the set $\{a \mid a \leq \overline{n}\}$, and let C_n be the topological closure of the set $\bigcup_{m \geq n} B_m$ with respect to the Scott topology on A. Let $C = \bigcap C_n$; then C is also a closed set. Note that for any closed set E the *characteristic function* $e : A \rightharpoonup A$ defined by

$$e(a)\uparrow \quad \text{if } a \in C, \qquad e(a) \;=\; a \quad \text{if } a \notin C$$

is a partial continuous function; let c_n, c be the characteristic functions of C_n, C respectively. Note that $\overline{n} \notin C_{n+1}$ since there exists $a \in A$ with $a \cdot \overline{n}\downarrow$ and $a \cdot \overline{m}\uparrow$ for $m > n$; hence $\overline{n} \notin C$, and so $c(\overline{n}) = \overline{n}$.

Now let $f : A \to A$ be the continuous function defined by $f(a) = \imath(\jmath(a) \circ c)$, and let $\omega', \overline{\omega}'$ be defined by

$$\begin{aligned} \|n \in \omega'\| \;=\; \|n \in \overline{\omega}'\| \;&=\; \{f(a) \mid a \in \|n \in \omega\|\}, \\ \|\infty \in \overline{\omega}'\| \;&=\; \{f(a) \mid a \in \|\infty \in \overline{\omega}\|\}. \end{aligned}$$

By the above remarks we see that $\|z \in \overline{\omega}'\| \subseteq \|z \in \overline{\omega}\|$ for each $z \in |\overline{\omega}|$; thus there is an isomorphism $\overline{\omega}' \cong \overline{\omega}$ tracked by i whose inverse is tracked by $\imath(f)$. Let $\theta' : \omega' \rightarrowtail \overline{\omega}'$ be the obvious inclusion; we wish to show that $(2_\perp)^{\theta'}$ is an isomorphism. Clearly the object $2_\perp$ may be presented by

$$\|tt\| \;=\; \{true\}, \qquad \|ff\| \;=\; \{false\}, \qquad \|\perp\| \;=\; \{\imath(\perp)\}$$

where $\perp : A \rightharpoonup A$ is the everywhere-undefined function. So as in the previous proof it is sufficient to show that every $x \in \|\infty \in \overline{\omega}'\|$ is the lub of a chain $x_0, x_1, \ldots$ where $x_n \in \|n \in \omega'\|$. Given x, take $x_n = \imath(\jmath(x) \circ c_n)$. It is easy to check that $x_n \in \|n \in \omega\|$ and moreover that that $f(x_n) = x_n$; hence $x_n \in \|n \in \omega'\|$. Furthermore we have $c = \bigsqcup c_n$, so $x = f(x) = \bigsqcup x_n$. $\square$

Remark 5.3.9 In the above example it is easy to check that ω is isomorphic to the "obvious" ascending chain ω'' defined by $\|n \in \omega''\| = \{\lambda^* f.f^n(\imath(\bot))\}$. It is a consequence of the above argument that this can be extended to an isomorphism $\overline{\omega} \cong \overline{\omega}''$, where

$$\|\infty \in \overline{\omega}''\| \;=\; \{\bigsqcup_n \lambda^* f.f^n(\imath(\bot))\,\}.$$

We have thus shown that several different dominances provide instances of our general theory. It seems clear that there are many more examples one could consider. There are also several technical questions we have not addressed. For instance, is there a good model in which the complete and well-complete objects do *not* coincide? Is there a model in which Axiom 1 holds but not Axiom 2? Are there useful dominances satisfying Axiom 2 that do not arise from divergences? Do any of the more pathological divergences given in Section 4.3 satisfy Axiom 2? Nevertheless, we hope that we have given enough examples to show that our attempt to give a unified treatment of synthetic domain theory in a range of models has been successful.

5.4 The fixed point property

We now use the results of Section 5.2 to develop the machinery that we will need in the next chapter in order to interpret PCF. We will introduce an abstract categorical notion of "PCF-model" (a category with all the structure needed to model PCF), and show that any category $\mathbf{Mod}(A)$ equipped with a dominance satisfying the Completeness Axiom gives an example of a PCF-model. First, however, we pause to introduce the Σ-*preorder* and the general notion of Σ-*limit*.

5.4.1 The Σ-preorder

We begin by defining the Σ-*preorder* on an object X. The ideas here are fairly standard, though our presentation of them will be much more "external" than that in e.g. [71].

Let $(\mathcal{C}, \Sigma)$ be any category equipped with a class of admissible monos, X an object of $\mathcal{C}$. The basic idea is to regard morphisms $X \to \Sigma$ as "observations" that can be performed on elements of X, and hence to define an "observational preorder" on $\mathrm{Hom}(1, X)$. We do this as follows:

Definition 5.4.1 (Σ-preorder) *The* (external) Σ-preorder on X *is the relation* $\preceq_\Sigma$ *on* $\mathrm{Hom}(1, X)$ *defined by*

$$x \preceq_\Sigma x' \quad \textit{iff} \quad \forall \phi : X \to \Sigma.\ \phi \circ x = \top \Rightarrow \phi \circ x' = \top.$$

In the case $\mathcal{C} = \mathbf{Ass}(A)$, we may regard $\preceq_\Sigma$ as a relation on $|X|$. It is clear that $\preceq_\Sigma$ is indeed a preorder. Moreover, it is easy to see that any morphism $f : X \to Y$ respects the Σ-preorder: if $x \preceq_\Sigma x'$ then $fx \preceq_\Sigma fx'$.

In fact, every morphism f is not only "monotone" but also "continuous" in an appropriate sense. In order to define continuity, we introduce the following notion:

Definition 5.4.2 (Σ-limit) *Suppose* $x : 1 \to X$, $S \subseteq \mathrm{Hom}(1, X)$. *We say* x *is a* Σ-limit *of* S *if*

$$\forall \phi : X \to \Sigma.\ \phi \circ x = \top \ \Leftrightarrow\ \exists y \in S.\ \phi \circ y = \top.$$

Clearly all morphisms f are continuous in the following sense: if x is a Σ-limit of S then fx is a Σ-limit of $\{fy \mid y \in S\}$. Note that Σ-limits (if they exist) need not be unique, as $\preceq_\Sigma$ need not be a partial order.

Suppose x is a Σ-limit of S; then clearly $y \preceq_\Sigma x$ for all $y \in S$. Moreover x is a *least* upper bound of S with respect to $\preceq_\Sigma$: if $y \preceq_\Sigma x'$ for all $y \in S$ then it is easy to see that $x \preceq_\Sigma x'$. However, not every least upper bound is a Σ-limit. For instance, in the model $\mathbf{Ass}(K_1)$ take $X = \Sigma \times \Sigma$, $x = \langle \top, \top \rangle$, $S = \{\langle \top, \bot \rangle, \langle \bot, \top \rangle\}$; then x is a least upper bound but not a Σ-limit of S.

We now consider some examples of the relation $\preceq_\Sigma$ and the notion of Σ-limit in our models of interest:

Proposition 5.4.3 *Suppose* $\mathcal{C} = \mathbf{Ass}(A)$, *and* Σ *is any non-trivial dominance satisfying Axiom 1. Then* *(i) The Σ-preorder on Σ is that generated by $\bot \preceq_\Sigma \top$.*

(ii) The Σ-preorder on ω is generated by $n \preceq_\Sigma n+1$ for all n.

(iii) The Σ-preorder on $\overline{\omega}$ is generated by $n \preceq_\Sigma n+1 \preceq_\Sigma \infty$ for all n.

(iv) In $\overline{\omega}$, the element ∞ is a Σ-limit of the elements of ω.

PROOF (i) The identity morphism on Σ shows that we do not have $\top \preceq_\Sigma \bot$. To see that $\bot \preceq_\Sigma \top$, suppose we have $\phi : \Sigma \to \Sigma$ where $\phi \circ \bot = \top$. Then by Proposition 5.2.3 we cannot have $\phi \circ \top = \bot$, so $\phi \circ \top = \top$ as required.

(ii) Using the isomorphism $\omega \cong \omega_\bot$ we may easily construct $f_0 : \Sigma \to \omega$ such that $f\bot = 0$, $f\top = 1$. By repeated use of $\omega \cong \omega_\bot$ we may hence construct a morphism f_n for each n such that $f_n\bot = n$ and $f_n\top = n+1$. Since all morphisms are monotone we thus have $n \preceq_\Sigma n+1$ by (i). Moreover, using the morphism μ_ω we may also construct morphisms $g_n : \omega \to \Sigma$ such that $g_n(m) = \bot$ for $m \leq n$ and $g_n(m) = \top$ for $m > n$. Thus if $m \preceq_\Sigma n$ for $m > n$ then since g_n is monotone we have $\top \preceq_\Sigma \bot$, contradicting (i).

(iii) Since all morphisms $\omega \to \Sigma$ extend uniquely to morphisms $\overline{\omega} \to \Sigma$, it is easy to see that we have $m \preceq_\Sigma n$ in $\overline{\omega}$ iff $m \leq n$. We also have $n \preceq_\Sigma \infty$ for each n by Corollary 5.2.4. Finally, since $n+1 \preceq_\Sigma \infty$ and $n+1 \not\preceq_\Sigma n$, we have $\infty \not\preceq_\Sigma n$.

(iv) Immediate from Corollary 5.2.4. □

We have thus made precise our intuition that $\overline{\omega}$ is an infinite ascending sequence of points with a limit point ∞. It is now relatively easy to obtain a "fixed point property" for well-complete objects. Of course, as in classical domain theory we only expect the fixed point property to hold for domains with a bottom element, so we first need to give an appropriate notion of "object with bottom" in our setting.

Definition 5.4.4 (Pointed object) *A* pointed object *is an object X together with a morphism $\alpha : X_\bot \to X$ such that $\alpha\eta_X = \mathrm{id}_X$.*

The informal idea here is that (in our concrete models) $\alpha(\bot)$ is a least element of X with respect to $\preceq_\Sigma$. Later we will see that in the cases of interest there is a *unique* morphism α with the above property.

Theorem 5.4.5 (Fixed point property) *Suppose $\mathcal{C} = \mathbf{Ass}(A)$ and Σ is a non-trivial dominance on $\mathcal{C}$ satisfying Axiom 1. Suppose (X, α) is a well-complete pointed object, and let $\bot_X$ denote the element $\alpha(\bot) \in |X|$ Then there is a morphism $\mathit{fix} : X^X \to X$ such that for all $f : X \to X$ we have*

(i) $\mathit{fix}(f) = f \circ \mathit{fix}(f)$;

(ii) $\mathit{fix}(f)$ is a Σ-limit of $\{\bot_X, f(\bot_X), f^2(\bot_X), \ldots\}$.

Proof The argument we use is by now fairly standard in axiomatic domain theory (see e.g. [28]). We define the morphism *fix* to be the composite

$$X^X \xrightarrow{X^{\alpha_X}} X^{X_\bot} \xrightarrow{g_X} X^\omega \xrightarrow{(X^\theta)^{-1}} X^{\overline{\omega}} \xrightarrow{\mathrm{ev}_\infty} X,$$

where g_X is the morphism given by Proposition 5.1.2. Given f, let $c : \omega \to X$ be the unique morphism making the square

$$\begin{tikzcd} \omega_\bot \arrow[r, "c_\bot"'] \arrow[d, "j"'] & X_\bot \arrow[d, "f\alpha"] \\ \omega \arrow[r, "c"] & X \end{tikzcd}$$

commute as in Proposition 5.1.2, and let $\overline{c}$ be the unique morphism $\overline{\omega} \to X$ extending c. Clearly $\overline{c}(\infty) = \mathit{fix}(f)$ and $\overline{c}(n) = f^n(\bot_X)$ for each n. But ∞ is a limit of $0, 1, \ldots$ in $\overline{\omega}$ and $\overline{c}$ respects limits, so $\mathit{fix}(f)$ is a Σ-limit of $\bot_X, f(\bot_X), \ldots$. This establishes (ii). To see (i), note first that $f\overline{c} = \overline{c}h^{-1}\eta_{\overline{\omega}}$ (where $h : \overline{\omega} \to \overline{\omega}_\bot$ is as in Proposition 5.1.5), since clearly these extend the same morphism $\omega \to X$. But $h^{-1}\eta_{\overline{\omega}}(\infty) = \infty$, and so

$$f \circ \mathit{fix}(f) \;=\; f\overline{c}(\infty) \;=\; \overline{c}h^{-1}\eta_{\overline{\omega}}(\infty) \;=\; \overline{c}(\infty) \;=\; \mathit{fix}(f). \quad \square$$

5.4.2 PCF-models

In the next chapter we will discuss interpretations of several versions of PCF in certain categories. Although our primary interest is in categories of the form $\mathbf{Mod}(A)$ or $\mathbf{Ass}(A)$, our results apply to other models as well and hold for quite general reasons, so it is convenient to formulate and prove them in an abstract categorical setting. For this reason we introduce a notion of *PCF-model*: essentially a category with all the structure we need in order to be able to interpret (various versions of) PCF. We first define a preliminary notion:

Definition 5.4.6 (PCF-premodel) *A* PCF-premodel *is a cartesian-closed category* $\mathcal{C}$ *with a natural number object* N *and initial object* 0*, together with a class of admissible monos such that* 0 *and* 1 *are the only admissible subobjects of* 1*.*

Note that the categories $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$, $\mathbf{RT}(A)$ equipped with any non-trivial dominance all provide examples of PCF-premodels. Other examples are the category **CPO** of (non-pointed) CPOs and continuous maps (where the admissible monos are given by Scott-opens), and the category **Set** (where we take all monos to be admissible). For any PCF-premodel we have a lift monad $(\perp, \eta, \mu)$ defined as in Section 4.1, and for every object X a morphism $\perp_{X_\perp} : 1 \to X_\perp$ obtained as the classifier of the partial map $(0 \rightarrowtail 1, 0 \to X)$.

The condition that 0 and 1 are the *only* admissible subobjects of 1 could probably be relaxed, but it seems to simplify some of the proofs in Chapter 6. It is an easy consequence of this condition that every morphism $1 \to X_\perp$ either factors through η_X or is equal to $\perp_{X_\perp}$.

A PCF-model will essentially be a PCF-premodel such that we have fixed point operators for all the objects used to model PCF types. In order to express this we use the following rather *ad hoc* definition:

Definition 5.4.7 (Useful object) *Suppose* $\mathcal{C}$ *is a PCF-premodel. The class of* useful *objects of* $\mathcal{C}$ *is inductively defined as follows:* Σ *and* $N_\perp$ *are useful; if* X *is useful then so are* $X_\perp$ *and* X^Y *for any* Y*; and that's all.*

Proposition 5.4.8 *In any PCF-premodel, all useful objects* X *are pointed (i.e. there is a morphism* $\alpha_X : X_\perp \to X$ *with* $\alpha_X \eta_X = \mathrm{id}$*).*

Proof We define α_X by induction over the class of useful objects. If X is a useful object of the form $Z_\perp$, we take $\alpha_X = \mu_Z$. Now suppose we already have α_X as required, and consider X^Y. Let $f : (X^Y)_\perp \to (X_\perp)^Y$ be defined as in the proof of Proposition 5.2.6, and take $\alpha_{X^Y} = (\alpha_X)^Y \circ f$. It is easy to check that $\alpha_{X^Y}\eta_{X^Y} = \mathrm{id}$. $\Box$

Thus for every useful X we have a morphism $\bot_X = \alpha_X \circ \bot_{X_\bot} : 1 \to X$; it is easy to see that $\bot_X$ is a least element of $\mathrm{Hom}(1, X)$ with respect to the Σ-preorder. We may now define our abstract notion of model for PCF:

Definition 5.4.9 (PCF-model) *A PCF-premodel $\mathcal{C}$ is a* PCF-model *if for every useful object X there is a morphism $Fix_X : X^X \to X$ such that for all $f : X \to X$ we have*

(i) $\mathit{fix}_X \circ \overline{f} = f \circ \mathit{fix}_X \circ \overline{f}$;

(ii) $\mathit{fix}_X \circ \overline{f}$ is a Σ-limit of $\{\bot_X, f \circ \bot_X, f^2 \circ \bot_X, \ldots\}$,

where $\overline{f} : 1 \to X^X$ is the exponential transpose of f.

This definition is not especially natural or elegant, but it serves our purpose. Although our notion of PCF-model is relatively weak (we do not require CPO-enrichment, for instance), it is likely that some of the conditions could be weakened still further, and it would be interesting to investigate this. It is easy to see that our categories of interest are indeed PCF-models:

Proposition 5.4.10 *Suppose (A, T, U) is a PCA-with-dominance satisfying the (strong) Completeness Axiom. Then the categories* **Mod**(A)*,* **Ass**(A)*,* **RT**(A) *(equipped with the dominance given by (T, U)) are all PCF-models.*

Proof Note that every useful object in these categories is well-complete by Theorem 5.2.12; moreover, every useful object is pointed by Proposition 5.4.8. Thus for every useful object X there is a morphism fix_X with the required properties by Theorem 5.4.5. $\Box$

Note also that the category **CPO** is a PCF-model, but that **Set** is not.

Besides the morphisms fix_X, we also have morphisms $\mathit{iter}^k_X : X^X \to X$ for each $k \geq 0$ defined as follows:

$$\mathit{iter}^0_X \;=\; \bot_X \circ u, \qquad \mathit{iter}^{k+1}_X \;=\; \mathit{eval} \circ \langle \mathrm{id}, \mathit{iter}^k_X \rangle$$

(where $u : X^X \to 1$ and *eval* is the evaluation morphism). Note that for any $f : X \to X$ we have $iter^k_X \circ \overline{f} = f^k \circ \bot$. Wherever it is safe to do so we will write $iter^k_X(f)$ in place of $iter^k_X \circ \overline{f}$, and $fix_X(f)$ in place of $fix_X \circ \overline{f}$.

Recall from Proposition 5.4.3(i) that in PCF-models $\mathbf{Ass}(A)$ we have $\bot \preceq_\Sigma \top$ in Σ but not $\top \preceq_\Sigma \bot$. It is easy to see that this holds in any PCF-model: the identity morphism $\Sigma \to \Sigma$ refutes $\top \preceq_\Sigma \bot$; and if $\phi(\bot) = \top$ then $fix(\phi) \neq \bot$ and so $fix(\phi) = \top$, whence $\phi(\top) = \top$. This has the following useful consequence:

Proposition 5.4.11 *The Σ-preorder on an object $X_\bot$ is given as follows: for all $x, x' \in$ Hom$(1, X)$ we have $\eta x \preceq_\Sigma \eta x'$ iff $x \preceq_\Sigma x'$ in X; $\bot \preceq_\Sigma \eta x$ always; and $\eta x \preceq_\Sigma \bot$ never.*

Proof If $x \preceq_\Sigma x'$ then $\eta x \preceq_\Sigma \eta x'$ since η_X is monotone; and the converse holds because every morphism $X \to \Sigma$ extends to a morphism $X_\bot \to \Sigma$. For any X the morphism $x_\bot : \Sigma \to X_\bot$ shows that $\bot \preceq_\Sigma \eta x$; and the morphism $X_\bot \to \Sigma$ classifying η_X shows that $\eta x \not\preceq_\Sigma \bot$. $\Box$

We will say that a PCF-model is *standard* if it is well-pointed (i.e. the functor Hom$(1, -)$ is faithful) and the only morphisms $1 \to N$ are the evident "numerals". Note that all PCF-models of the form $\mathbf{Mod}(A)$ or $\mathbf{Ass}(A)$ are standard (as is the PCF-model $\mathbf{CPO}$), though $\mathbf{RT}(A)$ is not. Standard PCF-models are particularly pleasant to work with as they have the following good property:

Proposition 5.4.12 *In any standard PCF-model $\mathcal{C}$, every useful object X is a* Σ-space, *i.e. the Σ-preorder on* Hom$(1, X)$ *is a partial order.*

Proof By induction on useful objects. The object Σ is clearly a Σ-space since $\top \not\preceq_\Sigma \bot$; and if X is a Σ-space then so is $X_\bot$ by Proposition 5.4.11. To show that $N_\bot$ is a Σ-space it thus suffices to see that N is a Σ-space. But for every numeral $n : 1 \to N$ one can construct a morphism $t_n : N \to N$ such that for all $m : 1 \to N$ we have $t_n \circ m = 0$ iff $m = n$ (see e.g. [52, Part III]); hence one can obtain $s_n : N \to \Sigma$ such that $s_n \circ m = \top$ iff $m = n$.

It remains to check that if X is a Σ-space then so is X^Y. So suppose we have $z, z' : 1 \rightrightarrows X^Y$ such that $z \preceq_\Sigma z' \preceq_\Sigma z$. For any $y : 1 \to Y$, define $x = eval \circ \langle z, y \rangle$,

$x' = eval \circ \langle z', y \rangle$; then clearly $x \preceq_\Sigma x' \preceq_\Sigma x$. But X is a Σ-space, so $x = x'$. Thus $z = z'$ by well-pointedness. $\Box$

It follows that for any useful X the set $\mathrm{Hom}(1, X)$ has a unique least element with respect to $\preceq_\Sigma$, and so there is a *unique* morphism α_X such that $\alpha_X \eta_X = \mathrm{id}$. Also the Σ-limit of a set $S \subseteq \mathrm{Hom}(1, X)$ is unique if it exists; hence it is easy to see that the morphisms fix_X are uniquely determined by conditions (i) and (ii) in Definition 5.4.9. These facts will be useful when we study further properties of standard PCF-models in Section 6.3.

Chapter 6

Interpretations of PCF

We now turn our attention at last to programming languages, and in this chapter consider interpretations in realizability models of Plotkin's language PCF (see [79]). In fact we will consider several versions of PCF, falling naturally into a three-by-three grid. Along one axis we have three different "degrees of parallelism" (cf. [86]), represented by the pure sequential PCF, the extension PCF^{+} including a "parallel-or" operator, and the further extension PCF^{++} including both parallel-or and an "exists" functional. (All three of these languages essentially feature in [79].) Along the other axis we have different "evaluation mechanisms", giving rise to *call-by-name, call-by-value* and *lazy* variants of PCF.

The goal of this chapter is twofold. First, we exploit the theory of well-complete objects developed in Chapter 5 to show that we can give an adequate interpretation of sequential PCF in any realizability model satisfying our strong completeness axiom (see Paragraph 5.2.2). But since there is a wide choice of realizability models available, it seems natural to try to find a good "fit" between particular languages and particular models—for example, to try to match up languages with models that are fully abstract or even universal for them. Our second goal in this chapter is to show that for these purposes it is only the degree of parallelism of the language that is significant, the three evaluation mechanisms being in some sense "equivalent". Thus, for example, a category with certain structure yields a fully abstract model of call-by-name PCF^{+} iff it also yields a fully abstract model of call-by-value or lazy PCF^{+}; and similarly for universality.

The chapter is structured as follows. In Section 6.1 we review the syntax and operational semantics of the various versions of PCF—the material here will be familiar to many readers. In Section 6.2 we show how to give interpretations of these languages in our models, and prove an adequacy theorem. The material here is "folklore"—the ideas are essentially just a simple categorical generalization of those in [79]. In Section 6.3 we consider syntactic translations between different versions of PCF, and in 6.4 we use these translations to obtain theorems asserting the "equivalence" of different evaluation mechanisms. These results appear to be new, though they are very much in the same spirit as those in [92,83].

Throughout this chapter we work in the abstract categorical setting of an arbitrary PCF-model. Thus this chapter is in principle entirely independent of the rest of the thesis apart from Paragraph 5.4.2 (though we also assume familiarity with the background material on dominances in Section 4.1).

6.1 Syntax and operational semantics

We begin by summarizing the syntax and operational semantics of the languages in question. The material here will be very familiar to readers of papers such as [79,92,83]—such readers may safely skip this section and refer back to it if necessary. We follow fairly closely the economical presentation given in [83]. In Paragraph 6.1.1 we describe the three forms of sequential PCF; in Paragraph 6.1.2 we describe certain parallel extensions of these languages.

6.1.1 Sequential PCF

PCF is a "prototypical" functional programming language based on the simply-typed λ-calculus with some arithmetic and recursion, deriving from Scott's LCF (see e.g. [34]). The original form of the language (see [79]) incorporated the following "design decisions". Firstly, β-redexes were reduced in a call-by-name fashion: a redex $(\lambda x.M)N$ could be contracted to $M[N/x]$ without any reduction of N.

Secondly, "observations" were admitted only on terms of ground type: if M was a term of higher type there was no way to observe whether the evaluation of M terminated. Other versions of PCF have since been studied that differ from the original version with respect to these decisions, and it is the relationship between these different versions of PCF that we wish to study here.

In the design of PCF there are thus two choices that we can make: whether the reduction is to proceed in a call-by-name or call-by-value fashion, and whether we are allowed to observe termination at higher types. In fact these choices are not independent, since if we adopt call-by-value reduction then it automatically becomes possible to observe termination at higher types via observations on ground types: the evaluation of $(\lambda x.5)M$ halts iff the evaluation of M halts. For call-by-name reduction there is a genuine choice regarding observations. One convenient way to force termination at higher types to be observable is to add a "convergence-testing" operator conv to the language: informally, the effect of $\mathsf{conv}\,MN$ is to evaluate M, discard the result, then evaluate N and return the result. We can then use the expression $\mathsf{conv}\,M\,5$ to "observe" the termination of M.

We thus have three versions of PCF to consider. First, there is the original version introduced in [79]: we call this PCF_N or *call-by-name* PCF. Second, there is the language obtained by adding the conv operator to PCF_N; this was studied in [13] and is called PCF_L or *lazy* PCF. Thirdly, there is the *call-by-value* language PCF_V; this was studied in [93]. We now describe the syntax and operational semantics of these languages.

The three languages have almost exactly the same syntax. Each language has a set of *types* freely constructed from a single ground type ι (representing the natural numbers) using the binary type constructor $\rightarrow$. (We eschew a separate ground type for the booleans, representing "true" by 0 and "false" by any other natural number.) Each language also has rules for forming judgements of the form $M : \sigma$, we say M is a *term* of the language if $M : \sigma$ is derivable for some σ. For each type σ we assume we have a countably infinite supply of *variables* $x_0^\sigma, x_1^\sigma, \ldots$ For PCF_N and PCF_V the term formation rules are as follows:

$$x_i^\sigma : \sigma \quad (i \in \mathrm{N}) \qquad\qquad 0, 1, 2, \ldots : \iota$$

$$\frac{M : \iota}{(\mathsf{succ}\ M) : \iota} \qquad\qquad \frac{M : \iota}{(\mathsf{pred}\ M) : \iota}$$

$$\frac{M : \iota \quad N : \sigma \quad P : \sigma}{(\mathsf{cond}\ MNP) : \sigma} \qquad\qquad \frac{M : \sigma}{(\mu x_i^\sigma.M) : \sigma}$$

$$\frac{M : \tau}{(\lambda x_i^\sigma.M) : \sigma \to \tau} \qquad\qquad \frac{M : \sigma \to \tau \quad N : \sigma}{(MN) : \tau}$$

For PCF_L we add one extra formation rule:

$$\frac{M : \sigma \quad N : \tau}{(\mathsf{conv}\ MN) : \tau}$$

A term is a *value* if it is either a numeral or a λ-abstraction; we use V as a meta-variable ranging over values. We frequently omit unnecessary parentheses from terms, taking juxtaposition to be left-associative; we also sometimes omit the type ascriptions from variables where these are unimportant or can be inferred. We use $M, N, P, \ldots$ as meta-variables ranging over terms, $m, n, \ldots$ as meta-variables ranging over numerals, and $x, y, z, \ldots$ as meta-variables ranging over variables. We define the notions of free and bound variables and open and closed terms in the obvious way (note that both λ and μ act as binders). For technical simplicity we will consider only terms in which the bound variables have been renamed so that there are no "holes in scope". We use the meta-notation $M[N/x]$ for capture-avoiding substitution.

If M is a term of type τ whose free variables are among $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$, we may write $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \vdash M : \tau$ and say M is a term in the *environment* $\langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \rangle$. We regard the order in which variables appear in an environment as irrelevant; we may also restrict our attention to environments in which no variable appears twice. If Γ is an environment and x a variable not in Γ, we write Γ, x for the environment obtained by appending x to Γ.

We give the operational semantics of these languages by defining a single-valued "evaluation" relation $\Downarrow$ from closed terms to values; we read $M \Downarrow V$ as "M halts with value V". For each of the three languages, a relation $\Downarrow$ is defined inductively by a set of derivation rules. We sometimes write $\Downarrow_N, \Downarrow_V, \Downarrow_L$ to distinguish between the evaluation relations.

The following derivation rules are common to all three languages. Here $\smile$ denotes truncated subtraction: $m \smile n = \max(m - n, 0)$.

$$V \Downarrow V \quad (V \text{ a value}) \qquad \frac{M[\mu x.M/x] \Downarrow V}{\mu x.M \Downarrow V}$$

$$\frac{M \Downarrow n}{\mathsf{succ}\ M \Downarrow n+1} \qquad \frac{M \Downarrow n}{\mathsf{pred}\ M \Downarrow n \smile 1}$$

$$\frac{M \Downarrow 0 \quad N \Downarrow V}{\mathsf{cond}\ MNP \Downarrow V} \qquad \frac{M \Downarrow n+1 \quad P \Downarrow V}{\mathsf{cond}\ MNP \Downarrow V}$$

We also have some rules specific to particular languages:

For PCF_N :
$$\frac{M \Downarrow_N \lambda x.M' \quad M'[N/x] \Downarrow_N V}{MN \Downarrow_N V}$$

For PCF_L :
$$\frac{M \Downarrow_L \lambda x.M' \quad M'[N/x] \Downarrow_L V}{MN \Downarrow_L V}$$

$$\frac{M \Downarrow_L V' \quad N \Downarrow_L V}{\mathsf{conv}\ MN \Downarrow_L V}$$

For PCF_V :
$$\frac{M \Downarrow_V \lambda x.M' \quad N \Downarrow_V V' \quad M'[V'/x] \Downarrow_V V}{MN \Downarrow_V V}$$

We write $M \Downarrow$ if $M \Downarrow V$ for some V, and $M \Uparrow$ otherwise. We also write $\Downarrow_N, \Uparrow_V$ etc. when we need to be specific about which language we mean.

6.1.2 Parallel extensions of PCF

The languages defined above are all intuitively "sequential" in the sense that there is a single thread of computation—no two sub-expressions of an expression are ever evaluated in parallel. We now introduce some extensions PCF^+ and PCF^{++} including parallel operators. Each of these comes in call-by-name, lazy and call-by-value flavours.

The PCF^+-languages are obtained from PCF by the addition of a parallel-or operator. The term formation rules for each of the languages PCF^+_N, PCF^+_L, PCF^+_V are those for the corresponding sequential language plus the extra rule

$$\frac{M : \iota \quad N : \iota}{(\mathsf{por}\ MN) : \iota}$$

The derivation rules for the evaluation relation are just those of the corresponding sequential language plus the following:

$$\frac{M \Downarrow 0}{\mathsf{por}\ MN \Downarrow 0} \qquad \frac{N \Downarrow 0}{\mathsf{por}\ MN \Downarrow 0} \qquad \frac{M \Downarrow m+1 \qquad N \Downarrow n+1}{\mathsf{por}\ MN \Downarrow 1}$$

Again we write $\Downarrow_N, \Downarrow_L, \Downarrow_V$ when the languages need to be distinguished. Note that there is no need for a notational distinction between the evaluation relations for PCF_N and PCF^+_N, for instance, since these agree for all terms of PCF_N.

(Note: Other authors consider the language obtained by adding a "parallel conditional" operator to PCF (see e.g. [79]). This language is essentially equivalent to our PCF^+, since por and the parallel conditional are interdefinable. This was shown for call-by-name PCF in [94]; it is easy to check that in fact the same argument works in the lazy and call-by-value cases.)

In [79] it was shown that the standard domain-theoretic model was fully abstract for PCF^+_N. However, to obtain the definability of all r.e. elements it was necessary to add one further parallel operator, the exists functional. Our languages PCF^{++} will be essentially PCF^+ with this functional added. In our presentation, the argument of exists will have a slightly more complicated type than in [79]—this is to allow us to define the three versions of PCF^{++} in a uniform way.

The term formation rules for each of the languages PCF_N^{++}, PCF_L^{++}, PCF_V^{++} are those for the corresponding version of PCF^+ with the extra rule

$$\frac{M : (\iota \to \iota) \to \iota}{(\mathsf{exists}\ M) : \iota}$$

The derivation rules for the $\Downarrow$ relation are those for the corresponding PCF^+-language plus the rules

$$\frac{M(\lambda x.n) \Downarrow 0}{\mathsf{exists}\ M \Downarrow 0} \qquad \frac{M(\lambda x.\mu y.y) \Downarrow m+1}{\mathsf{exists}\ M \Downarrow 1}$$

Here the role of the type $\iota \to \iota$ is simply to encode the lifted natural numbers: we represent the number n by the constant function $\lambda x.n$, and bottom by the everywhere-undefined function (note that $\mu y.y$ diverges under each of the evaluation mechanisms). This encoding trick is needed to give the correct definition in the call-by-value case. In the call-by-name case it is easy to see that our exists operator and the functional $\exists$ appearing in [79] are interencodable.

6.2 Denotational semantics of PCF

We now show how to interpret these languages in our models. More specifically, we show that in any PCF-model we can give adequate interpretations of the three forms of sequential PCF, and that these can be extended to PCF^+ and PCF^{++} provided the model contains morphisms corresponding to por and exists. The material here may be reasonably regarded as "folklore"—most of the ideas are straightforward adaptations of those in [79]—though it does not seem to have appeared in the literature in the form we require. A similar "categorical" adequacy result has been proved by Fiore and Plotkin [23], who consider interpretations of FPC (a more powerful language than PCF) in a general setting given by CPO-enriched categories.

6.2.1 Interpretation of PCF

We first present the interpretations of PCF_N, PCF_L and PCF_V in any PCF-model. If M is a term of type τ in environment Γ, the denotation of M (in environment Γ) will be a morphism from the object denoting the environment Γ to the object denoting the type τ. For each type σ we will distinguish between an object V^σ of *values* and an object C^σ of *computations*, in the spirit of [63]. We may think of values as the entities that may be bound to variables (and in particular which may be passed as parameters to a λ-abstraction), and computations as the entities which may be denoted by arbitrary terms.

The objects C^σ, V^σ are defined differently for each of the three evaluation mechanisms. For typographical convenience we write $X \to Y$ for Y^X.

$$\begin{array}{rclrclrcl}
V_N^\iota &=& N_\perp, & V_L^\iota &=& N_\perp, & V_V^\iota &=& N, \\
V_N^{\sigma\to\tau} &=& V_N^\sigma \to V_N^\tau, & V_L^{\sigma\to\tau} &=& (V_L^\sigma \to V_L^\tau)_\perp, & V_V^{\sigma\to\tau} &=& V_V^\sigma \to V_{V\perp}^\tau, \\
C_N^\sigma &=& V_N^\sigma, & C_L^\sigma &=& V_L^\sigma, & C_V^\sigma &=& V_{V\perp}^\sigma.
\end{array}$$

(The distinction between computations and values is only really important in the case of PCF_V.) Notice that all the objects C^σ are *useful*. In each case we have an obvious inclusion $\kappa^\sigma : V^\sigma \to C^\sigma$ given as follows:

$$\kappa_N^\sigma \;=\; \mathrm{id}_{V_N^\sigma}, \qquad \kappa_L^\sigma \;=\; \mathrm{id}_{V_L^\sigma}, \qquad \kappa_V^\sigma \;=\; \eta_{V_V^\sigma}.$$

The denotation $[\![\,\tau\,]\!]$ of a type τ will be the object C^τ, and the denotation $[\![\,\Gamma\,]\!]$ of an environment $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$ will be the object $V^{\sigma_1} \times \cdots \times V^{\sigma_n}$. (Strictly speaking, this object depends on the particular order of the $x_i^{\sigma_i}$ and also on the choice of bracketing for the n-fold product. However, for simplicity we will ignore this fact and understand the denotation of an environment to be given only "up to canonical isomorphism".) We also write $[\Gamma]$ for the object $C^{\sigma_1} \times \cdots \times C^{\sigma_n}$.

Next we introduce some basic general notions. If $\mathcal{L}$ is one of our nine languages, a *pre-interpretation* for $\mathcal{L}$ is a mapping assigning to each term $M : \tau$ in environment Γ a morphism $[M]^\Gamma : [\Gamma] \to [\![\,\tau\,]\!]$. We say a pre-interpretation is *compositional* if whenever M is a term in environment $\Delta = \langle x_1^{\sigma_1}, \ldots, x_m^{\sigma_m} \rangle$ and $N_1 : \sigma_1, \ldots, N_m : \sigma_m$ are terms in environment Γ we have

$$[M[\vec{N}/\vec{x}]]^{\Gamma} \;=\; [M]^{\Delta} \circ \langle [N_1]^{\Gamma}, \ldots, [N_m]^{\Gamma} \rangle.$$

Given a pre-interpretation for $\mathcal{L}$ the corresponding *interpretation* assigns $M : \tau$ in environment Γ the morphism

$$[\![M]\!]^{\Gamma} \;=\; [M]^{\Gamma} \circ (\kappa^{\sigma_1} \times \cdots \times \kappa^{\sigma_n}) : [\![\Gamma]\!] \longrightarrow [\![\tau]\!].$$

Note that the pre-interpretation of M is slightly more informative than the interpretation since it tells us the semantic value of M when non-values are substituted for the free variables. We need this extra information in order to interpret the recursion operator. The reason for the final step from the pre-interpretation to the interpretation is that there are natural situations in which the interpretation is fully abstract but the pre-interpretation is not (this nicety is explained in [92]).

In order to define the (canonical) pre-interpretation of our languages in a PCF-model, some more notation is helpful. For any X we write u_X for the unique morphism $X \to 1$. Given $f : X \times Y \to Z$ we write $curry_Y(f)$ for its exponential transpose $X \to Z^Y$. For any X, Y we write $eval : Y^X \times X \to Y$ for the evaluation map. We also write $strict : (Y_\perp)^X \to (Y_\perp)^{X_\perp}$ for the exponential transpose of $\mu_X \circ strict'$, where $strict' : (Y_\perp)^X \times X_\perp \to Y_{\perp\perp}$ is the classifier of the partial map $(\mathrm{id} \times \eta_X, eval_{XY_\perp})$. If X is a useful object, we write $\alpha_X : X_\perp \to X$ for the morphism given by Proposition 5.4.8.

We also want some notation for morphisms corresponding to particular PCF constructs. For each $n \in \mathrm{N}$ we write $k_n : 1 \to N_\perp$ for the morphism corresponding to n. We write $succ$, $pred : N_\perp \to N_\perp$ for the lifts of the successor and predecessor morphisms respectively, where $pred\ 0 = 0$ (for a proof that such a predecessor morphism exists see e.g. [52, Part III]). For each X we define the morphism $cond_X : N_\perp \times X \times X \to X$ as follows: First let $C : N \to X^{X \times X}$ be the mediating morphism determined by the diagram

$$1 \xrightarrow{\;\overline{\pi_0}\;} X^{X \times X} \xrightarrow{\;\overline{\pi_1} \circ u\;} X^{X \times X}$$

using the defining property of the natural number object. Let $C' : N \times X \times X \to X$ be the exponential transpose of C, let $cond'_X$ be the classifier of the partial map $(\eta_N \times \mathrm{id} \times \mathrm{id}, C')$, and let $cond_X = \mu_N \circ cond'_X$. Note that

$$cond_X \circ \langle k_0, \mathrm{id}\rangle \;=\; \pi_0, \qquad cond_X \circ \langle k_{n+1}, \mathrm{id}\rangle \;=\; \pi_1, \qquad cond_X \circ \langle \bot, \mathrm{id}\rangle \;=\; \bot \circ u.$$

Finally, for any useful X, Y we define the morphism $conv_{XY} : X_\bot \times Y_\bot \to Y_\bot$ to be $\mu_Y \circ conv'_{XY}$, where $conv'_{XY}$ classifies the partial map $(\eta_X \times \mathrm{id}, \pi_{Y_\bot})$. Note that

$$conv_{XY} \circ (\eta_X \times \mathrm{id}) \;=\; \pi_{Y_\bot}, \qquad conv_{XY} \circ (\bot_X \times \mathrm{id}) \;=\; \bot_{Y_\bot} \circ u_{Y_\bot}.$$

We now define $[M]^\Gamma$ by induction on the structure of M. The following clauses in the definition are common to all three evaluation mechanisms:

$$\begin{aligned}
[x_i^{\sigma_i}]^\Gamma &= \kappa^{\sigma_i} \circ \pi_i \\
[n]^\Gamma &= k_n \circ u_{[\Gamma]} \\
[\mathsf{succ}\ M]^\Gamma &= succ \circ [M]^\Gamma \\
[\mathsf{pred}\ M]^\Gamma &= pred \circ [M]^\Gamma \\
[\mathsf{cond}\ MNP]^\Gamma &= cond_{[\![\sigma]\!]} \circ \langle [M]^\Gamma, [N]^\Gamma, [P]^\Gamma \rangle \\
[\mu x^\sigma . M]^\Gamma &= fix_{[\![\sigma]\!]} \circ curry_{[\![\sigma]\!]}([M]^{\Gamma, x^\sigma})
\end{aligned}$$

We also have the following clauses specific to particular languages.

$$\begin{aligned}
\text{For PCF}_N: \quad [\lambda x^\sigma . M]^\Gamma &= curry_{[\![\sigma]\!]}([M]^{\Gamma, x^\sigma}) \\
[MN]^\Gamma &= eval \circ \langle [M]^\Gamma,\ [N]^\Gamma \rangle
\end{aligned}$$

$$\begin{aligned}
\text{For PCF}_L: \quad [\lambda x^\sigma . M]^\Gamma &= \eta \circ curry_{[\![\sigma]\!]}([M]^{\Gamma, x^\sigma}) \\
[MN]^\Gamma &= eval \circ \langle \alpha_{V^\sigma \to V^\tau} \circ [M]^\Gamma,\ [N]^\Gamma \rangle \\
[\mathsf{conv}\ MN]^\Gamma &= conv_{[\![\sigma]\!][\![\tau]\!]} \circ \langle [M]^\Gamma,\ [N]^\Gamma \rangle
\end{aligned}$$

$$\begin{aligned}
\text{For PCF}_V: \quad [\lambda x^\sigma . M]^\Gamma &= \eta \circ (C^\tau)^{\kappa^\sigma} \circ curry_{[\![\sigma]\!]}([M]^{\Gamma, x^\sigma}) \\
[MN]^\Gamma &= eval \circ \langle strict \circ \alpha_{V^\sigma \to \tau} \circ [M]^\Gamma,\ [N]^\Gamma \rangle
\end{aligned}$$

We write $[-]_N$, $[\![-]\!]_V$ etc. when we need to be specific about the language. If M is a closed term then we may write $[M]$ and $[\![M]\!]$ in place of $[M]^{\langle\rangle}$ and $[\![M]\!]^{\langle\rangle}$ respectively. It is easy to see that these pre-interpretations are *compositional* in the sense defined above.

This completes the definition of the interpretations. Note that a closed term of type σ is interpreted as a morphism $1 \to [\![\sigma]\!]$, or equivalently (in the case of $\mathbf{Mod}(A)$) as an element of the set $|[\![\sigma]\!]|$.

6.2.2 The adequacy theorem

For any of our nine languages, the "correctness" of some interpretation with respect to the operational semantics is expressed by the following property:

Definition 6.2.1 *Suppose $\mathcal{L}$ is one of our languages, $[\![-]\!]$ an interpretation in some PCF-model. We say the interpretation is* adequate *if for all closed terms $M : \iota$ of $\mathcal{L}$ we have $[\![M]\!]_{\mathcal{L}} = k_n$ if $M \Downarrow_{\mathcal{L}} n$, and $[\![M]\!]_{\mathcal{L}} = \bot$ if $M \Uparrow$.*

This paragraph is devoted to the proof that the canonical interpretations of PCF_N, PCF_L and PCF_V defined above are adequate. For the remainder of the paragraph, we use $\mathcal{L}$ to stand for any one of the (sequential) languages PCF_N, PCF_L, PCF_V.

Theorem 6.2.2 (Adequacy for PCF) *For any PCF-model $\mathcal{C}$, the interpretation of $\mathcal{L}$ in $\mathcal{C}$ defined in Paragraph 6.2.1 is adequate.*

The proof is very similar indeed to the proof of the adequacy theorem in [79]. In order to keep our account self-contained, however, we present the argument here but omit some of the details. For the most part our proof works uniformly for the three languages PCF_N, PCF_L, PCF_V.

To prove the theorem, it is convenient first to enrich $\mathcal{L}$ to a language $\mathcal{L}'$ by adding a new binder μ^k for each k—these will serve as "approximants" to μ. More formally, the term formation rules of $\mathcal{L}'$ are those of $\mathcal{L}$ augmented by rules

$$\frac{M : \sigma}{(\mu^k x_i^\sigma . M) : \sigma}$$

for each $k \geq 0$. The derivation rules for the evaluation relation of $\mathcal{L}'$ are those for $\mathcal{L}$ together with rules

$$\frac{M[\mu^k x.M/x] \Downarrow V}{\mu^{k+1}x.M \Downarrow V}$$

(Note that the absence of a rule for $\mu^0 x.M$ means that $\mu^0 x.M \Uparrow$.)

The introduction of the μ^k is purely a technical convenience, and adds nothing to the "expressivity" of $\mathcal{L}$. (In fact the notation $\mu^k x.M$ can be viewed as syntactic sugar for a certain term of $\mathcal{L}$.) If M is a closed term of $\mathcal{L}$ then clearly $M \Downarrow_{\mathcal{L}} V$ iff $M \Downarrow_{\mathcal{L}'} V$. Moreover it is easy to check the following syntactic fact:

Lemma 6.2.3 *Suppose $D[-]$ is a one-place term context. Then $D[\mu x.M] \Downarrow$ iff $D[\mu^k x.M] \Downarrow$ for some k; moreover $D[\mu x.M] \Downarrow n$ iff $D[\mu^k x.M] \Downarrow n$ for some k.* $\Box$

We can extend our interpretation of $\mathcal{L}$ to $\mathcal{L}'$ by adding the following clause to our definition of the pre-interpretation of M:

$$[\mu^k x^\sigma.M]^\Gamma \quad = \quad iter^k_X \circ curry_{[\![\sigma]\!]}([M]^{\Gamma,x^\sigma})$$

where the morphism $iter^k_X : X^X \to X$ is defined as in Paragraph 5.4.2. It is now easy to verify the following:

Proposition 6.2.4 *If $M \Downarrow V$ then $[\![M]\!] = [\![V]\!]$.*

Proof A straightforward induction on the derivation of $M \Downarrow V$, with one induction case for each evaluation rule of $\mathcal{L}'$. $\Box$

This proves one half of the Adequacy Theorem since $[\![n]\!] = k_n$. To prove the other half, we use the idea of *computable terms* as in [79]. We can define these uniformly in $\mathcal{L}$ as follows:

Definition 6.2.5 (Computable terms) *A closed term $M : \iota$ is* computable *iff $M \Uparrow$ implies $[\![M]\!] = \bot$. A closed term $M : \sigma \to \tau$ is computable iff $M \Uparrow$ implies $[\![M]\!] = \bot$, and $N : \sigma$ closed and computable implies MN computable. An open term M with free variables $x_1, \ldots, x_m$ is computable iff $M[\vec{N}/\vec{x}]$ is computable whenever $N_1, \ldots, N_m$ are computable terms of the appropriate types.*

Our goal is to show that all terms of $\mathcal{L}'$ are computable (this implies the second half of the Adequacy Theorem). The following facts will be useful:

Lemma 6.2.6 *Suppose M, M' are closed terms of type σ.*

(i) If $M \Uparrow$ and $[\![M]\!] = \bot$, then M is computable.

(ii) If $[\![M]\!] = [\![M']\!]$ and $M \Downarrow V \Leftrightarrow M' \Downarrow V$, then M is computable iff M' is computable.

(iii) If $M \Downarrow V$ then M is computable iff V is computable.

PROOF (i) and (ii) are easy by induction on the structure of σ. (iii) follows from (ii), since $V \Downarrow V$ and $[\![M]\!] = [\![V]\!]$ by Proposition 6.2.4. $\Box$

Proposition 6.2.7 *All terms M are computable.*

PROOF By induction on the structure of M. Most of the proof is uniform in $\mathcal{L}$. If M is a variable or a numeral then M is trivially computable. It is also an easy exercise to verify that if $M, N, P : \iota$ are computable then so are $\mathsf{succ}\ M$, $\mathsf{pred}\ M$, $\mathsf{cond}\ MNP$.

If $M : \sigma \to \tau$, $N : \sigma$ are computable then so is MN: If M, N are both closed this is immediate. If the free variables of M, N are among $x_1, \ldots, x_m$ then for any computable terms $P_1, \ldots, P_m$ we have $M[\vec{P}/\vec{x}], N[\vec{P}/\vec{x}]$ computable so $(MN)[\vec{P}/\vec{x}]$ is computable; thus MN is computable.

If $M : \tau$ is computable then so is $\lambda x^\sigma.M$: First suppose $\lambda x.M$ is closed. Note that $\lambda x.M \Downarrow$ so the clause regarding divergence is trivial. Now suppose $N : \sigma$ is closed and computable. The proof that $(\lambda x.M)N$ is computable depends on the particular language $\mathcal{L}$.

- For PCF_N and PCF_L we have $(\lambda x.M)N \Downarrow V \Leftrightarrow M[N/x] \Downarrow V$; moreover it is easy to check in each case that $[(\lambda x.M)N] = [M[N/x]]$ so $[\![(\lambda x.M)N]\!] = [\![M[N/x]]\!]$. But $M[N/x]$ is computable, so $(\lambda x.M)N$ is computable by Lemma 6.2.6(ii).

- For PCF_V, if $N \Downarrow V'$ for some V' then $(\lambda x.M)N \Downarrow V \Leftrightarrow M[V'/x] \Downarrow V$ and $[\![(\lambda x.M)N]\!] = [\![M[V'/x]]\!]$. But $M[V'/x]$ is computable, so $(\lambda x.M)N$

is computable. Otherwise $N \Uparrow$ and so $[\![N]\!] = \bot$; hence $(\lambda x.M)N \Uparrow$ and $[\![(\lambda x.M)N]\!] = \bot$, so $(\lambda x.M)N$ is computable by Lemma 6.2.6(i).

Thus for all three languages $\mathcal{L}$ we have that $\lambda x.M$ is computable. The extension to the case when $\lambda x.M$ is open is straightforward.

Next, if M, N are computable then so is $\mathsf{conv}\ MN$ (in the case of PCF_L). For closed terms this is easy by cases according to whether $M \Downarrow$; the extension to open terms is again straightforward.

It remains to show that if M is computable then so are $\mu^k x.M$, $\mu x.M$. As in the above cases, the extension from closed terms to open terms is easy, so let us assume M is computable and $\mu x.M$ is closed. The proof that $\mu^k x.M$ is computable is easy by induction on k. The fact that $\mu x.M$ is computable is then immediate from the following lemma.

Lemma 6.2.8 *Suppose $D[-]$ is a one-place term context such that $D[P]$ is computable whenever P is computable, and suppose M is computable and $\mu x.M$ is closed. Then $D[\mu x.M]$ is computable.*

Proof By induction on the type of $D[M]$. First suppose $D[M] : \iota$. If $D[\mu x.M] \Uparrow$ then for all k we have $D[\mu^k x.M] \Uparrow$ by Lemma 6.2.3. But $\mu^k x.M$ is computable by the above, so $D[\mu^k x.M]$ is computable and thus $[\![D[\mu^k x.M]]\!] = \bot$. But clearly $[\mu x.M]$ is a Σ-limit of the $[\mu^k x.M]$ from the definitions, hence $[\![D[\mu x.M]]\!]$ is a Σ-limit of the $[\![D[\mu^k x.M]]\!]$, and so we have $[\![D[\mu x.M]]\!] = \bot$. Thus $D[\mu x.M]$ is computable.

Now suppose $D[M] : \sigma \to \tau$. If $D[\mu x.M] \Uparrow$ then exactly as above we have $[\![D[\mu x.M]]\!] = \bot$. So suppose $N : \sigma$ is computable, and define $D'[-] = D[-]N$. Clearly if P is computable then so is $D'[P]$, so by the induction hypothesis we have $D'[\mu x.M]$ computable. Thus $D[\mu x.M]$ is computable. $\square$

This completes the proof of Theorem 6.2.2. Note that for all closed $M : \iota$ the morphism $[\![M]\!]$ is either $\bot$ or k_n for some n, although there may be PCF-models in which not every morphism $1 \to N_\bot$ is of one of these forms.

6.2.3 Interpretations of PCF$^+$ and PCF^{++}

We have seen that any PCF-model provides an adequate and compositional interpretation for any version of sequential PCF. We now consider the circumstances under which this interpretation can be smoothly extended to PCF$^+$ and PCF^{++}.

Suppose $\mathcal{L}$ is any one of our nine languages and $\mathcal{L}_0$ is the version of sequential PCF with the same evaluation mechanism as $\mathcal{L}$. We say that a PCF-model $\mathcal{C}$ *supports* $\mathcal{L}$ if the interpretation of $\mathcal{L}_0$ given in Paragraph 6.2.1 can be extended to an adequate and compositional interpretation of $\mathcal{L}$. (Thus any PCF-model supports any version of sequential PCF.) We first characterize the PCF-models that support the various versions of PCF$^+$.

Proposition 6.2.9 *Suppose $\mathcal{C}$ is a PCF-model. Then the following conditions are equivalent:*

(i) There is a morphism $por\colon N_\perp \times N_\perp \to N_\perp$ in $\mathcal{C}$ such that whenever x is k_n or $\perp$ and y is k_{n+1} or $\perp$ we have

$$por \circ \langle x, k_0\rangle \;=\; por \circ \langle k_0, x\rangle \;=\; k_0, \qquad por \circ \langle k_{m+1}, k_{n+1}\rangle \;=\; k_1,$$
$$por \circ \langle y, \perp\rangle \;=\; por \circ \langle \perp, y\rangle \;=\; \perp.$$

(ii) $\mathcal{C}$ supports PCF^+_N*;*

(iii) $\mathcal{C}$ supports PCF^+_L*;*

(iv) $\mathcal{C}$ supports PCF^+_V*.*

Proof (i) $\Rightarrow$ (ii),(iii),(iv): For each of the three PCF$^+$-languages, we can extend the above interpretation of the corresponding sequential language by adding the clause

$$[\mathsf{por}\; MN]^\Gamma \;=\; por \circ \langle [M]^\Gamma, [N]^\Gamma\rangle$$

to the definition of the pre-interpretation. The resulting interpretation of PCF$^+$ is clearly compositional. The proof of adequacy is a simple modification of the proof in the last section: we just add some easy extra induction steps to the proofs of Propositions 6.2.4 and 6.2.7.

(ii),(iii),(iv) $\Rightarrow$ (i): Let $\Gamma = \langle x^\iota, y^\iota\rangle$, and take $por = [\mathsf{por}\; xy]^\Gamma$. It is easy to check that por satisfies the above equations. $\Box$

If the object $2 = 1+1$ exists in $\mathcal{C}$, condition (i) above is obviously equivalent to the more natural requirement that there be a morphism $2_\perp \times 2_\perp \to 2_\perp$ satisfying certain equations. The reader may like to verify directly at this stage that for the PCF-models $\mathbf{Mod}(K_1)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ (with the dominances of 4.3.5) the morphism *por* does exist, whereas in $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ it does not (these facts will become obvious in the next chapter).

The following proposition characterizes the models that support PCF^{++}. For technical reasons it seems simplest to restrict our attention here to *well-pointed* PCF-models, although it seems that a similar result holds in the more general situation.

Proposition 6.2.10 *Suppose $\mathcal{C}$ is a well-pointed PCF-model satisfying the equivalent conditions of Proposition 6.2.9. Then the following are equivalent:*

(i) There is a morphism $\exists : (N_\perp)^{N_\perp} \to N_\perp$ in $\mathcal{C}$ such that for any $f : N_\perp \to N_\perp$ we have

$$\exists \circ \overline{f} \;=\; \begin{cases} k_0 & \textit{if } f \circ k_n = k_0 \textit{ for some } n \\ k_1 & \textit{if } f \circ \perp = k_{m+1} \textit{ for some } m \\ \perp & \textit{otherwise;} \end{cases}$$

(ii) $\mathcal{C}$ supports PCF$^{++}_N$*;*

(iii) $\mathcal{C}$ supports PCF$^{++}_L$*;*

(iv) $\mathcal{C}$ supports PCF$^{++}_V$*.*

Proof (i) $\Rightarrow$ (ii),(iii),(iv): Let $q : C^{(\iota\to\iota)\to\iota} \times C^\iota \to C^\iota$ be $[F(\lambda y^\iota.x)]^\Gamma$, where $\Gamma = \langle F^{(\iota\to\iota)\to\iota}, x^\iota \rangle$; and let $\overline{q} : C^{(\iota\to\iota)\to\iota} \to (N_\perp)^{N_\perp}$ be the exponential transpose of q. We can extend the above interpretation of PCF$^+$ by adding the clause

$$[\mathsf{exists}\ M]^\Gamma \;=\; \exists \circ \overline{q} \circ [M]^\Gamma$$

to the definition of the pre-interpretation. The resulting interpretation is clearly still compositional. To show that it is adequate we need some extra induction steps in the proofs of Propositions 6.2.4 and 6.2.7, but these are straightforward using the above properties of $\exists$.

(ii) $\Rightarrow$ (i): Let $\Delta = \langle h^{\iota\to\iota} \rangle$, and take $\exists = [\mathsf{exists}\ (\lambda g^{\iota\to\iota}.h(g0))]^\Delta$. Then it is easy to check that $\exists \circ \overline{f}$ satisfies condition (i) whenever f is *denotable* (i.e. $f = [P]$ for some closed $P : \iota \to \iota$). Now suppose $f : N_\perp \to N_\perp$ is arbitrary; the idea is to exhibit f as the limit of a chain of denotable morphisms. Let $G : (N_\perp)^{N_\perp} \to (N_\perp)^{N_\perp}$ be the morphism $[\lambda x^\iota.\, \mathsf{cond}\ x\ 0\ (\mathsf{succ}\,(h(\mathsf{pred}\ x)))]^{\Delta}_{N}$; let g be the exponential transpose of $\mathit{fix}(G)$, and for each k let $g_k : N_\perp \to N_\perp$ be the transpose of $\mathit{iter}^k_X(G)$. Then $g = \mathrm{id}$, $g_k(n) = n$ for $n < k$, and $g_k(n) = \perp$ for $n \geq k$. Moreover g is the Σ-limit of the g_k. Now define $f_k = f \circ g_k$ for each k; then the f_k are clearly denotable and hence satisfy condition (i) above. Hence $f = f \circ g$ satisfies condition (i) by continuity.

(iii) $\Rightarrow$ (i), (iv) $\Rightarrow$ (i) are similar, except that in the definition $\exists$ we need some obvious coercion morphisms $(N_\perp)^{N_\perp} \to C^{\iota\to\iota}$. $\square$

Again, if $\mathcal{C}$ has an object 2, we may clearly replace condition (i) by the condition that there be a morphism $(2_\perp)^{N_\perp} \to 2_\perp$ with certain properties. It is easy to verify directly that in both $\mathbf{Mod}(K_1)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ the morphism $\exists$ does indeed exist (again we will prove this in the next chapter).

The following theorem is now immediate from Propositions 6.2.9 and 6.2.10:

Theorem 6.2.11 *Suppose $\mathcal{C}$ is a well-pointed PCF-model, and $\mathcal{L}_1, \mathcal{L}_2$ are two of our nine languages having the same degree of parallelism. Then $\mathcal{C}$ supports $\mathcal{L}_1$ iff $\mathcal{C}$ supports $\mathcal{L}_2$.* $\square$

In view of this theorem, in future we shall simply say that a PCF-model supports PCF [resp. PCF$^+$, PCF^{++}], without reference to a particular evaluation mechanism.

6.3 Relating variants of PCF

It is natural to try to match our languages with PCF-models in which they have a fully abstract or even a universal interpretation (cf. Section 6.4). In the next chapter we will see several examples of such matching pairs. In this section and the next our goal is to show that for this purpose the evaluation mechanism of the language is irrelevant—it is only the degree of parallelism that matters. In particular we can show that our three evaluation mechanisms are in some sense equi-expressive, by exhibiting good syntactic translations between the respective languages.

In Paragraph 6.3.1 we consider the three type hierarchies in a PCF-model corresponding to the three evaluation mechanisms, and show how each of these type hierarchies can be "embedded" in the others. One consequence of the existence of these embeddings is that each of the type hierarchies is in some sense determined by either of the others. In Paragraph 6.3.2 we give syntactic translations between languages with different evaluation mechanisms, and show that these correspond exactly to our semantic embeddings. In Section 6.4 we will use this information to deduce results relating full abstraction and universality results for languages with different evaluation mechanisms.

We should admit that the proofs throughout this section and the next are in general less detailed than those elsewhere in the thesis—a lot of tedious checking is left to the long-suffering reader. Even so, our written proofs include some quite heavy calculation, and it seemed that more complete proofs would be almost unreadable. It would be interesting to find alternative ways of proving the main results that were at once clean and rigorous. (We suggest one possible approach at the end of the chapter.)

Throughout this section we take $\mathcal{C}$ to be a *standard* PCF-model in the sense of Paragraph 5.4.2. It is not clear how necessary this condition is for our results, but it seems to simplify the proofs considerably. We will frequently show that a pair of morphisms $X \rightrightarrows Y$ are equal by checking that they agree on all global elements—this is often easier than a proof by diagram-chasing.

By an *evaluation mechanism* we will officially just mean one of the three formal symbols N, L, V. We will sometimes write "E is an evaluation mechanism", meaning that E is a meta-variable ranging over the set $\{N, L, V\}$. If $D[-, \ldots, -] : \tau$ is an r-place term context of PCF^{++}_E, with holes of type $\sigma_1, \ldots, \sigma_r$ and with no other free variables, we write $[\![D]\!]_E$ for the evident interpretation of D as a morphism $C_E^{\sigma_1} \times \cdots \times C_E^{\sigma_r} \to C_E^{\tau}$.

6.3.1 Type hierarchies in PCF-models

Recall from Paragraph 6.2.1 that for each evaluation mechanism E and each PCF type σ we may define objects V_E^σ, C_E^σ as follows:

$$
\begin{array}{rclrclrcl}
V_N^\iota & = & N_\perp, & V_L^\iota & = & N_\perp, & V_V^\iota & = & N, \\
V_N^{\sigma\to\tau} & = & V_N^\sigma \to V_N^\tau, & V_L^{\sigma\to\tau} & = & (V_L^\sigma \to V_L^\tau)_\perp, & V_V^{\sigma\to\tau} & = & V_V^\sigma \to V_{V\perp}^\tau, \\
C_N^\sigma & = & V_N^\sigma, & C_L^\sigma & = & V_L^\sigma, & C_V^\sigma & = & V_{V\perp}^\sigma.
\end{array}
$$

For each E we refer to the family of objects C_E^σ as the *E-type hierarchy* associated with $\mathcal{C}$. (Thus we speak of the call-by-name, call-by-value and lazy type hierarchies.) Our main task in this paragraph is to show that each of these type hierarchies can be "embedded" in each of the others. Specifically, we show that the following "generic theorem" holds, where E, E' are two different evaluation mechanisms:

Theorem 6.3.1 *For any PCF type σ, there exist a type σ' and a pair of morphisms $C_E^\sigma \leftrightarrows C_{E'}^{\sigma'}$ such that following triangle commutes:*

$$
\begin{tikzcd}
C_E^\sigma \arrow[r] \arrow[dr, "\mathrm{id}"'] & C_{E'}^{\sigma'} \arrow[d] \\
& C_E^\sigma.
\end{tikzcd}
$$

We can abbreviate the statement of this theorem by the phrase "E embeds in E'." To show that the theorem holds for all distinct E, E' it will suffice to show that V embeds in L, N embeds in V and L embeds in N: the other three embeddings can then be obtained by composition.

Remark 6.3.2 The existence of these embeddings has an interesting consequence involving *finite type structures* (cf. [99]). Loosely speaking, for each evaluation mechanism E we can define a (partial) finite type structure F, consisting of the family of sets $F^\sigma = \mathrm{Hom}(1, V_E^\sigma)$ together with suitable (partial) application operations. Each of these type structures can be seen as giving a measure of the "computational power" associated with $\mathcal{C}$. Using the above theorem it can be shown that each of the three finite type structures arising from $\mathcal{C}$ is determined up to isomorphism by either of the other two. Thus the three type structures all contain exactly the same amount of information about the computational power of $\mathcal{C}$. However, we will not pursue this line of thought here, since formalizing the details is rather laborious and we will not need the result in what follows.

All three of the embeddings that we give will play a prominent role in the rest of this section, so we will introduce some specific notation pertaining to each one. We first describe the embedding of V in L; the notation introduced by the following proposition remains in force for the rest of the chapter.

Proposition 6.3.3 (Embedding V in L) *For each type σ, define $\overline{\sigma} = \sigma$. Then for each σ there exist morphisms $\beta^\sigma : C_V^\sigma \to C_L^{\overline{\sigma}}$ and $\gamma^\sigma : C_L^{\overline{\sigma}} \to C_V^\sigma$ such that $\gamma^\sigma \beta^\sigma = \mathrm{id}$. Moreover, there is a term context $P^\sigma[-]$ of* PCF_L *such that $\beta^\sigma\gamma^\sigma = [\![P^\sigma]\!]_L$.*

Proof We define $\beta^\sigma, \gamma^\sigma$ simultaneously by induction on σ: Let $\beta^\iota = \gamma^\iota = \mathrm{id}_{N_\perp}$; define $\beta^{\sigma\to\tau}$ to be the composite

$$(V_V^\sigma \to C_V^\tau)_\perp \xrightarrow{\ strict_\perp\ } (C_V^\sigma \to C_V^\tau)_\perp \xrightarrow{\ ((\beta^\tau)^{\gamma^\sigma})_\perp\ } (C_L^{\overline{\sigma}} \to C_L^{\overline{\tau}})_\perp;$$

and define $\gamma^{\sigma\to\tau}$ to be the composite

$$(C_L^{\overline{\sigma}} \to C_L^{\overline{\tau}})_\perp \xrightarrow{\ ((\gamma^\tau)^{\beta^\sigma})_\perp\ } (C_V^\sigma \to C_V^\tau)_\perp \xrightarrow{\ ((C_V^\tau)^\eta)_\perp\ } (V_V^\sigma \to C_V^\tau)_\perp.$$

We check by induction on σ that $\gamma^\sigma\beta^\sigma = \mathrm{id}$. We trivially have $\gamma^\iota\beta^\iota = \mathrm{id}$; and for the induction step we have

$$\gamma^{\sigma\to\tau}\beta^{\sigma\to\tau} \;=\; ((C_V^\tau)^\eta)_\perp \circ ((\gamma^\tau\beta^\tau)^{\gamma^\sigma\beta^\sigma})_\perp \circ \mathit{strict}_\perp \;=\; ((C_V^\tau)^\eta)_\perp \circ \mathit{strict}_\perp \;=\; \mathrm{id}.$$

We now define term contexts $P^\sigma[-]$ of PCF_L inductively by

$$\begin{array}{rcl} P^\iota[x^{\overline{\iota}}] & \equiv & x, \\ P^{\sigma\to\tau}[x^{\overline{\sigma\to\tau}}] & \equiv & \mathsf{conv}\;\; x(\lambda y^{\overline{\sigma}}.\,\mathsf{conv}\;\; y(P^\tau[x(P^\sigma[y])])). \end{array}$$

We show by induction on σ that P^σ denotes the morphism $\beta^\sigma\gamma^\sigma$. The case ι is trivial; the induction step provides a good example of the "pointwise" style of reasoning about morphisms. Suppose given $x : 1 \to (C_L^{\overline{\sigma}} \to C_L^{\overline{\tau}})_\perp$. If $x = \perp$ then by the definitions of $\beta^{\sigma\to\tau}$ and $\gamma^{\sigma\to\tau}$ we have $\beta^{\sigma\to\tau}\gamma^{\sigma\to\tau}x = \perp$; but also $[\![\, P^{\sigma\to\tau}]\!]_L \circ x = \mathit{conv} \circ \langle x, g\rangle = \perp$ for an appropriate morphism g. Otherwise we have $x = \eta \circ \mathit{curry}(f)$ for some $f : C_L^{\overline{\sigma}} \to C_L^{\overline{\tau}}$. It is now easy to see that

$$\begin{array}{rrcl} & \beta^{\sigma\to\tau}\gamma^{\sigma\to\tau}x & = & \eta \circ \mathit{curry}(\beta^\tau f'\gamma^\sigma), \\ \text{where} & \mathit{curry}(f') & = & \mathit{strict} \circ (C_V^\tau)^\eta \circ \mathit{curry}(\gamma^\tau f \beta^\sigma). \end{array}$$

But since $\mathit{conv} \circ \langle \eta \circ \mathit{curry}(f), g\rangle = g$ for any g, we also have

$$[\![\, P^{\sigma\to\tau}]\!]_L \circ x \;=\; [\![\, \lambda y^{\overline{\sigma}}.\,\mathsf{conv}\; y\, Z[x,y]]\!]_L \;=\; \eta \circ \mathit{curry}([\![\, \mathsf{conv}\; y\, Z[x,y]]\!]_L^{y^{\overline{\sigma}}})$$

where $Z[x,y] \equiv P^\tau[x(P^\sigma[y])]$. (Here we abuse notation slightly and understand the syntactic variable x to be interpreted by the morphism x.) So it is enough to show that $\beta^\tau f'\gamma^\sigma = [\![\, \mathsf{conv}\; y\, Z[x,y]]\!]_L^y$. Suppose given $y : 1 \to C_L^{\overline{\sigma}}$. If $y = \perp$ then $\gamma^\sigma y = \perp$; but $f'\perp = \perp$ and so $\beta^\tau f'\gamma^\sigma y = \perp$. But also $[\![\, \mathsf{conv}\; y\, Z[x,y]]\!]_L^y \circ y = \mathit{conv} \circ \langle y, h\rangle = \perp$ for an appropriate h. If $y \neq \perp$ then $\gamma^\sigma y \neq \perp$, and so we have

$$\beta^\tau f'\gamma^\sigma y \;=\; \beta^\tau\gamma^\tau f\,\beta^\sigma\gamma^\sigma y \;=\; [\![\, Z[x,y]]\!]_L^y \circ y \;=\; [\![\, \mathsf{conv}\; y\, Z[x,y]]\!]_L^y \circ y. \quad \square$$

The syntactic definability of $\beta^\sigma\gamma^\sigma$ is a technical fact that we will need in the next paragraph. The contexts P^σ given above are also used by Riecke in [83], where they play a very similar role. We have given the proof that P^σ denotes $\beta^\sigma\gamma^\sigma$ in some detail as an example of this kind of reasoning. In future we will seldom give such arguments in full, since when written out they often seem harder than they really are. Typically we will claim that some equality can be "checked pointwise", and leave it to the patient reader to fill in the details.

We next describe the embedding of N in V. This follows a very similar pattern, although the fact that the objects V_V^σ, C_V^σ are different adds a slight extra twist in the definition of the embedding.

Proposition 6.3.4 (Embedding N in V) *For each type σ, define $\widehat{\sigma}$ by*

$$\widehat{\iota} \;=\; \iota \to \iota, \qquad \widehat{\sigma \to \tau} \;=\; \widehat{\sigma} \to \widehat{\tau}.$$

Then for each type σ there exist morphisms $\delta^\sigma : C_N^\sigma \to C_V^{\widehat{\sigma}}$ and $\epsilon^\sigma : C_V^{\widehat{\sigma}} \to C_N^\sigma$ such that $\epsilon^\sigma \delta^\sigma = \mathrm{id}$. Moreover, there is a term context $Q^\sigma[-]$ of PCF_V *such that $\delta^\sigma \epsilon^\sigma = [\![\, Q \,]\!]_V^\sigma$.*

Proof In order to define $\delta^\sigma, \epsilon^\sigma$ we first define morphisms $\delta'^\sigma : C_N^\sigma \to V_V^{\widehat{\sigma}}$ and $\epsilon'^\sigma : V_V^{\widehat{\sigma}} \to C_N^\sigma$ as follows. Let $\delta'^\iota : N_\perp \to (N_\perp)^N$ be the "constant" morphism (the transpose of the evident projection), and let $\epsilon'^\iota : (N_\perp)^N \to N_\perp$ be the "evaluation at zero" morphism (the composite $\mathit{eval} \circ \langle \mathrm{id}, 0\rangle$). Define $\delta'^{\sigma\to\tau}$ to be the composite

$$(C_N^\sigma \to C_N^\tau) \xrightarrow{\;(\delta'^\tau)^{\epsilon'^\sigma}\;} (V_V^{\widehat{\sigma}} \to V_V^{\widehat{\tau}}) \xrightarrow{\;\eta^{V_V^{\widehat{\sigma}}}\;} (V_V^{\widehat{\sigma}} \to C_V^{\widehat{\tau}}),$$

and $\epsilon'^{\sigma\to\tau}$ to be the composite

$$(V_V^{\widehat{\sigma}} \to C_V^{\widehat{\tau}}) \xrightarrow{\;(\alpha_{V_V^{\widehat{\tau}}})^{V_V^{\widehat{\sigma}}}\;} (V_V^{\widehat{\sigma}} \to V_V^{\widehat{\tau}}) \xrightarrow{\;(\epsilon'^\tau)^{\delta'^\sigma}\;} (C_N^\sigma \to C_N^\tau).$$

It is easy to check by induction on σ that $\epsilon'^\sigma \delta'^\sigma = \mathrm{id}$. We now define

$$\delta^\sigma \;=\; \eta_{V_V^{\widehat{\sigma}}} \circ \delta'^\sigma, \qquad \epsilon^\sigma \;=\; \epsilon'^\sigma \circ \alpha_{V_V^{\widehat{\sigma}}}.$$

Then $\epsilon^\sigma \delta^\sigma = \mathrm{id}$ since $\alpha_X \circ \eta_X = \mathrm{id}$. The term contexts Q^σ may be defined thus:

$$Q^\iota[x^{\widehat{\iota}}] \;\equiv\; \lambda y^\iota.\, x0, \qquad Q^{\sigma\to\tau}[x^{\widehat{\sigma\to\tau}}] \;\equiv\; \lambda y^{\widehat{\sigma}}.\, Q^\tau[x(Q^\sigma[y])].$$

The proof that $\delta^\sigma \epsilon^\sigma = [\![\, Q \,]\!]_V^\sigma$ is by induction on σ: both the base case and the induction step can be routinely checked pointwise. □

The description of the embedding of L in N is slightly more involved. We make use of the fact that for any object $X = [\![\sigma]\!]_N$ we can embed $X_\perp$ in the object $(N_\perp \to X)$. Informally, an "element" $x : 1 \to X_\perp$ can be represented by a "function" $x' : N_\perp \to X$. The intuitive idea is as follows: the value of x' at 0 just flags whether or not $x = \perp$, while the value of x' at 1 determines x assuming $x \neq \perp$. This idea can be made more precise in two stages as follows:

Lemma 6.3.5 *(i) For each type σ there exist morphisms $s^\sigma : N_\perp \to C_N^\sigma$ and $t^\sigma : C_N^\sigma \to N_\perp$, denoted respectively by* PCF_N *term contexts $S^\sigma[-]$ and $T^\sigma[-]$, such that $t^\sigma s^\sigma = \mathrm{id}_{N_\perp}$.*

(ii) For each type σ there exist morphisms $\phi^\sigma : (C_N^\sigma)_\perp \to C_N^{\iota\to\sigma}$ and $\psi^\sigma : C_N^{\iota\to\sigma} \to (C_N^\sigma)_\perp$ such that $\psi^\sigma\phi^\sigma = \mathrm{id}$.

Proof (i) Suppose $\sigma = \sigma_1 \to \cdots \to \sigma_r \to \iota$. Let $\Omega^\tau \equiv \mu y^\tau.y$, and define term contexts S^σ, T^σ by

$$S^\sigma[z^\iota] \;\equiv\; \lambda x_1^{\sigma_1} \ldots x_r^{\sigma_r}.\, z, \qquad T^\sigma[x^\sigma] \;\equiv\; x\Omega^{\sigma_1} \ldots \Omega^{\sigma_r}.$$

Take $s^\sigma = [\![\, S^\sigma \,]\!]_N$ and $t^\sigma = [\![\, T^\sigma \,]\!]_N$. It is easy to check pointwise that $t^\sigma s^\sigma = \mathrm{id}$.

(ii) Let $X = C_N^\sigma$, and let $v^\sigma : X_\perp \to N_\perp$ be the lift of $(X \to 1 \xrightarrow{0} N)$. Take ϕ^σ to be the exponential transpose of the composite

$$X_\perp \times N_\perp \xrightarrow{\langle \pi_{N_\perp},\, s^\sigma v^\sigma \pi_{X_\perp},\, \alpha_X \pi_{X_\perp} \rangle} N_\perp \times X \times X \xrightarrow{\quad cond_X \quad} X.$$

For ψ^σ, let $m : Y \rightarrowtail (N_\perp \to X)$ be the Σ-mono obtained as the pullback of η_N along $[\![\, \lambda f^{\iota\to\sigma}.\, T^\sigma[f0] \,]\!]_N$, and let ψ^σ classify the partial map $(m, eval \circ (m \times k_1)) : (N_\perp \to X) \rightharpoonup X$. It is easy to check pointwise that $\psi^\sigma\phi^\sigma = \mathrm{id}$. $\Box$

Proposition 6.3.6 (Embedding L in N) *For each type σ, define $\tilde\sigma$ by*

$$\tilde\iota \;=\; \iota, \qquad \widetilde{\sigma\to\tau} \;=\; \iota \to \tilde\sigma \to \tilde\tau.$$

Then for each type σ there exist morphisms $\zeta^\sigma : C_L^\sigma \to C_N^{\tilde\sigma}$ and $\xi^\sigma : C_N^{\tilde\sigma} \to C_L^\sigma$ such that $\xi^\sigma\zeta^\sigma = \mathrm{id}$. Moreover, there is a context $R^\sigma[-]$ of PCF_N *denoting $\zeta^\sigma\xi^\sigma$.*

Proof Let $\zeta^\iota = \xi^\iota = \mathrm{id}_{N_\perp}$; define $\zeta^{\sigma\to\tau}$ to be the composite

$$(C_L^\sigma \to C_L^\tau)_\perp \xrightarrow{\quad ((\zeta^\tau)^{\xi^\sigma})_\perp \quad} (C_N^{\tilde\sigma} \to C_N^{\tilde\tau})_\perp \xrightarrow{\quad \phi^{\tilde\sigma\to\tilde\tau} \quad} (N_\perp \to (C_N^{\tilde\sigma} \to C_N^{\tilde\tau});$$

and define $\xi^{\sigma\to\tau}$ to be the composite

$$(N_\perp \to (C_N^{\widetilde{\sigma}} \to C_N^{\widetilde{\tau}}) \xrightarrow{\psi^{\widetilde{\sigma}\to\widetilde{\tau}}} (C_N^{\widetilde{\sigma}} \to C_N^{\widetilde{\tau}})_\perp \xrightarrow{((\xi^\tau)^{\zeta^\sigma})_\perp} (C_L^\sigma \to C_L^\tau)_\perp.$$

Using the above lemma it is easy to check that $\xi^\sigma \zeta^\sigma = \mathrm{id}$ for all σ. Now define the term contexts $R^\sigma[-]$ of PCF_N as follows:

$$\begin{array}{rcl} R^\iota[x^{\widetilde{\iota}}] & \equiv & x, \\ R^{\sigma\to\tau}[x^{\widetilde{\sigma\to\tau}}] & \equiv & conv\,(T^{\widetilde{\sigma}\to\widetilde{\tau}}[x0])\ \ (\lambda z^\iota.\,\mathsf{cond}\ z\ (S[0])\ (\lambda y^{\widetilde{\sigma}}.\,R^\tau[x1(R^\sigma[y])])), \end{array}$$

where $conv \equiv (\lambda p^\iota q^{\widetilde{\sigma\to\tau}}.\,\mathsf{cond}\ pqq)$. Again we can prove by induction on σ that $\zeta^\sigma\xi^\sigma = [\![R]\!]_N^\sigma$: the base case is trivial, and the induction step is easily checked pointwise, arguing by cases according to whether $[\![T[x0]]\!]_N = \perp$. $\Box$

6.3.2 Translations between variants

We have shown above how each of the three type hierarchies associated with $\mathcal{C}$ can be embedded in the others. In this paragraph we define syntactic translations between languages that mirror these embeddings. Specifically, suppose $\mathcal{L}, \mathcal{L}'$ are two languages with the same degree of parallelism but with different evaluation mechanisms E, E', and for each type σ let $e^\sigma : C_E^\sigma \to C_{E'}^{\sigma'}$ be the embedding defined above. Our task is to establish the following generic theorem:

Theorem 6.3.7 *There is a syntactic translation $M \mapsto M'$ from terms of $\mathcal{L}$ to terms of $\mathcal{L}'$ such that*

(i) if $M : \sigma$ then $M' : \sigma'$;

(ii) if $M : \sigma$ is a closed $\mathcal{L}$-term then $[\![M']\!]_{E'} = e^\sigma \circ [\![M]\!]_E$.

As in the previous paragraph, we will prove our generic theorem in three instances—the remaining instances then follow by composing translations.

We first give a translation from call-by-value to lazy PCF, corresponding to the embeddings β^σ of the previous paragraph. This translation is not novel: it is essentially identical to one given by Riecke [83], who attributes it to Ong [67].

Definition 6.3.8 (Translating V to L) *Recall that $\overline{\sigma} = \sigma$ for all σ. The translation $M \mapsto \overline{M}$ from PCF_V^{++}-terms to PCF_L^{++}-terms is defined inductively as follows:*

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\overline{x^\sigma} & \equiv & P^\sigma[x^{\overline{\sigma}}] & \overline{n} & \equiv & n \\
\overline{\mathsf{succ}\ M} & \equiv & \mathsf{succ}\ \overline{M} & \overline{\mathsf{pred}\ M} & \equiv & \mathsf{pred}\ \overline{M} \\
\overline{\mathsf{cond}\ MNP} & \equiv & \mathsf{cond}\ \overline{M}\ \overline{N}\ \overline{P} & \overline{MN} & \equiv & \overline{M}\ \overline{N} \\
\overline{\lambda x^\sigma . M} & \equiv & \lambda x^{\overline{\sigma}} .\, \mathsf{conv}\ x\, \overline{M} & \overline{\mu x^\sigma . M} & \equiv & \mu x^{\overline{\sigma}} . \overline{M} \\
\overline{\mathsf{por}\ MN} & \equiv & \mathsf{por}\ \overline{M}\ \overline{N} & \overline{\mathsf{exists}\ M} & \equiv & \mathsf{exists}\ \overline{M}
\end{array}$$

Note at once that this restricts to give translations $\mathrm{PCF}_V \to \mathrm{PCF}_L$ and $\mathrm{PCF}_V^+ \to \mathrm{PCF}_L^+$, and that if $M : \sigma$ then $\overline{M} : \overline{\sigma}$. We wish to show that $[\![\, \overline{M} \,]\!]_L = \beta^\sigma \circ [\![\, M \,]\!]_V$ for all closed $M : \sigma$. In order to establish this, we in fact need to prove a more general result for *open* terms. Given an environment $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_m^{\sigma_m} \rangle$, let us write $\overline{\Gamma}$ for the environment $\langle x_1^{\overline{\sigma_1}}, \ldots, x_m^{\overline{\sigma_m}} \rangle$, and $\beta^\Gamma, \gamma^\Gamma$ respectively for the morphisms

$$\beta^{\sigma_1} \times \cdots \times \beta^{\sigma_m} : [\Gamma]_V \to [\overline{\Gamma}]_L, \qquad \gamma^{\sigma_1} \times \cdots \times \gamma^{\sigma_m} : [\overline{\Gamma}]_L \to [\Gamma]_V.$$

The relationship between the translation and the embedding may then be expressed as follows. Here we use $\mathcal{L}$ as a meta-variable ranging over the lexical tokens PCF, PCF^+, PCF^{++}.

Proposition 6.3.9 *Suppose $\mathcal{C}$ supports $\mathcal{L}$, and $M : \sigma$ is a term of $\mathcal{L}_V$ in environment Γ. Then the following diagram commutes:*

$$\begin{tikzcd}
{[\Gamma]_V} \arrow[r, "{[M]_V^\Gamma}"] & {[\![\, \sigma \,]\!]_V} \arrow[d, "\beta^\sigma"] \\
{[\overline{\Gamma}]_L} \arrow[u, "\gamma^\Gamma"] \arrow[r, "{[\overline{M}]_L^{\overline{\Gamma}}}"] & {[\![\, \overline{\sigma} \,]\!]_L}
\end{tikzcd}$$

In particular, if M is closed then $[\![\, \overline{M} \,]\!]_L = \beta^\sigma \circ [\![\, M \,]\!]_V$.

Proof By induction on the structure of M. If x^σ is a variable then by Proposition 6.3.3 we have

$$\beta^\sigma \circ [x]_V^\Gamma \circ \gamma^\Gamma \;=\; \beta^\sigma \circ \gamma^\sigma \circ [x]_L^{\overline{\Gamma}} \;=\; [P^\sigma[x]]_L^{\overline{\Gamma}}.$$

The cases corresponding to n, succ and pred are trivial: for instance, for succ we just need to observe that $succ \circ \beta^\iota = \beta^\iota \circ succ$. For cond it is enough to verify that

$$cond \circ (\beta^\iota \times \beta^\sigma \times \beta^\sigma) \;=\; \beta^\sigma \circ Cond.$$

But this is easily checked pointwise, by cases according to whether the first component is $\bot$, k_0 or k_{n+1}. (Note that every morphism $1 \to N_\bot$ is of one of these forms since $\mathcal{C}$ is standard.)

For the case corresponding to application, it suffices to check that

$$[xy]_L^{\overline{\Delta}} \circ \beta^\Delta \;=\; \beta^\tau \circ [xy]_V^\Delta,$$

where $\Delta = \langle x^{\sigma\to\tau}, y^\sigma\rangle$. But

$$\begin{aligned}
[xy]_L^{\overline{\Delta}} \circ \beta^\Delta &= eval \circ (\alpha \times \mathrm{id}) \circ (\beta^{\sigma\to\tau} \times \beta^\sigma) \\
&= eval \circ ((\alpha \circ ((\beta^\tau)^{\gamma^\sigma})_\bot \circ strict_\bot) \times \beta^\sigma) \\
&= eval \circ ((\beta^\tau)^{\gamma^\sigma} \times \beta^\sigma) \circ ((strict \circ \alpha) \times \mathrm{id}) \\
&= \beta^\tau \circ eval \circ ((strict \circ \alpha) \times \mathrm{id}) \\
&= \beta^\tau \circ [xy]_V^\Delta.
\end{aligned}$$

For the case corresponding to λ-abstraction, suppose M is a term in environment Γ, x^σ for which the proposition holds. Then

$$\begin{aligned}
[\overline{\lambda x^\sigma.M}]_L^{\overline{\Gamma}} &= \eta \circ curry(conv \circ \langle \pi^{\overline{\sigma}}, [\overline{M}]_L^{\overline{\Gamma,x}}\rangle) \\
&= \eta \circ strict \circ (C_L^{\overline{\tau}})^\eta \circ curry(\beta^\tau \circ [M]_V^{\Gamma,x} \circ (\gamma^\Gamma \times \gamma^\sigma)) \\
&= \eta \circ strict \circ (C_L^{\overline{\tau}})^\eta \circ (\beta^\tau)^{\gamma^\sigma} \circ curry([M]_V^{\Gamma,x}) \circ \gamma^\Gamma \\
&= ((\beta^\tau)^{\gamma^\sigma})_\bot \circ strict_\bot \circ \eta \circ (C_V^\tau)^\eta \circ curry([M]_V^{\Gamma,x}) \circ \gamma^\Gamma \\
&= \beta^{\sigma\to\tau} \circ [\lambda x^\sigma.M]_V^\Gamma \circ \gamma^\Gamma.
\end{aligned}$$

For the case corresponding to the fixed point operator, we first show that $\mathit{fix} \circ (\beta^\sigma)^{\gamma^\sigma} = \beta^\sigma \circ \mathit{fix}$ for any σ. This can be checked pointwise as follows: given any $f : C_V^\sigma \to C_V^\sigma$ we have that $\mathit{fix} \circ (\beta^\sigma)^{\gamma^\sigma} \circ \overline{f}$ is a Σ-limit of $\{(\beta^\sigma \circ f \circ \gamma^\sigma)^k \circ \bot \mid k \geq 0\}$, and $\beta^\sigma \circ \mathit{fix} \circ \overline{f}$ is a Σ-limit of $\{\beta^\sigma \circ f^k \circ \bot \mid k \geq 0\}$. But for each k we have $(\beta^\sigma \circ f \circ \gamma^\sigma)^k \circ \bot = \beta^\sigma \circ f^k \circ \gamma^\sigma \circ \bot = \beta^\sigma \circ f^k \circ \bot$ (note that $\gamma^\sigma \circ \bot \preceq_\Sigma \gamma^\sigma \circ \beta^\sigma \circ \bot$ and so $\gamma^\sigma \circ \bot = \bot$). Thus by Proposition 5.4.12 we have $\mathit{fix} \circ (\beta^\sigma)^{\gamma^\sigma} \circ \overline{f} = \beta^\sigma \circ \mathit{fix} \circ \overline{f}$ as required. Now suppose M is a term in environment Γ, x^σ for which the proposition holds. Then

$$\begin{aligned}
[\overline{\mu x^\sigma . M}]_L^{\overline{\Gamma}} &= \mathit{fix} \circ \mathit{curry}([\overline{M}]_L^{\overline{\Gamma,x}}) \\
&= \mathit{fix} \circ (\beta^\sigma)^{\gamma^\sigma} \circ \mathit{curry}([M]_V^{\Gamma,x}) \circ \gamma^\Gamma \\
&= \beta^\sigma \circ \mathit{fix} \circ \mathit{curry}([M]_V^{\Gamma,x}) \circ \gamma^\Gamma \\
&= \beta^\sigma \circ [\mu x^\sigma . M]_V^\Gamma \circ \gamma^\Gamma .
\end{aligned}$$

The case corresponding to por is trivial. For exists, it suffices to show that

$$[\mathsf{exists}\ x]_L^{\overline{x}} \circ \beta^{(\iota \to \iota) \to \iota} \ = \ \beta^\iota \circ [\mathsf{exists}\ x]_V^x .$$

But this can be checked pointwise, by cases according to whether $[\mathsf{exists}\ x]_V$ is k_0, k_{n+1} or $\perp$. This completes the induction. The specialization to closed terms is immediate. $\square$

The results for the other two translations follow a very similar pattern. We next give a translation from call-by-name to call-by-value, corresponding to the embeddings δ^σ.

Definition 6.3.10 (Translating N to V) *Recall that $\widehat{\iota} = \iota \to \iota$, $\widehat{\sigma \to \tau} = \widehat{\sigma} \to \widehat{\tau}$. The translation $M \mapsto \widehat{M}$ from PCF_N^{++} to PCF_V^{++} is defined as follows:*

$$\begin{aligned}
\widehat{x^\sigma} &\equiv Q^\sigma[x^{\widehat{\sigma}}] \\
\widehat{n} &\equiv \lambda z^\iota . n \\
\widehat{\mathsf{succ}\ M} &\equiv \lambda z^\iota . \mathsf{succ}\ (\widehat{M} 0) \\
\widehat{\mathsf{pred}\ M} &\equiv \lambda z^\iota . \mathsf{pred}\ (\widehat{M} 0) \\
\widehat{\mathsf{cond}\ MNP} &\equiv \lambda x_1^{\widehat{\sigma_1}} \ldots x_r^{\widehat{\sigma_r}} . \lambda z^\iota . \mathsf{cond}\ (\widehat{M} 0)\ (\widehat{N} x_1 \ldots x_r y)\ (\widehat{P} x_1 \ldots x_r y) \\
\widehat{MN} &\equiv \widehat{M} \widehat{N} \\
\widehat{\lambda x^\sigma . M} &\equiv \lambda x^{\widehat{\sigma}} . \widehat{M} \\
\widehat{\mu x^\sigma . M} &\equiv \mu x^{\widehat{\sigma}} . \widehat{M} \\
\widehat{\mathsf{por}\ MN} &\equiv \lambda z^\iota . \mathsf{por}\ (\widehat{M} 0)\ (\widehat{N} 0) \\
\widehat{\mathsf{exists}\ M} &\equiv \lambda z^\iota . \mathsf{exists}\ (\lambda f^{\iota \to \iota} . \widehat{M}(\lambda g^{\iota \to \iota} . \lambda w^\iota . f(g0))0)
\end{aligned}$$

(where in the clause for cond *we have $N, P : \sigma_1 \to \cdots \to \sigma_r \to \iota$).*

Remark 6.3.11 A different translation from PCF_N to PCF_V appears in Riecke [83]. Our translation is in some sense structurally simpler, since at the level of types Riecke's translation gives $\iota' = \iota \to \iota$, $(\sigma \to \tau)' = \iota \to (\sigma' \to \tau')$.

Note as before that our translation restricts well to PCF and PCF^+, and that if $M : \sigma$ then $\widehat{M} : \widehat{\sigma}$. If Γ is an environment we define $\widehat{\Gamma}, \delta^\Gamma, \epsilon^\Gamma$ in the obvious way by analogy with $\overline{\Gamma}, \beta^\Gamma, \gamma^\Gamma$. The following proposition now gives the relationship between the translation and the embedding. Since the proof is very similar in outline to that of Proposition 6.3.9, we omit many of the details and concentrate on the points at which it differs.

Proposition 6.3.12 *Suppose $\mathcal{C}$ supports $\mathcal{L}$, and $M : \sigma$ is a term of $\mathcal{L}_N$ in environment Γ. Then $[\widehat{M}]_V^{\widehat{\Gamma}} = \delta^\sigma \circ [M]_N^\Gamma \circ \epsilon^\Gamma$. In particular, if M is closed then we have $[\![\, \widehat{M} \,]\!]_V = \delta^\sigma \circ [\![\, M \,]\!]_N$.*

Proof By induction on M. The case for variables is as before. The cases for n, succ and pred are easy: for instance, for succ we just check pointwise that $[\lambda z^\iota.\,\mathsf{succ}\;(-0)]_V \circ \delta^\iota = \delta^\iota \circ \mathit{succ}$. The cond case is somewhat delicate. Suppose $\sigma = \sigma_1 \to \cdots \to \sigma_r \to \iota$. First check by induction on σ that $\delta^\sigma \circ \bot = [\![\, \lambda x_1 \ldots x_r z.\Omega^\iota \,]\!]_V$ and that $\delta^\sigma = [\![\, \lambda x_1 \ldots x_r z.\; - x_1 \ldots x_r z \,]\!]_V \circ \delta^\sigma$; we can then verify pointwise that

$$[\widehat{\mathsf{cond}\; x\, y\, y'}]_V^{x,y,y'} \circ (\delta^\iota \times \delta^\sigma \times \delta^\sigma) \;=\; \delta^\sigma \circ \mathit{cond},$$

arguing by cases according to whether the first component is $\bot$, k_0 or k_{n+1}.

For the application case, let $\Delta = \langle x^{\sigma\to\tau}, y^\sigma \rangle$ as before. Then we have

$$\begin{aligned}
[xy]_V^{\widehat{\Delta}} \circ \delta^\Delta &= \mathit{eval} \circ ((\mathit{strict} \circ \alpha) \times \mathrm{id}) \circ (\delta^{\sigma\to\tau} \times \delta^\sigma) \\
&= \mathit{eval} \circ (\delta'^{\sigma\to\tau} \times \delta'^{\sigma}) \\
&= \eta \circ \mathit{eval} \circ ((\delta'^{\tau})^{\epsilon'^{\sigma}} \times \delta'^{\sigma}) \\
&= \eta \circ \delta'^{\tau} \circ \mathit{eval} \;=\; \delta^\tau \circ [xy]_N^\Delta.
\end{aligned}$$

For the λ-abstraction case, we have

$$\begin{aligned}
[\widehat{\lambda x^\sigma.M}]_V^{\widehat{\Gamma}} &= \eta \circ (C^{\widehat{\tau}})^\eta \circ \mathit{curry}(\delta^\tau \circ [M]_N^{\Gamma,x} \circ \epsilon^{\Gamma,x}) \\
&= \eta \circ (\delta^\tau)^{\epsilon'^{\sigma}} \circ \mathit{curry}([M]_N^{\Gamma,x}) \circ \epsilon^\Gamma \\
&= \delta^{\sigma\to\tau} \circ [\lambda x^\sigma.M]_N^\Gamma \circ \epsilon^\Gamma.
\end{aligned}$$

The case for the fixed point operator is precisely similar to the corresponding case in Proposition 6.3.9; and the case for por is easy. For exists it suffices to verify that

$$\widehat{[\mathsf{exists}\ x]}^{\widehat{x}}_{V} \circ \delta^{(\iota\to\iota)\to\iota} \;=\; \delta^{\iota} \circ [\mathsf{exists}\ x]^{x}_{N}.$$

It is tedious but straightforward to check this pointwise, by cases according to the value of $[\mathsf{exists}\ x]_N$. □

Finally we give a translation from lazy to call-by-name PCF, corresponding to the embeddings ζ^σ. Here we make use of the contexts S^σ, T^σ defined in Lemma 6.3.5. The idea behind this translation appears to be new.

Definition 6.3.13 (Translating L to N) *Recall* $\widetilde{\iota} = \iota$, $\widetilde{\sigma \to \tau} = \iota \to \widetilde{\sigma} \to \widetilde{\tau}$. *The translation* $M \mapsto \widetilde{M}$ *from* PCF^{++}_L *to* PCF^{++}_V *is defined as follows:*

$$\begin{array}{rcl}
\widetilde{x^\sigma} & \equiv & R^\sigma[x^{\widetilde{\sigma}}] \\
\widetilde{n} & \equiv & n \\
\widetilde{\mathsf{succ}\ M} & \equiv & \mathsf{succ}\ \widetilde{M} \\
\widetilde{\mathsf{pred}\ M} & \equiv & \mathsf{pred}\ \widetilde{N} \\
\widetilde{\mathsf{cond}\ MNP} & \equiv & \mathsf{cond}\ \widetilde{M}\ \widetilde{N}\ \widetilde{P} \\
\widetilde{MN} & \equiv & \widetilde{M}\ 1\ \widetilde{N} \\
\widetilde{\lambda x^\sigma.M} & \equiv & \lambda z^\iota.\,\mathsf{cond}\ z\ (S^{\widetilde{\sigma}\to\widetilde{\tau}}[0])\ (\lambda x^{\widetilde{\sigma}}.\widetilde{M}) \\
\widetilde{\mathsf{conv}\ MN} & \equiv & \lambda x_1^{\widetilde{\tau_1}}\ldots x_r^{\widetilde{\tau_r}}.\ conv\ (T^\sigma[\widetilde{M}])\ (\widetilde{N} x_1 \ldots x_r) \\
\widetilde{\mu x^\sigma.M} & \equiv & \mu x^{\widetilde{\sigma}}.\widetilde{M} \\
\widetilde{\mathsf{por}\ MN} & \equiv & \mathsf{por}\ \widetilde{M}\ \widetilde{N} \\
\widetilde{\mathsf{exists}\ M} & \equiv & \mathsf{exists}\ (\lambda f^{\iota\to\iota}.\,\widetilde{M}\ 1\ (\lambda z^\iota.\,\mathsf{cond}\ z\,(S^{\iota\to\iota}[0])\,f))
\end{array}$$

(where in the clause for conv *we have* $M : \sigma$, $N : \tau_1 \to \cdots \to \tau_r \to \iota$ *and* $conv \equiv (\lambda pq.\,\mathsf{cond}\ pqq)$ *).*

Again our translation restricts well to PCF and PCF^+, and if $M : \sigma$ then $\widetilde{M} : \widetilde{\sigma}$. Defining $\widetilde{\Gamma}, \zeta^\Gamma, \xi^\Gamma$ in the obvious way, we have the following relationship between the translation and the embedding.

Proposition 6.3.14 *Suppose $\mathcal{C}$ supports $\mathcal{L}$, and $M : \sigma$ is a term of $\mathcal{L}_L$ in environment Γ. Then $[\widetilde{M}]_N^{\widehat{\Gamma}} = \zeta^\sigma \circ [M]_L^\Gamma \circ \xi^\Gamma$. In particular, if M is closed then we have $[\![\widetilde{M}]\!]_N = \zeta^\sigma \circ [\![M]\!]_L$.*

Proof By induction on M. The cases for variables, numerals, succ, pred and cond are precisely similar to the corresponding cases of Proposition 6.3.9. For application, taking $\Delta = \langle x^{\sigma\to\tau}, y^\sigma \rangle$ we have

$$\begin{aligned}
[x\,1\,y]_N^{\widetilde{\Delta}} \circ \zeta^\Delta &= eval \circ \langle eval \circ \langle \pi^{\widetilde{\sigma\to\tau}}, k_1 \rangle, \pi^{\widetilde{\sigma}} \rangle \circ (\zeta^{\sigma\to\tau} \times \zeta^\sigma) \\
&= eval \circ ((eval \circ \langle \phi^{\widetilde{\sigma}\to\widetilde{\tau}} \circ (((\zeta^\tau)^{\xi^\sigma})_\perp, k_1 \rangle) \times \zeta^\sigma) \\
&= eval \circ ((\alpha \circ ((\zeta^\tau)^{\xi^\sigma})_\perp) \times \zeta^\sigma) \\
&= \zeta^\tau \circ eval \circ (\alpha \times \mathrm{id}) \;=\; \zeta^\tau \circ [xy]_L^\Delta.
\end{aligned}$$

For λ-abstraction we have

$$\begin{aligned}
[\widetilde{\lambda x^\sigma . M}]_N^{\widetilde{\Gamma}} &= curry_{N_\perp}(cond \circ \langle \pi_{N_\perp}, s^{\widetilde{\sigma}\to\widetilde{\tau}} \circ k_0 \circ u,\; curry([\widetilde{M}]_N^{\widetilde{\Gamma, z^\iota, x}}) \rangle) \\
&= \phi^{\widetilde{\sigma}\to\widetilde{\tau}} \circ \eta \circ curry(\zeta^\tau \circ [M]_L^{\Gamma,z,x} \circ \xi^{\Gamma,z,x}) \\
&= \phi^{\widetilde{\sigma}\to\widetilde{\tau}} \circ ((\zeta^\tau)^{\xi^\sigma})_\perp \circ \eta \circ curry([M]_L^{\Gamma,z,x}) \circ \xi^\Gamma \\
&= \zeta^{\sigma\to\tau} \circ [\lambda x^\sigma . M]_L^\Gamma \circ \xi^\Gamma .
\end{aligned}$$

For conv, suppose $\tau = \tau_1 \to \cdots \to \tau_r \to \iota$. We wish to show that

$$[\widetilde{\mathsf{conv}\ x\, y}]_N^{\widetilde{x^\sigma, y^\tau}} \circ (\zeta^\sigma \times \zeta^\tau) \;=\; \zeta^\tau \circ conv;$$

we do this by checking that these morphisms agree on all global elements $\langle x, y \rangle$. If $x = \perp$ then $[\![T^\sigma]\!]_N \circ \zeta^\sigma \circ x = \perp$ and so we have

$$[\widetilde{\mathsf{conv}\ x\, y}]_N^{x,y} \circ (\zeta^\sigma x \times \zeta^\tau y) \;=\; \perp \;=\; \zeta^\tau \circ \perp \;=\; \zeta^\tau \circ conv \circ \langle x, y \rangle.$$

Otherwise we have $[\![T^\sigma]\!]_N \circ \zeta^\sigma \circ x \neq \perp$ and so

$$\begin{aligned}
[\widetilde{\mathsf{conv}\ x\, y}]_N^{x,y} \circ (\zeta^\sigma x \times \zeta^\tau y) &= [\lambda x_1 \dots x_r .\, y x_1 \dots x_r]_N^y \circ \zeta^\tau \circ y \\
&= \zeta^\tau \circ y \;=\; \zeta^\tau \circ conv \circ \langle x, y \rangle.
\end{aligned}$$

The cases for μ and por are exactly analogous to the corresponding cases in Proposition 6.3.9. For exists we need to show that $[\widetilde{\mathsf{exists}\ x}]_N^{\widetilde{x}} \circ \zeta^{(\iota\to\iota)\to\iota} = \zeta^\iota \circ [\mathsf{exists}\ x]_L^x$; this can be checked pointwise using a case split as before. $\square$

Remark 6.3.15 Note that this last translation is not *functional* in the sense of Riecke [83]—indeed he proves (in effect) that there can be *no* functional translation from PCF_L to PCF_N. This is essentially because the definition of functional translation requires that there should be a term-context $D[-]$ such that $\widetilde{MN}$ and $D[\widetilde{M}]\widetilde{N}$ are observationally equivalent for all M, N, and also $M \Downarrow$ iff $D[\widetilde{M}] \Downarrow$; whereas (under Riecke's conventions) all PCF_N-terms of higher type converge. Riecke concludes that PCF_N is in some sense less expressive than PCF_L and PCF_V.

In our view, however, the above translation is a perfectly respectable one, and it is the definition of functional translation that is somewhat unnatural. In particular, it seems arbitrary to insist that the context $D[-]$ used in translating applications should be the same as the context used for performing observations on translated terms. It seems that under a minor modification of Riecke's definition the above translation is indeed "functional".

6.4 Full abstraction and universality

We now apply the above results to obtain theorems concerning fully abstract and universal models for our languages. We first recall some standard definitions:

Definition 6.4.1 (Full abstraction) *Suppose $\mathcal{L}$ is one of our nine languages, and $\mathcal{C}$ is a PCF-model supporting $\mathcal{L}$.*

(i) If $M, N : \sigma$ are terms of $\mathcal{L}$, we say M, N are observationally equivalent *(and write $M \approx N$ or $M \approx_{\mathcal{L}} N$) if for all contexts $D[-] : \iota$ with hole of type σ and no other free variables we have*

$$D[M] \Downarrow n \iff D[N] \Downarrow n.$$

(ii) We say the interpretation of $\mathcal{L}$ in $\mathcal{C}$ is (equationally) fully abstract *if for all closed terms $M, N : \sigma$ we have*

$$M \approx N \implies [\![M]\!] = [\![N]\!].$$

The converse implication $[\![M]\!] = [\![N]\!] \Rightarrow M \approx N$ holds automatically since our interpretation of $\mathcal{L}$ is compositional. The reader may like to check that (for each of our languages) we obtain an equivalent definition of full abstraction if we omit the word "closed" in the above; however, the definition we have given is slightly more convenient for our purposes. The next definition is also fairly standard:

Definition 6.4.2 (Universality) *Suppose $\mathcal{L}$ is one of our languages and $\mathcal{C}$ supports $\mathcal{L}$. We say the interpretation of $\mathcal{C}$ in $\mathcal{L}$ is* universal *if, for each type σ, every morphism $1 \to C^{\sigma}_{\mathcal{L}}$ is of the form $[\![M]\!]$ for some closed term $M : \sigma$ of $\mathcal{L}$.*

Again it is an easy exercise to check that this is equivalent to the following condition involving open terms: For each type σ and environment Γ, every morphism $[\![\Gamma]\!] \to [\![\sigma]\!]$ is of the form $[\![M]\!]^{\Gamma}$ for some term $M : \sigma$ in environment Γ.

The following relationship between these two notions is well-known:

Proposition 6.4.3 *Suppose $\mathcal{C}$ is a well-pointed PCF-model supporting $\mathcal{L}$. If the interpretation of $\mathcal{L}$ in $\mathcal{C}$ is universal then it is fully abstract.*

Proof By induction on types. We prove the result for the call-by-value case, the other two cases being similar but easier. At type ι the result is trivial, since if $M \approx N$ then $M \Downarrow n \Leftrightarrow N \Downarrow n$ and so $[\![M]\!] = [\![N]\!]$. So assume the result holds at type τ and suppose we have $M, N : \sigma \to \tau$ such that $M \approx N$. First, if $M \Uparrow$ then also $N \Uparrow$ and we have $[\![M]\!] = [\![N]\!] = \bot$. Otherwise $M \Downarrow$ and $N \Downarrow$, so there exist $f, g : V^{\sigma}_{V} \to C^{\tau}_{V}$ such that $[\![M]\!] = \eta \circ \overline{f}$ and $[\![N]\!] = \eta \circ \overline{g}$. It suffices to show that $f \circ x = g \circ x$ for all $x : 1 \to V^{\sigma}_{V}$. Given x, we may take $P : \sigma$ such that $[\![P]\!] = \eta \circ x$. Then $[\![MP]\!] = f \circ x$ and $[\![NP]\!] = g \circ x$. But we have $MP \approx NP$ and so $[\![MP]\!] = [\![NP]\!]$ by the induction hypothesis. $\square$

We will show in this section that from the point of view of full abstraction and universality the three evaluation mechanisms may be regarded as "equivalent". More specifically:

Theorem 6.4.4 *Suppose $\mathcal{C}$ is a standard PCF-model; suppose also that $\mathcal{L}_1, \mathcal{L}_2$ are two languages with the same degree of parallelism, and that $\mathcal{C}$ supports $\mathcal{L}_1, \mathcal{L}_2$. Then*

(i) $\mathcal{C}$ is fully abstract for $\mathcal{L}_1$ iff it is fully abstract for $\mathcal{L}_2$;

(ii) $\mathcal{C}$ is universal for $\mathcal{L}_1$ iff it is universal for $\mathcal{L}_2$.

We prove this theorem by chasing round the triangle given by our three translations: we use the translation from $\mathcal{L}_L$ to $\mathcal{L}_N$ to show that if $\mathcal{C}$ is fully abstract for $\mathcal{L}_L$ then it is fully abstract for $\mathcal{L}_N$, and so on. In order to achieve this, however, we need some rather technical lemmas, asserting (for instance) that if we compose our embeddings β, δ, ζ going round the triangle then the composite morphism is definable by a term context of the appropriate language. Since fully formal proofs of all these lemmas would be very tedious, and the lemmas themselves are so unsurprising, the proofs we give will be somewhat condensed. On a first reading, the reader is advised to skip the proofs of these lemmas completely and to concentrate on the proof of Proposition 6.4.6.

6.4.1 The definability lemmas

We start by using the translation from N to V to relate full abstraction and universality properties for $\mathcal{L}_V$ and $\mathcal{L}_N$. The appropriate definability lemma is the following:

Lemma 6.4.5 *Let $\sigma^* = ((\widehat{\sigma})^-)^\smile$. Then (omitting some superscripts) the composites $\zeta\beta\delta : C_N^\sigma \to C_N^{\sigma^*}$ and $\epsilon\gamma\xi : C_N^{\sigma^*} \to C_N^\sigma$ are denotable by contexts of* PCF_N.

Proof First note that

$$\iota^* \;=\; \iota \to \iota \to \iota, \qquad (\sigma \to \tau)^* \;=\; \iota \to \sigma^* \to \tau^*.$$

Write $\Phi^\sigma = \zeta\beta\delta : C_N^\sigma \to C_N^{\sigma^*}$ and $\Psi^\sigma = \epsilon\gamma\xi : C_N^{\sigma^*} \to C_N^\sigma$. We will inductively define contexts $X^\sigma[-], Y^\sigma[-]$ such that $[\![\, X^\sigma \,]\!]_N = \Phi^\sigma$ and $[\![\, Y^\sigma \,]\!]_N = \Psi^\sigma$. For the base case, take

$$X^\iota[x^\iota] \;\equiv\; \lambda z^\iota.\, \mathsf{cond}\; z\, (\lambda w^\iota.0)\, (\lambda w^\iota.\, conv\, w\; x),$$
$$Y^\iota[y^{\iota\to\iota\to\iota}] \;\equiv\; conv\, (y\, 0\, 0)\; (y\, 1\, 0).$$

It is easy to check pointwise that these have the required properties. For $X^{\sigma\to\tau}$, first check that $\Phi^{\sigma\to\tau}$ is equal to the composite

$$C_N^{\sigma\to\tau} \xrightarrow{(\Phi^\tau)^{\Psi^\sigma}} C_N^{\sigma^*\to\tau^*} \xrightarrow{\phi^{\sigma^*\to\tau^*}\circ\eta} C_N^{\iota\to\sigma^*\to\tau^*}.$$

But the two morphisms here are denoted respectively by the contexts

$$A[x^{\sigma\to\tau}] \;\equiv\; \lambda y^{\sigma^*}.\, X^\tau[x(Y^\sigma[y])], \qquad B[x^{\sigma^*\to\tau^*}] \;\equiv\; \lambda z^\iota.\, \mathsf{cond}\; z\; (S^{\sigma^*\to\tau^*}[0])\; x.$$

So taking $X^{\sigma\to\tau}[-] \equiv B[A[-]]$ gives a context with the desired property. Likewise for $Y^{\sigma\to\tau}$, first check that $\Psi^{\sigma\to\tau}$ is equal to the composite

$$C_N^{\iota\to\sigma^*\to\tau^*} \xrightarrow{\alpha\circ\psi^{\sigma^*\to\tau^*}} C_N^{\sigma^*\to\tau^*} \xrightarrow{(\Psi^\tau)^{\Phi^\sigma}} C_N^{\sigma\to\tau}.$$

These two morphisms are denoted respectively by

$$\begin{aligned} B'[x^{(\sigma\to\tau)^*}] \;&\equiv\; \lambda y^{\sigma^*}.\lambda z_1 \ldots z_r.\, conv\,(T^{\sigma^*\to\tau^*}[x0])(x\,1\,y\,z_1\ldots z_r), \\ A'[x^{\sigma^*\to\tau^*}] \;&\equiv\; \lambda y^\sigma.\, Y^\tau[x(X^\sigma[y])]. \end{aligned}$$

So the context $Y^{\sigma\to\tau}[-] \equiv A'[B'[-]]$ has the desired property. □

Once we have this definability lemma, the relationships between the programming languages follow for quite general reasons. Here again, $\mathcal{L}$ stands for one of the tokens PCF, PCF$^+$, PCF^{++}.

Proposition 6.4.6 *(i) If $\mathcal{C}$ is fully abstract for $\mathcal{L}_L$ then it is fully abstract for $\mathcal{L}_N$.*

(ii) If $\mathcal{C}$ is universal for $\mathcal{L}_L$ then it is universal for $\mathcal{L}_N$.

Proof (i) Assume $\mathcal{C}$ is fully abstract for $\mathcal{L}_L$, and suppose $M, N : \sigma$ are closed terms of $\mathcal{L}_N$ with $M \approx N$. Let $M' \equiv (\widehat{M})^-$, $N' \equiv (\widehat{N})^-$ and $\sigma' = (\widehat{\sigma})^-$. We will first show that $M' \approx N'$ in $\mathcal{L}_L$. Suppose $D[-] : \iota$ is a context of $\mathcal{L}_L$ with hole of type σ'. Since the translation $\sim : \mathcal{L}_L \to \mathcal{L}_N$ is compositional, there is clearly a context $\widetilde{D}[-]$ of $\mathcal{L}_N$ such that $\widetilde{D}[\widetilde{P}] \equiv \widetilde{D[P]}$ for all $P : \sigma'$ in $\mathcal{L}_V$. So by Proposition 6.3.14 we have $[\![\, \widetilde{D}[\widetilde{P}] \,]\!]_N = \zeta^\iota \circ [\![\, D[P] \,]\!]_L = [\![\, D[P] \,]\!]_L$ since $\zeta^\iota = \mathrm{id}$. Now suppose $X^\sigma[-]$ denotes $\zeta\beta\delta$ as in the above lemma, and let $E[-] \equiv \widetilde{D}[X[-]]$; then since $M \approx N$ we have

$$[\![\, D[M'] \,]\!]_L \;=\; [\![\, E[M] \,]\!]_N \;=\; [\![\, E[N] \,]\!]_N \;=\; [\![\, D[N'] \,]\!]_L$$

since $M \approx N$ and $E[M], E[N]$ are of type ι. Hence, by the adequacy theorem for $\mathcal{L}_L$, we have $D[M'] \Downarrow n$ iff $D[N'] \Downarrow n$. Thus $M' \approx N'$. But now since $\mathcal{C}$ is fully abstract for $\mathcal{L}_L$ we have $[\![M']\!]_L = [\![N']\!]_L$, and so using Propositions 6.3.3 and 6.3.4 we have

$$[\![M]\!]_N \;=\; \epsilon\gamma\beta\delta \circ [\![M]\!]_N \;=\; \epsilon\gamma \circ [\![M']\!]_L \;=\; \epsilon\gamma \circ [\![N']\!]_L \;=\; [\![N]\!]_N.$$

(ii) Assume $\mathcal{C}$ is universal for $\mathcal{L}_L$. Given a morphism $x : 1 \to C_N^\sigma$, let $y = \beta\delta \circ x$ and take $M : \sigma'$ in $\mathcal{L}_L$ such that $[\![M]\!]_L = y$. Then $[\![\widetilde{M}]\!]_N = \zeta \circ y$ by Proposition 6.3.14. Now suppose $Y^\sigma[-]$ denotes $\epsilon\gamma\xi$ as in the lemma. Then

$$[\![Y[\widetilde{M}]]\!]_N \;=\; \epsilon\gamma\xi\zeta \circ y \;=\; \epsilon\gamma \circ y \;=\; x. \quad \Box$$

Next we prove that a similar relationship holds between $\mathcal{L}_N$ and $\mathcal{L}_V$—for this we need a lemma asserting that certain morphisms are definable in $\mathcal{L}_V$. However, we have to work slightly harder to obtain this lemma than we did for Lemma 6.4.5. First we need the following sublemma, which says essentially that under the embedding of C_N^σ in $V_V^{\widehat{\sigma}}$ the morphisms ϕ^σ, ψ^σ can be "simulated" by morphisms $\widehat{\phi}^\sigma, \widehat{\psi}^\sigma$:

Lemma 6.4.7 *For each type σ, there exist morphisms $\widehat{\phi}^\sigma, \widehat{\psi}^\sigma$ such that the following squares commute:*

$$\begin{tikzcd}
(C_N^\sigma)_\perp \arrow[r, "\phi^\sigma"] \arrow[d, "\delta'^\sigma_\perp"'] & (N_\perp \to C_N^\sigma) \arrow[d, "(\delta'^\sigma)^{N_\perp}"] \\
(V_V^{\widehat{\sigma}})_\perp \arrow[r, "\widehat{\phi}^\sigma"'] & (N_\perp \to V_V^{\widehat{\sigma}})
\end{tikzcd}
\qquad
\begin{tikzcd}
(C_N^\sigma)_\perp & (N_\perp \to C_N^\sigma) \arrow[l, "\psi^\sigma"'] \\
(V_V^{\widehat{\sigma}})_\perp \arrow[u, "\epsilon'^\sigma_\perp"] & (N_\perp \to V_V^{\widehat{\sigma}}) \arrow[l, "\widehat{\psi}^\sigma"] \arrow[u, "(\epsilon'^\sigma)^{N_\perp}"']
\end{tikzcd}$$

Moreover, $\widehat{\phi}^\sigma, \widehat{\psi}^\sigma$ are "essentially denotable" in PCF_V*, in the sense that there are contexts $Z^\sigma[-^{\widehat{\sigma}}, -^\iota]$ and $W^\sigma[-^{\iota \to \widehat{\sigma}}]$ of* PCF_V *such that*

$$\begin{aligned}
\eta^{N_\perp} \circ \widehat{\phi}^\sigma &= curry_{N_\perp}([\![Z^\sigma]\!]_V) &: C_V^\sigma \to (N_\perp \to C_V^{\widehat{\sigma}}),\\
\widehat{\psi}^\sigma &= [\![W^\sigma]\!]_V \circ \eta \circ (\eta^\eta) &: (N_\perp \to V_V^{\widehat{\sigma}}) \to C_V^{\widehat{\sigma}}.
\end{aligned}$$

Proof Suppose $\sigma = \sigma_1 \to \cdots \to \sigma_r \to \iota$. First define contexts $\widehat{S}^\sigma[-], \widehat{T}^\sigma[-]$ by

$$\widehat{S}^\sigma[z^\iota] \;\equiv\; \lambda x_1^{\widehat{\sigma_1}} \ldots x_r^{\widehat{\sigma_r}}.\lambda y^\iota.z, \qquad \widehat{T}^\sigma[x^{\widehat{\sigma}}] \;\equiv\; x\,(\widehat{S}^{\sigma_1}\Omega^\iota)\ldots(\widehat{S}^{\sigma_r}\Omega^\iota)\,0.$$

Clearly there are unique morphisms $\widehat{s}^\sigma : N_\perp \to V_V^{\widehat{\sigma}}$ and $\widehat{t}^\sigma : V_V^{\widehat{\sigma}} \to N_\perp$ such that $[\![\, \widehat{S}^\sigma \,]\!] = \eta \circ \widehat{s}^\sigma$ and $[\![\, \widehat{T}^\sigma \,]\!] \circ \eta = \widehat{t}^\sigma$. It is easily checked that

$$\widehat{s}^\sigma \;=\; \delta'^\sigma \circ s^\sigma, \qquad \widehat{t}^\sigma \;=\; t^\sigma \circ \epsilon'^\sigma, \qquad \widehat{t}^\sigma \circ \widehat{s}^\sigma \;=\; \mathrm{id}_{N_\perp}.$$

Now define contexts Z^σ and W^σ by

$$\begin{aligned} Z^\sigma[x^{\widehat{\sigma}}, z^\iota] &\equiv \lambda x_1 \ldots x_r.\lambda y^\iota.\,\mathsf{cond}\; z\; (\widehat{T}^\sigma[x])\, (x x_1 \ldots x_r y), \\ W^\sigma[x^{\iota \to \widehat{\sigma}}] &\equiv \mathit{conv}'\; (\widehat{T}^\sigma[x\,0])\, (\lambda y.\, x\,1\,y) \end{aligned}$$

where $\mathit{conv}' \equiv (\lambda pq.q)$; and let $\widehat{\phi}^\sigma, \widehat{\psi}^\sigma$ be the morphisms determined by the above two equations. Then it is straightforward to check pointwise that the two squares commute. □

Lemma 6.4.8 *Let* $\sigma^+ = ((\overline{\sigma})\check{\ })\hat{\ }$. *Then the composites* $\delta\zeta\beta : C_V^\sigma \to C_V^{\sigma^+}$ *and* $\gamma\xi\epsilon : C_V^{\sigma^+} \to C_V^\sigma$ *are denotable by contexts of* PCF_V.

Proof First note that $\iota^+ = \iota \to \iota$ and $(\sigma \to \tau)^+ = (\iota \to \iota) \to \sigma^+ \to \tau^+$. Write $\Phi^\sigma = \delta\zeta\beta$, $\Psi^\sigma = \gamma\xi\epsilon$, $\Phi'^\sigma = \delta'\zeta\beta$ and $\Psi'^\sigma = \gamma\xi\epsilon'$. We will inductively define contexts X^σ, Y^σ denoting Φ^σ, Ψ^σ respectively. For the base case, take

$$X^\iota[x^\iota] \;\equiv\; \lambda z^\iota.x, \qquad Y^\iota[y^{\iota\to\iota}] \;\equiv\; y\,0.$$

It is easy to see that these have the required properties. For $X^{\sigma\to\tau}$, we first check via a diagram-chase involving the previous lemma that $\Phi^{\sigma\to\tau}$ is equal to the composite

$$C_V^{\sigma\to\tau} \xrightarrow{\;\mathit{strict}_\perp\;} (C_V^\sigma \to C_V^\tau)_\perp \xrightarrow{\;((\Phi^\tau)^{\Psi'^\sigma})_\perp\;} C_V^\rho \cdots$$

$$\cdots \xrightarrow{\;\eta^{N_\perp} \circ \widehat{\phi}^\rho\;} (N_\perp \to C_V^\rho) \xrightarrow{\;\eta \circ (C_V^\rho)^{\epsilon'^\iota}\;} C_V^{(\sigma\to\tau)^+}.$$

where $\rho = \sigma^+ \to \tau^+$. But here the composites of the first two and the last two morphisms are denoted respectively by the contexts

$$A[x^{\sigma\to\tau}] \;\equiv\; \mathit{conv}'\; x\; (\lambda y^{\sigma^+}.\, X^\tau[x(Y^\sigma[y])]), \qquad B[x^\rho] \;\equiv\; \lambda f^{\iota\to\iota}.\, Z^\rho[x, f0]$$

with Z^ρ as in the previous lemma. Thus $X^{\sigma\to\tau}[-] \equiv B[A[-]]$ denotes $\Phi^{\sigma\to\tau}$. Likewise for $Y^{\sigma\to\tau}$, we check via a diagram-chase that $\Psi^{\sigma\to\tau}$ is equal to the composite

$$C_V^{(\sigma\to\tau)^+} \xrightarrow{\;(C_V^\rho)^{\delta'^\iota} \circ \alpha\;} (N_\perp \to C_V^\rho) \xrightarrow{\;\widehat{\psi}^\rho \circ \alpha^{N_\perp}\;} C_V^\rho \cdots$$

$$\cdots \xrightarrow{\;((\Psi^\tau)^{\Phi'^\sigma})_\perp\;} (C_V^\sigma \to C_V^\tau)_\perp \xrightarrow{\;((C_V^\tau)^\eta)_\perp\;} C_V^{\sigma\to\tau}.$$

Here the composites of the first two and the last two morphisms are denoted respectively by the contexts

$$B'[x^{(\sigma\to\tau)^+}] \equiv W^\rho[\lambda z^\iota.\, x(\lambda y^\iota.z)], \qquad A'[x^\rho] \equiv \mathit{conv}'\ x\ (\lambda y^\sigma.\, Y^\tau[x(X^\sigma[y])]).$$

Thus $Y^{\sigma\to\tau}[-] \equiv A'[B'[-]]$ denotes $\Psi^{\sigma\to\tau}$. □

We can now obtain the result for $\mathcal{L}_N$ and $\mathcal{L}_V$ analogous to Proposition 6.4.6:

Proposition 6.4.9 *(i) If $\mathcal{C}$ is fully abstract for $\mathcal{L}_N$, it is fully abstract for $\mathcal{L}_V$.*

(ii) If $\mathcal{C}$ is universal for $\mathcal{L}_N$, it is universal for $\mathcal{L}_V$.

Proof (i) The proof of Proposition 6.4.6(i) works with just one small modification. Recall that there we exploited the fact that $\widetilde{\iota} = \iota$ and $\zeta^\iota = \mathrm{id}$. But here we have $\widehat{\iota} = \iota \to \iota$. However, we do have $[\![-0]\!]_V \circ \delta^\iota = \mathrm{id}_{N_\perp}$, and so at the corresponding juncture in the proof we have $[\![\widehat{D}[\widehat{P}]\,0]\!]_V = [\![D[P]]\!]_N$. Thus we need to define $E[-] \equiv \widehat{D}[X[-]]\,0$, and the rest of the proof works as before.

(ii) Exactly similar to the proof of Proposition 6.4.6(ii). □

Finally we prove the corresponding relationship between $\mathcal{L}_V$ and $\mathcal{L}_L$. In this case, to obtain the appropriate definability lemma we need both Lemma 6.4.7 and a further sublemma:

Lemma 6.4.10 *For each type σ, there exist morphisms $\overline{\hat\phi}^\sigma, \overline{\hat\psi}^\sigma$ such that the following squares commute:*

$$\begin{tikzcd}
C_V^{\widehat{\sigma}} \arrow[r, "\eta^{N_\perp} \circ \widehat{\phi}^\sigma"] \arrow[d, "\beta^{\widehat{\sigma}}"'] & (N_\perp \to C_V^{\widehat{\sigma}}) \arrow[d, "(\beta^{\widehat{\sigma}})^{N_\perp}"] \\
C_L^{\overline{\widehat{\sigma}}} \arrow[r, "\overline{\hat\phi}^\sigma"] & (N_\perp \to C_L^{\overline{\widehat{\sigma}}})
\end{tikzcd}
\qquad
\begin{tikzcd}
C_V^{\widehat{\sigma}} & (N_\perp \to C_V^{\widehat{\sigma}}) \arrow[l, "\widehat{\psi}^\sigma \circ \alpha^{N_\perp}"'] \\
C_L^{\overline{\widehat{\sigma}}} \arrow[u, "\gamma^{\widehat{\sigma}}"] & (N_\perp \to C_L^{\overline{\widehat{\sigma}}}) \arrow[l, "\overline{\hat\psi}^\sigma"'] \arrow[u, "(\gamma^{\widehat{\sigma}})^{N_\perp}"']
\end{tikzcd}$$

Moreover, the morphisms $\eta \circ \overline{\hat\phi}^\sigma : C_L^{\overline{\widehat{\sigma}}} \longrightarrow C_L^{\iota\to\overline{\widehat{\sigma}}}$ and $\overline{\hat\psi}^\sigma \circ \alpha : C_L^{\iota\to\overline{\widehat{\sigma}}} \longrightarrow C_L^{\overline{\widehat{\sigma}}}$ are denotable by contexts $\overline{Z}^\sigma, \overline{W}^\sigma$ of PCF_L *respectively.*

Proof Suppose $\widehat{\sigma} = \nu_1 \to \cdots \to \nu_r \to \iota$. First define contexts $\overline{\hat S}^\sigma[-], \overline{\hat T}^\sigma[-]$ by

$$\begin{aligned}
\overline{\hat S}^\sigma[z^\iota] &\equiv \lambda x_1^{\overline{\nu_1}}.\, \mathsf{conv}\ x_1\ (\ldots(\lambda x_r^{\overline{\nu_r}}.\, \mathsf{conv}\ x_r\ z)\ldots), \\
\overline{\hat T}^\sigma[x^{\overline{\widehat{\sigma}}}] &\equiv x\,(\overline{\hat S}^{\nu_1}\,\Omega^\iota)\ldots(\overline{\hat S}^{\nu_r}\,\Omega^\iota).
\end{aligned}$$

and let $\overline{\hat{s}}^\sigma = [\![\overline{\hat{S}}^\sigma]\!]_L$, $\overline{\hat{t}}^\sigma = [\![\overline{\hat{T}}^\sigma]\!]_L$. It is easy to check that

$$\overline{\hat{s}}^\sigma \;=\; \beta^{\widehat{\sigma}} \circ \eta \circ \widehat{s}^\sigma, \qquad \overline{\hat{t}}^\sigma \;=\; \widehat{t}^\sigma \circ \alpha \circ \gamma^{\widehat{\sigma}}, \qquad \overline{\hat{t}}^\sigma \circ \overline{\hat{s}}^\sigma = \mathrm{id}_{N_\perp}.$$

Now define contexts $\overline{Z}^\sigma$ and $\overline{W}^\sigma$ by

$$\begin{aligned} \overline{Z}^\sigma[x^{\overline{\widehat{\sigma}}}] \;&\equiv\; \lambda z^\iota.\lambda x_1.\mathsf{conv}\; x_1(\ldots(\lambda x_r.\mathsf{conv}\; x_r(\mathsf{cond}\; z(\overline{\hat{T}}^\sigma[x])(x x_1 \ldots x_r)))\ldots), \\ \overline{W}^\sigma[x^{\iota\to\overline{\widehat{\sigma}}}] \;&\equiv\; \mathsf{conv}\;(\overline{\hat{T}}^\sigma[x\,0])\;(\lambda y.\, x\,1\,y), \end{aligned}$$

and let $\overline{\hat{\phi}}^\sigma, \overline{\hat{\psi}}^\sigma$ be the morphisms determined by the equations $\eta \circ \overline{\hat{\phi}}^\sigma = [\![\overline{Z}^\sigma]\!]_L$ and $\overline{\hat{\psi}}^\sigma \circ \alpha \;=\; [\![\overline{W}^\sigma]\!]_L$. Then it is routine to check pointwise that the above squares commute. $\square$

Lemma 6.4.11 *Let* $\sigma^! = ((\widetilde{\sigma})\widehat{\ })^-$. *Then the composites* $\beta\delta\zeta : C_L^\sigma \to C_L^{\sigma^!}$ *and* $\xi\epsilon\gamma : C_L^{\sigma^!} \to C_L^\sigma$ *are denotable by contexts of* PCF_L.

Proof First note that $\iota^! = \iota \to \iota$ and $(\sigma \to \tau)^! = (\iota \to \iota) \to \sigma^! \to \tau^!$. Let $\Phi^\sigma = \beta\delta\zeta$, $\Psi^\sigma = \xi\epsilon\gamma$; we will construct contexts X^σ, Y^σ denoting Φ^σ, Ψ^σ respectively. First define

$$X^\iota[x^\iota] \;\equiv\; \lambda z^\iota.\,\mathsf{conv}\; z\; x, \qquad Y^\iota[y^{\iota\to\iota}] \;\equiv\; y\;0.$$

It is easy to check that these have the required properties. For $X^{\sigma\to\tau}$, a routine diagram-chase involving Lemmas 6.4.7 and 6.4.10 shows that $\Phi^{\sigma\to\tau}$ is equal to the composite

$$C_L^{\sigma\to\tau} \xrightarrow{((\Phi^\tau)^{\Psi^\sigma})_\perp} C_L^\rho \xrightarrow{\eta\,\circ\,\overline{\hat{\phi}}^{(\widetilde{\sigma}\to\widetilde{\tau})}} C_L^{\iota\to\rho} \xrightarrow{strict_\perp\,\circ\,((C_L^\rho)^{e_0})_\perp} C_L^{(\sigma\to\tau)^!}$$

where $\rho = \sigma^! \to \tau^!$ and $e_0 : (N_\perp)^{N_\perp} \to N_\perp$ is given by evaluation at zero. But the three morphisms here are denoted respectively by the contexts

$$\mathsf{conv}\;-\;(\lambda y^{\sigma^!}.\,X^\tau[-(Y^\sigma[y])]), \qquad \overline{Z}^\sigma[-], \qquad \mathsf{conv}\;-\;(\lambda f^{\iota\to\iota}.\,-(f\,0)).$$

So by composing these we get a context $X^{\sigma\to\tau}$ denoting $\Phi^{\sigma\to\tau}$. For $Y^{\sigma\to\tau}$, a similar diagram-chase shows that $\Psi^{\sigma\to\tau}$ is equal to the composite

$$C_L^{(\sigma\to\tau)^!} \xrightarrow{((C_L^\rho)^{\eta\circ K})_\perp} C_L^{\iota\to\rho} \xrightarrow{\overline{\hat{\psi}}^{(\widetilde{\sigma}\to\widetilde{\tau})}\,\circ\,\alpha} C_L^\rho \xrightarrow{((\Psi^\tau)^{\Phi^\sigma})_\perp} C_L^{\sigma\to\tau}$$

where $K : N_\perp \to (N_\perp)^{N_\perp}$ is the "constant" morphism. But these three morphisms are denoted respectively by the contexts

$$\mathsf{conv}\ -\ (\lambda z^\iota.\ -(\lambda y^\iota.z)), \qquad \overline{W}^\sigma[-], \qquad \mathsf{conv}\ -\ (\lambda y^\sigma.\,Y^\tau[-(X^\sigma[y])]).$$

So composing these we get a context $Y^{\sigma\to\tau}$ as required. □

We thus obtain the following relationship between $\mathcal{L}_V$ and $\mathcal{L}_L$:

Proposition 6.4.12 *(i) If $\mathcal{C}$ is fully abstract for $\mathcal{L}_V$, it is fully abstract for $\mathcal{L}_L$.*
(ii) If $\mathcal{C}$ is universal for $\mathcal{L}_V$, it is universal for $\mathcal{L}_L$.

PROOF The proof is exactly similar to that of Proposition 6.4.6. (Note that we have $\bar{\iota} = \iota$ and $\beta^\iota = \mathrm{id}$, so the extra twist required in the proof of Proposition 6.4.9(i) is not needed here.) □

The proof of Theorem 6.4.4 is now complete.

6.4.2 A more abstract approach

The results in this section and the last show that our three evaluation mechanisms are equivalently powerful in some sense. However, our proofs using syntactic translations have been rather messy, and we would anticipate that our results could be presented more cleanly in abstract categorical terms.

One idea would be as follows. Given a language $\mathcal{L}$, it should be possible to construct a "syntactic PCF-model" $\mathcal{C}(\mathcal{L})$ for $\mathcal{L}$, whose objects are freely generated from 1 and N using products, exponentials and lifting, and whose morphisms are generated by the terms of $\mathcal{L}$. One would then expect to be able to show that the canonical interpretation of $\mathcal{L}$ in $\mathcal{C}(\mathcal{L})$ was universal, both in the sense of Definition 6.4.2 and in the sense that adequate interpretations of $\mathcal{L}$ in $\mathcal{D}$ corresponded precisely to functors $\mathcal{C}(\mathcal{L}) \to \mathcal{D}$ preserving the PCF-model structure. (This is analogous to the situation in categorical logic, where interpretations of a theory T in a category $\mathcal{D}$ correspond to certain functors $\mathcal{C}(T) \to \mathcal{D}$, where $\mathcal{C}(T)$ is the *classifying* model for T—see e.g. [24].) Moreover, the translations $\mathcal{L}_1 \to \mathcal{L}_2$ of the previous section should induce functors $\mathcal{C}(\mathcal{L}_1) \to \mathcal{C}(\mathcal{L}_2)$, and the contexts X^σ, Y^σ given in this section should give rise to natural transformations.

Now suppose that $\mathcal{L}_1, \mathcal{L}_2$ have the same degree of parallelism. Then on the grounds of Theorem 6.2.11 we would conjecture that $\mathcal{C}(\mathcal{L}_1) \simeq \mathcal{C}(\mathcal{L}_2)$ holds (for the right definition of $\mathcal{C}(\mathcal{L})$!) The "equivalence" of the evaluation mechanisms, then, would be expressed by saying that they generate the same syntactic category. (Again this is analogous to categorical logic, where different presentations of a theory give rise to the same classifying model.) Furthermore, universal interpretations of $\mathcal{L}$ correspond to *full* functors $\mathcal{C}(\mathcal{L}) \to \mathcal{D}$, and so Theorem 6.4.4(ii) would also drop out as a corollary.

In general we would not expect the interpretation of $\mathcal{L}$ in $\mathcal{C}(\mathcal{L})$ to be fully abstract, since $\mathcal{C}(\mathcal{L})$ would not be well-pointed. However, we would expect that $\mathcal{C}(\mathcal{L})$ could be extensionally collapsed to a well-pointed PCF-model $\mathcal{C}_0(\mathcal{L})$ that furnished a fully abstract interpretation of $\mathcal{L}$. Thus Theorem 6.4.4(i) would emerge as a corollary of the fact that $\mathcal{C}_0(\mathcal{L}_1) \simeq \mathcal{C}_0(\mathcal{L}_2)$.

Chapter 7

Examples of realizability models

This chapter continues our investigation of interpretations of PCF-like languages in realizability models. In Chapter 6 we developed a general categorical theory of such interpretations; in this chapter we focus on some specific examples and applications of the theory. We have already seen in Section 5.3 many concrete examples of PCF-models of the form $\mathbf{Mod}(A)$; the emphasis in the present chapter is on special properties of particular models. Specifically, our interest is in finding models that provide fully abstract or universal interpretations of particular programming languages.

In obtaining such results as these, it turns out that we can save a great deal of work by showing how certain classical categories of domains can be embedded in realizability models. Using these embeddings, known full abstraction and universality results for the classical categories can be transferred directly to the realizability models. We begin in Section 7.1 by reviewing some standard results from classical domain theory—the material here is culled from [79,80]. In Section 7.2 we show that the category of *effective domains* can be fully embedded in $\mathbf{Mod}(K_1)$, and hence that $\mathbf{Mod}(K_1)$ provides a universal model for PCF^{++}. The results here may be seen as reformulations of results of McCarty (see [60,59]). In Section 7.3 we obtain similar results for $\mathbf{Mod}(\mathcal{P}\omega_{re})$, and discover a sense in which this is mathematically a "better" model than $\mathbf{Mod}(K_1)$. We also show that the classical category of Scott domains (without effective structure) can be fully embedded in $\mathbf{Mod}(\mathcal{P}\omega)$, illustrating how one version of classical domain theory may be recovered as an instance of synthetic domain theory. The results in this section appear to be new.

Section 7.4 is independent of the other three—here we consider models of the form $\mathbf{Mod}(\Lambda^0/T)$. We conjecture that these provide universal (and hence fully abstract) models of sequential PCF, although we have been unable to prove this. However, we obtain some partial results in this direction; we also discuss a strategy for proving the conjecture and some of the difficulties that arise.

7.1 Review of classical domain theory

In this section we recall the standard material from classical domain theory that we will need. We define the category **Dom** of *(Scott) domains* and its subcategory **EDom** of *effective domains*, and discuss the interpretations of PCF-like languages in these categories. Readers familiar with Plotkin's paper [79] may skim this section.

7.1.1 Domains and continuous maps

We first introduce the classical category **Dom** of domains and continuous maps. It is this category that is implicitly used in [79] for the denotational semantics of PCF. We start with some basic notions:

Definition 7.1.1 (DCPO) *Suppose $(D, \sqsubseteq)$ is a partially ordered set.*

(i) A subset $S \subseteq D$ is directed *if $S \neq \emptyset$ and for all $x, y \in S$ there exists $z \in S$ such that $x \sqsubseteq z$, $y \sqsubseteq z$.*

(ii) We say $(D, \sqsubseteq)$ is a (pointed) directed-complete partial order (DCPO) *if D has a least element $\perp$ and every directed subset $S \subseteq D$ has a least upper bound in D (denoted $\bigsqcup S$).*

(iii) A function $f : D \to E$ between DCPOs is a continuous map *if for all directed $S \subseteq D$ we have $f(\bigsqcup S) = \bigsqcup\{f(x) \mid x \in S\}$.*

We write $x \sqcup y$ for $\bigsqcup\{x, y\}$ when this exists; we also write $x \uparrow y$ to mean that x, y are bounded above (i.e. there exists $z \in D$ such that $x \sqsubseteq z$, $y \sqsubseteq z$). The above definition gives us the category **DCPO** of DCPOs and continuous maps. This is already a good enough category for many purposes; however, in order to address questions of full abstraction and definability it is convenient to consider a somewhat smaller category. For this we require a few more definitions.

Definition 7.1.2 (Finite element) *Suppose $(D, \sqsubseteq)$ is a DCPO. An element $d \in D$ is* finite *if whenever $S \subseteq D$ is directed and $d \sqsubseteq \bigsqcup S$ we have $d \sqsubseteq x$ for some $x \in S$. For each $x \in D$ we write F_x for the set of finite elements $d \sqsubseteq x$ in D.*

The finite elements of D are elsewhere called the *compact* or *isolated* elements. One intuition (loosely) is that they are the elements that give only a "finite amount of information". It is easy to see that if x, y are finite then $x \sqcup y$ is finite if it exists. We now consider DCPOs for which the finite elements form a "basis":

Definition 7.1.3 (Algebraic DCPO) *A DCPO D is called* algebraic *if for all $x \in D$ the set F_x is directed and $\bigsqcup F_x = x$. We say D is ω-*algebraic *if D is algebraic and has only countably many finite elements altogether.*

It is not hard to see that an algebraic DCPO is determined up to isomorphism by the poset of its finite elements. Moreover, a continuous map between algebraic DCPOs is completely determined by what it does to the finite elements. The countability condition will come into its own in the next paragraph, when we will consider "effective" analogues of the notions introduced here.

There is one further condition on DCPOs that we need:

Definition 7.1.4 (Consistent completeness) *A DCPO D is* consistently complete *if whenever $x, y \in D$ and $x \uparrow y$ then $x \sqcup y$ exists.*

If D is consistently complete and $d_1, \ldots, d_n$ are finite elements of D with an upper bound in D, clearly $\bigsqcup\{d_1, \ldots, d_n\}$ exists and is a finite element. (Thus the set F_x is automatically directed for any $x \in D$.) We now put all the above definitions together:

Definition 7.1.5 *A* Scott domain *is a consistently complete, ω-algebraic DCPO. We write* **Dom** *for the category of Scott domains and continuous maps.*

Henceforth we shall refer to Scott domains simply as *domains*. The following proposition gives us what we need to know about the structure of **Dom**. We give a sketch of the proof, since explicit descriptions of the structure will be useful later.

Proposition 7.1.6 *(i) The DCPO $N_\perp$ (the lifted natural numbers) is a domain;*

(ii) The category **Dom** *is cartesian-closed;*

(iii) For every domain D there is a continuous map $\mathit{fix}_D : D^D \to D$ such that for any $f \in D^D$ the element $\mathit{fix}_D(f)$ is the least fixed point of the map f.

PROOF (Sketch.) (i) is trivial.

(ii) The terminal object is the one-element domain. The product of domains D_1, D_2 is just their product as posets; The finite elements of $D_1 \times D_2$ are precisely the elements of the form (d_1, d_2), where d_i is a finite element of D_i. One can easily check that $D_1 \times D_2$ is indeed a domain, that the projections are continuous maps and that a pairing of continuous maps is a continuous map. For the exponential E^D of domains, take the set of continuous maps $D \to E$ equipped with the pointwise ordering. Given finite elements d, e of D, E respectively, let us write $d \Rightarrow e$ for the continuous map $D \to E$ given by

$$(d \Rightarrow e)(x) = \begin{cases} e & \text{if } d \sqsubseteq x \\ \perp & \text{otherwise.} \end{cases}$$

It can be shown that the finite elements of E^D are precisely the existing finite lubs of elements of the form $d \Rightarrow e$ where d, e are finite, and hence that E^D is indeed a domain. (The details are given in Section 4 of [79].) It is easy to check that the evaluation map $E^D \times D \to E$ is continuous, and that currying a continuous map yields a continuous map.

(iii) Define $\mathit{fix}_D : D^D \to D$ by

$$\mathit{fix}_D(f) \;=\; \bigsqcup\{f^k(\perp) \mid k \geq 0\}.$$

It is easy to check that $\mathit{fix}_D(f)$ is the least fixed point of f, and that the map fix_D is continuous. □

7.1.2 Effective domains

The category **Dom** gives us a clean mathematical setting for denotational semantics, in which the topological idea of continuity serves as a substitute for computability. However, for some purposes we wish to know what maps between domains are actually computable, not merely continuous. One way to make precise the idea of a computable map is to dirty our hands by adding in some "effective" structure to **Dom**. The key observation is that since continuous maps between domains are determined by their action on finite elements, it suffices to supply effective enumerations of the finite elements of each domain.

Definition 7.1.7 (Effective domain) *(i) An* enumerated domain *consists of a domain D together with a set-theoretic function $\nu : \mathrm{N} \to D$ whose image consists of precisely the finite elements of D.*

(ii) An effective domain *is an enumerated domain (D, ν) such that the relations $\nu(m) \uparrow \nu(n)$ and $\nu(m) = \nu(n) \sqcup \nu(p)$ are decidable (that is, recursive in m, n, p).*

(iii) A computable map *between effective domains (D, ν) and (E, ν') is a continuous map $f : D \to E$ such that the relation $\nu'(n) \sqsubseteq f(\nu(m))$ is semi-decidable (that is, r.e. in m, n). We write* **EDom** *for the category of effective domains and computable maps.*

It is easy to see that in an effective domain (D, ν) the relation $\nu(m) \sqsubseteq \nu(n)$ is decidable, and that there is a binary partial recursive function φ such that $\nu(m) \sqcup \nu(n) = \nu(\varphi(m, n))$ whenever $\nu(m) \uparrow \nu(n)$. We will sometimes abuse language and say "D is an effective domain" omitting mention of the enumeration. The following notion is closely related to the idea of computable map:

Definition 7.1.8 (R.e. element) *Given an element x of an effective domain (D, ν), define the* support *of x to be the set* $\mathrm{supp}(x) = \{n \mid \nu(n) \sqsubseteq x\}$. *We say x is an* r.e. element *of D if $\mathrm{supp}(x)$ is r.e. We write D_{re} for the set of r.e. elements of D.*

We think of the r.e. elements as the "computable" elements of an effective domain. It is easy to see that a map $1 \to D$ is computable iff the corresponding element of D is r.e., and hence that a continuous map $D \to E$ is computable iff the corresponding element of E^D is r.e.

We write $U : \mathbf{EDom} \to \mathbf{Dom}$ for the obvious "forgetful" functor—note that U is trivially faithful. We now show that **EDom** has all the structure described in Proposition 7.1.6 for **Dom**, and that U preserves this structure. Again we give a sketch of the proof—for more details see Chapter 7 of [80].

Proposition 7.1.9 *(i) The domain $N_\perp$ can be equipped with a (standard) effective enumeration ν_0;*

(ii) The category **EDom** *is cartesian-closed, and its cartesian-closed structure is inherited from* **Dom***;*

(iii) For any effective domain (D, ν), the continuous map $\mathit{fix}_D : D \to D$ is computable.

Proof (Sketch.) (i) Note that all the elements of $N_\perp$ are finite. We take the standard enumeration ν_0 to be given by $\nu_0(0) = \perp$, $\nu_0(n) = n + 1$; the enumerated domain $(N_\perp, \nu)$ is then clearly effective.

(ii) The terminal object of **Dom** can trivially be made into an effective domain, and for any effective domain D the unique map $D \to 1$ is computable. For products, suppose (D_1, ν_1) and (D_2, ν_2) are effective domains. Take some recursive and surjective pairing function $(-;-) : \mathrm{N} \times \mathrm{N} \to \mathrm{N}$, and let $\pi_1, \pi_2 : \mathrm{N} \to \mathrm{N}$ be the corresponding (recursive) projections. Define an enumeration ν of the finite elements of $D_1 \times D_2$ by $\nu((n_1; n_2)) = (\nu_1(n_1), \nu_2(n_2))$. It is routine to check that this enumeration is effective, that the projection maps are computable, and that a pairing of computable maps is a computable map.

For exponentials, suppose (D, ν) and (E, ν') are effective domains. Let $[\ldots]$ be some recursive and surjective coding function for finite subsets of N (cf. Section 3.3), and define an enumeration ρ of the finite elements of E^D by

$$\rho([(m_1; n_1), \ldots, (m_j; n_j)]) = \begin{cases} \bigsqcup_{1 \le i \le j} \nu(m_i) \Rightarrow \nu'(n_i) & \text{if this exists} \\ \perp & \text{otherwise.} \end{cases}$$

It can be shown that given $[(m_1; n_1), \ldots, (m_j; n_j)]$ it is decidable whether the lub $\bigsqcup_{1 \le i \le j} \nu(m_i) \Rightarrow \nu'(n_i)$ exists; hence we can check that ρ is an effective enumeration.

Moreover, it can be checked that the evaluation map $E^D \times D \to E$ is computable, and that currying a computable map yields a computable map.

(iii) Suppose $f : D \to D$ is any computable map and e is a recursive index for the set $\{(m; n) \mid \nu(n) \sqsubseteq f(\nu(m))\}$. One can check that the set $I_k = \{n \mid \nu(n) \sqsubseteq f^k(\bot)\}$ is r.e. uniformly in e and k. Thus the set $\mathrm{supp}(\mathit{fix}_D(f)) = \bigcup I_k$ is r.e. uniformly in e, and so the map fix_D is computable. $\Box$

7.1.3 Interpretation of PCF^{++}

Although the categories **Dom** and **EDom** are not PCF-models in the sense of Chapter 6, they can be used to give interpretations of the language PCF_N^{++}. In this paragraph we describe these interpretations and summarize the results obtained by Plotkin in [79].

We will write $[\![-]\!]$ for the interpretation of PCF_N^{++} in **EDom**, and $[\![-]\!]'$ for its interpretation in **Dom**. The interpretations of PCF types are given by

$$\begin{aligned} [\![\iota]\!] &= (N_\bot, \nu_0), & [\![\sigma \to \tau]\!] &= [\![\tau]\!]^{[\![\sigma]\!]}, \\ [\![\iota]\!]' &= N_\bot, & [\![\sigma \to \tau]\!]' &= [\![\tau]\!]'^{[\![\sigma]\!]'} \end{aligned}$$

where ν_0 is the standard enumeration for $N_\bot$ given above. Since U preserves exponentials, these are related by $[\![\sigma]\!]' = U([\![\sigma]\!])$. Given an environment $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \rangle$, we define

$$[\![\Gamma]\!] = [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!], \qquad [\![\Gamma]\!]' = [\![\sigma_1]\!]' \times \cdots \times [\![\sigma_n]\!]'.$$

If $M : \sigma$ is a term in environment Γ, its interpretation $[\![M]\!]^\Gamma$ will be a morphism $[\![\Gamma]\!] \to [\![\sigma]\!]$ in **EDom**. Likewise, the interpretation $[\![M]\!]'^\Gamma$ will be a morphism $[\![\Gamma]\!]' \to [\![\sigma]\!]'$ in **Dom**.

In order to define $[\![M]\!]^\Gamma$, we first define continuous maps

$$\begin{aligned} k_{n,D} &: \ D \to N_\bot \qquad (n \in \mathrm{N}) \\ \mathit{succ}, \mathit{pred} &: \ N_\bot \to N_\bot \\ \mathit{cond}_D &: \ N_\bot \times D \times D \to D \\ \mathit{por} &: \ N_\bot \times N_\bot \to N_\bot \end{aligned}$$

in the obvious way (cf. Paragraph 6.2.1). It is easy to check that these maps are computable, and so they may be regarded as morphisms in either **Dom** or **EDom**. We also define a map $exists : (N_\perp^{N_\perp} \to N_\perp) \to N_\perp$ by

$$exists(F) \;=\; \begin{cases} 0 & \text{if } F(k_{n,N_\perp}) = 0 \text{ for some } n \\ 1 & \text{if } F(\perp) = m+1 \text{ for some } m \\ \perp & \text{otherwise.} \end{cases}$$

Again it is easy to check that this map is computable.

We can now give the definition of $[\![M]\!]^\Gamma$ by induction on the structure of M. The definition is formally almost identical to the one appearing in Section 6.2:

$$\begin{array}{rcl}
[\![x_i^{\sigma_i}]\!]^\Gamma & = & \pi_i \\
[\![n]\!]^\Gamma & = & k_{n[\![\Gamma]\!]} \\
[\![\mathsf{succ}\; M]\!]^\Gamma & = & succ \circ [\![M]\!]^\Gamma \\
[\![\mathsf{pred}\; M]\!]^\Gamma & = & pred \circ [\![M]\!]^\Gamma \\
[\![\mathsf{cond}\; MNP]\!]^\Gamma & = & cond_{[\![\sigma]\!]} \circ \langle [\![M]\!]^\Gamma, [\![N]\!]^\Gamma, [\![P]\!]^\Gamma \rangle \\
[\![\mu x^\sigma . M]\!]^\Gamma & = & fix_{[\![\sigma]\!]} \circ curry_{[\![\sigma]\!]}([\![M]\!]^{\Gamma, x^\sigma}) \\
[\![\lambda x^\sigma . M]\!]^\Gamma & = & curry_{[\![\sigma]\!]}([\![M]\!]^{\Gamma, x^\sigma}) \\
[\![MN]\!]^\Gamma & = & eval \circ \langle [\![M]\!]^\Gamma, [\![N]\!]^\Gamma \rangle \\
[\![\mathsf{por}\; MN]\!]^\Gamma & = & por \circ \langle [\![M]\!]^\Gamma, [\![N]\!]^\Gamma \rangle \\
[\![\mathsf{exists}\; M]\!]^\Gamma & = & exists \circ [\![M]\!]^\Gamma
\end{array}$$

The definition of $[\![M]\!]'^\Gamma$ is given in exactly the same way. Since U preserves all the structure that we have used in giving the definition, we have $[\![M]\!]'^\Gamma = U([\![M]\!]^\Gamma)$.

The following properties of these interpretations were proved by Plotkin in [79]. Theorem 7.1.12 was also discovered independently by Sazonov [86]. From our point of view these results are extremely useful as they allow us to bypass a great deal of technical work involving PCF_N^{++}.

Theorem 7.1.10 (Adequacy for PCF^{++}) *Suppose $M : \iota$ is a closed PCF_N^{++} term. Then $M \Downarrow n$ iff $[\![M]\!] = k_{n,1}$ iff $[\![M]\!]' = k_{n,1}$. Thus $[\![-]\!]$, $[\![-]\!]'$ are adequate interpretations of PCF_N^{++}.*

Theorem 7.1.11 (Full abstraction for PCF$^+$) *Suppose $M, N : \sigma$ are closed terms of* PCF_N^+*. Then $M \approx_{\mathrm{PCF}_N^+} N$ iff $[\![M]\!] = [\![N]\!]$ iff $[\![M]\!]' = [\![N]\!]'$. Thus $[\![-]\!]$, $[\![-]\!]'$ give fully abstract interpretations of* PCF_N^{++}.

Theorem 7.1.12 (Universality for PCF^{++}) *Suppose x is an element of the effective domain $[\![\sigma]\!]$. Then the corresponding map $1 \to [\![\sigma]\!]$ is denotable by a closed term $M : \sigma$ of* PCF_N^{++} *iff x is an r.e. element. Hence the interpretation $[\![-]\!]$ of* PCF_N^{++} *in* **EDom** *is universal.*

Remark 7.1.13 We should point out that our language PCF_N^{++} differs from the language considered by Plotkin in a few technical respects. Perhaps most significantly, his language has an extra ground type o corresponding to the booleans, and the types for cond and exists are modified accordingly. He also incorporates a “zero testing” operation; the conditional operator is provided only for ground types; and in place of our por he has a “parallel conditional” operation. There are other minor differences: for instance, in his formulation the expression $\mathsf{pred}\ 0$ diverges. However, the reader familiar with [79] will readily agree that these differences do not matter for the purposes of the above theorems, and only trivial modifications of Plotkin's proofs are needed to yield these results for our language.

7.2 The model Mod(K_1)

In this section we show that our category **EDom** admits a full, faithful and cartesian-closed embedding into **Mod**(K_1), the usual category of PERs. Combining this fact with the above results of Plotkin, we are able to obtain various universality and full abstraction results for **Mod**(K_1). Our embedding is essentially the same as the one discovered by McCarty [60,59]. However, McCarty's setting is superficially different from ours: he considers the category of *effectively given information systems* and shows that it embeds fully in a model of intuitionistic ZF set theory based on Kleene realizability. In order to keep our exposition self-contained, therefore, we include our own account of this material adapted to our setting. First we introduce a useful piece of notation:

Definition 7.2.1 *If* $R \subseteq \mathrm{N}$ *is an r.e. set, we write* $\mathrm{ind}(R)$ *for the set of* (recursive) indices *for* R*, i.e. the set*

$$\mathrm{ind}(R) \;=\; \{e \mid \mathrm{Im}\,\{e\} = R\}$$

(where $\{e\}$ *denotes the partial recursive function coded by* e*).*

Remark 7.2.2 Alternatively we could have taken the set of indices for R to be $\mathrm{ind}'(R) = \{e \mid \mathrm{Dom}\,\{e\} = R\}$. It is easy to see that $\mathrm{ind}(R)$ and $\mathrm{ind}'(R)$ are "recursively equivalent", i.e. there exist $b, c \in K_1$ such that $x \in \mathrm{ind}(R) \Rightarrow \{b\}(x) \in \mathrm{ind}'(R)$ and $y \in \mathrm{ind}'(R) \Rightarrow \{c\}(y) \in \mathrm{ind}(R)$.

7.2.1 Embedding EDom in Mod(K_1)

We now define our embedding $G : \mathbf{EDom} \to \mathbf{Mod}(K_1)$ as follows:

Definition 7.2.3 *If* D *is an effective domain with enumeration* ν*, let* GD *be the modest set over* K_1 *defined by*

$$|GD| \;=\; D_{re}, \qquad \|x \in GD\| \;=\; \mathrm{ind}(\mathrm{supp}(x)).$$

If $f : D \to E$ *is a computable map between effective domains, let* $Gf : |GD| \to |GE|$ *be the restriction of* f *to* D_{re}*.*

There are several things to check in connection with this definition. First, note that if $x \in D_{re}$ then the set $\mathrm{supp}(x)$ is r.e. by definition, so $\|d \in GD\|$ is non-empty. Secondly, if x, y are distinct elements of D_{re} then the sets $\mathrm{supp}(x)$, $\mathrm{supp}(y)$ are different, so $\|x \in GD\| \cap \|y \in GD\| = \emptyset$; hence the assembly GD is indeed a modest set. Thirdly, if x is r.e. and f is computable then it is easy to see that $f(d)$ is r.e.; hence f does indeed restrict to a function $D_{re} \to E_{re}$. Thus the above definition is sound. Furthermore, we have:

Proposition 7.2.4 *The above defines a functor* $G : \mathbf{EDom} \to \mathbf{Mod}(K_1)$*.*

PROOF Given $f : D \to E$ as above, we need to check that the function Gf has a realizer. So suppose D, E are equipped with enumerations ν, ν' respectively, e

is a recursive index for the set $\{(m;n) \mid \nu'(n) \sqsubseteq f(\nu(m))\}$, and x is an element of D_{re}. Since each $\nu'(n)$ is finite, we have $\nu'(n) \sqsubseteq f(x)$ iff $\nu'(n) \sqsubseteq f(\nu(m))$ for some $m \in \mathrm{supp}(x)$. Thus

$$\mathrm{supp}(f(x)) \;=\; \bigcup_{\nu(m)\sqsubseteq x} \mathrm{supp}(f(\nu(m))).$$

Hence if r is an index for $\mathrm{supp}(x)$, it is easy to see that $\lambda^* z.\, k\ (p_2(ez))\ ((br)(p_1(ez)))$ is an index for $\mathrm{supp}(f(x))$, where b is as in Remark 7.2.2 and p_1, p_2 are codes for the projections π_1, π_2. Thus Gf is tracked by $\lambda^* rz.\, k\ (p_2(ez))\ ((br)(p_1(ez)))$. Finally it is trivial that G respects identities and composition, so G is a functor. $\Box$

In the next few propositions we establish some of the categorical properties of G: that it is full and faithful, and (roughly speaking) preserves all the structure given by Proposition 7.1.9.

Proposition 7.2.5 *The functor G is faithful.*

Proof Suppose $Gf = Gg$ for $f, g : D \rightrightarrows E$. Then f, g agree on all r.e. elements of D, and so in particular on all finite elements. Thus for a general element $x \in D$ we have $f(x) = \bigsqcup f(F_x) = \bigsqcup g(F_x) = g(x)$, and so $f = g$. $\Box$

Proposition 7.2.6 *G preserves finite products.*

Proof Clearly $G(1) \cong 1$. For binary products, note that if (D_1, ν_1) and (D_2, ν_2) are effective domains then according to our definitions we have

$$\begin{aligned}
|G(D_1 \times D_2)| \;&=\; |GD_1 \times GD_2| \;=\; D_{1re} \times D_{2re},\\
\|(x_1,x_2) \in G(D_1 \times D_2)\| \;&=\; \mathrm{ind}\{(n_1;n_2) \mid\ n_i \in \mathrm{supp}(x_i)\},\\
\|(x_1,x_2) \in GD_1 \times GD_2\| \;&=\; \{\langle e_1, e_2\rangle \mid\ e_i \in \mathrm{ind}(\mathrm{supp}(x_i))\}.
\end{aligned}$$

So the canonical morphism $G(D_1 \times D_2) \to GD_1 \times GD_2$ is tracked by the element $\lambda^* r.\, \langle \lambda^* z.p_1(rz),\, \lambda^* z.p_2(rz)\rangle$, and its inverse by $\lambda^* rz.\, q\,((\mathit{fst}\ r)(p_1 z))\,((\mathit{snd}\ r)(p_2 z))$ where q codes the pairing function $(-;-)$. Thus $G(D_1 \times D_2) \cong GD_1 \times GD_2$, via an isomorphism that identifies the projections. $\Box$

Proposition 7.2.7 *The object $G(N_\perp, \nu_0)$ is canonically isomorphic to $N_\perp$, where $\perp$ is the lift functor on* $\mathbf{Mod}(K_1)$ *arising from the r.e. subobject classifier Σ.*

Proof Let $N'_\perp = G(N_\perp, \nu_0)$. From our definitions and the description of the lift functor in Proposition 4.2.4, we see that

$$\begin{array}{rclrcl} & & |N_\perp| \;=\; |N'_\perp| \;=\; \mathrm{N} \sqcup \{\perp\}, & & & \\ \|n \in N_\perp\| & = & T \wedge (I \Rightarrow \{n\}), & \|\perp \in N_\perp\| & = & U \wedge \mathrm{N}, \\ \|n \in N'_\perp\| & = & \mathrm{ind}\{0, n+1\}, & \|\perp \in N'_\perp\| & = & \mathrm{ind}\{0\}, \end{array}$$

where $T = \{m \mid \{m\}(0)\downarrow\}$, $U = \{m \mid \{m\}(0)\uparrow\}$ and $I = \{i\}$. So the canonical morphism $N_\perp \to N'_\perp$ is tracked by

$$\lambda^* rm.\, \mathit{if}\,(\mathit{eq}\; m\; 0)\; 0\; (k(\mathit{inc}(\mathit{snd}\, r\; i))(\mathit{fst}\, r\; i)),$$

where eq codes equality testing and inc codes the successor operation. To give a realizer for the inverse, first take $\mathit{search} \in K_1$ such that for all $r \in K_1$ we have

$$\begin{array}{rl} \text{either} & \exists n.\, (rn > 0 \;\wedge\; \mathit{search}\; r = rn) \\ \text{or} & (\neg \exists n.\, rn > 0) \;\wedge\; \mathit{search}\; r \uparrow . \end{array}$$

(Here we exploit the fact that we can compute $r0, r1, \ldots$ "in parallel".) The canonical morphism $N'_\perp \to N_\perp$ is then tracked by

$$\lambda^* r.\, \langle \lambda^* z.\, \mathit{search}\; r,\;\; \lambda^* z.\, \mathit{dec}(\mathit{search}\; r) \rangle,$$

where dec codes the predecessor operation. □

For the remaining properties of G we have to work rather harder. We need the following result from classical recursion theory—note the close kinship with the Myhill-Shepherdson Theorem (see Theorem 5.3.1).

Theorem 7.2.8 (Rice-Shapiro) *Any morphism $\Sigma^N \to \Sigma$ in* $\mathbf{Mod}(K_1)$ *is monotone and continuous (i.e. preserves existing lubs of directed sets) with respect to the pointwise order on $|\Sigma^N|$.*

Proof An easy consequence of the classical Rice-Shapiro Theorem as stated in [17, Chapter 7]. □

Lemma 7.2.9 *Suppose (D,ν) is an effective domain. Then every morphism $\phi : GD \to \Sigma$ is monotone and continuous with respect to the order $\sqsubseteq$ on $|GD|$.*

Proof Let us call a set $R \subseteq \mathrm{N}$ *consistent* if $\{\nu(n) \mid n \in R\}$ has an upper bound in D. Clearly R is consistent iff $\nu(m) \uparrow \nu(n)$ for all $m, n \in R$. Since the relation $\nu(m) \uparrow \nu(n)$ is recursive, it is semi-decidable whether an r.e. set R is inconsistent: more formally, there exists $t \in K_1$ such that for all e we have $te\downarrow$ iff $\mathrm{Im}\ \{e\}$ is inconsistent. Next, if R is consistent, define its *directed closure* to be the set

$$\overline{R} \;=\; \{\, m \mid \nu(m) \sqsubseteq \nu(n_1) \sqcup \ldots \sqcup \nu(n_j),\ n_1, \ldots n_j \in R\}.$$

Since the relation $\nu(m) \sqsubseteq \nu(n)$ is recursive and there is a recursive φ such that $\nu(\varphi(m,n)) = \nu(m) \sqcup \nu(n)$ for all $m, n \in R$, we may effectively obtain $\overline{R}$ from R. More formally, there exists $u \in K_1$ such that whenever $\mathrm{Im}\ \{e\}$ is consistent we have $ue\downarrow$ and $\mathrm{Im}\ \{ue\} = \overline{\mathrm{Im}\ \{e\}}$.

Now suppose $\phi : GD \to \Sigma$ is a morphism tracked by r. Clearly if R is r.e. and consistent then $\overline{R} = \mathrm{supp}(x)$, where $x = \bigsqcup\{\nu(n) \mid n \in \overline{R}\} \in D_{re}$. Thus if $R = \mathrm{Im}\ \{e\}$ then $ue \in \|x \in GD\|$. We will define a morphism $\phi' : \Sigma^N \to \Sigma$ using ϕ. First note that the object Σ^N can be presented thus:

$$|\Sigma^N| \;=\; \{R \subseteq \mathrm{N} \mid R \text{ is r.e.}\}, \qquad \|R \in \Sigma^N\| \;=\; \mathrm{ind}(R).$$

Now define the function $\phi' : |\Sigma^N| \to |\Sigma|$ by

$$\phi'(R) \;=\; \begin{cases} \top & \text{if } R \text{ is inconsistent} \\ \phi(x) & \text{if } R \text{ is consistent and } \overline{R} = \mathrm{supp}(x). \end{cases}$$

Note that ϕ' is tracked by $\lambda^* e.\, p\,\langle \lambda^* w.te,\ r(ue)\rangle$, where p tracks the "parallel convergence" morphism $\Sigma \times \Sigma \to \Sigma$.

To see that ϕ is monotone, suppose $x_1, x_2 \in D_{re}$ with $x_1 \sqsubseteq x_2$. Take $R_i = \mathrm{supp}(x_i)$; then R_1, R_2 are r.e. and $\phi'(R_i) = \phi(x_i)$. Moreover $R_1 \subseteq R_2$, so by the Rice-Shapiro Theorem we have $\phi'(R_1) \leq \phi'(R_2)$ where $\leq$ is the evident order on $|\Sigma|$; thus $\phi(x_1) \leq \phi(x_2)$. To see that ϕ is continuous, suppose $S \subseteq D_{re}$ is directed and x is the least upper bound of S in D_{re}. Then it is easy to see that $x = \bigsqcup S$ in D. Take $R_x = \mathrm{supp}(x)$ and $R_y = \mathrm{supp}(y)$ for each $y \in S$; then $R_x = \bigcup_{y \in S} R_y$. So by the Rice-Shapiro Theorem, $\phi'(R_x) = \top$ iff $\phi'(R_y) = \top$ for some $y \in S$. Thus $\phi(x) = \top$ iff $\phi(y) = \top$ for some $y \in S$. This completes the proof. $\Box$

Lemma 7.2.10 *Suppose (D, ν) is an effective domain.*

(i) The order $\sqsubseteq$ on D_{re} coincides with the Σ-order on GD.

(ii) Suppose $S \subseteq D_{re}$ is directed. Then S has a least upper bound x in D_{re} iff $S \subseteq |GD|$ has a Σ-limit x' in the sense of Definition 5.4.2, and in this case $x = x'$.

PROOF (i) First suppose $x_1 \sqsubseteq x_2$ in D_{re}. Then for any $\phi : GD \to \Sigma$ we have $\phi(x_1) \leq \phi(x_2)$ by the above lemma; thus $x_1 \preceq_\Sigma x_2$. Conversely, if $x_1 \not\sqsubseteq x_2$ then there exists n such that $\nu(n) \sqsubseteq x_1$ but not $\nu(n) \sqsubseteq x_2$. Define $\phi : GD \to \Sigma$ by $\phi(x) = \top$ iff $\nu(n) \sqsubseteq x$. Note that ϕ is tracked by $\lambda zw.(bz)n$, where b is as in Remark 7.2.2. Now $\phi(x_1) = \top$ and $\phi(x_2) = \bot$, thus $x_1 \not\preceq_\Sigma x_2$.

(ii) First suppose S has a least upper bound x in D_{re}. Then for any $\phi : GD \to \Sigma$, by the above lemma we have $\phi(x) = \top$ iff $\phi(y) = \top$ for some $y \in S$; thus x is a Σ-limit of S. Now suppose x' is a Σ-limit of S. Then for all $y \in S$ we have $y \preceq_\Sigma x'$, hence $y \sqsubseteq x'$ by (i); thus $x \sqsubseteq x'$ in D where $x = \bigsqcup S$. If $x \neq x'$, take n such that $\nu(n) \sqsubseteq x'$ but not $\nu(n) \sqsubseteq x$, and as in (i) construct $\phi : GD \to \Sigma$ such that $\phi(z) = \top$ iff $\nu(n) \sqsubseteq z$. Then clearly $\phi(x') = \top$ but $\phi(y) = \bot$ for all $y \in S$, contradicting the assertion that x' is a Σ-limit for S. Hence $x' = x$. $\Box$

Proposition 7.2.11 *The functor G is full.*

PROOF Suppose (D, ν) and (E, ν') are effective domains, and $h : GD \to GE$ is a morphism tracked by r. Then h is monotone with respect to $\preceq_\Sigma$, and hence with respect to $\sqsubseteq$ by Lemma 7.2.10(i). Hence for every $x \in D$ the set $h(F_x)$ is directed. So define a function $f : D \to E$ by $f(x) = \bigsqcup h(F_x)$. First note that f is monotone: if $x \sqsubseteq y$ then $F_x \subseteq F_y$ so $h(F_x) \subseteq h(F_y)$ so $f(x) \sqsubseteq f(y)$. Secondly, f is continuous: if $S \subseteq D$ is directed and $x = \bigsqcup S$ then

$$\begin{aligned} f(x) \;&=\; \bigsqcup h(F_x) \;=\; \bigsqcup h(\textstyle\bigcup_{y \in S} F_y) \;=\; \bigsqcup(\textstyle\bigcup_{y \in S} h(F_y)) \\ &=\; \bigsqcup\{\bigsqcup h(F_y) \mid y \in S\} \;=\; \bigsqcup f(S). \end{aligned}$$

Thirdly, f is computable: since the relation $\nu(m') \sqsubseteq \nu(m)$ is recursive, we may construct $v \in K_1$ such that for each m we have $vm\downarrow$ and $\mathrm{Im}\ \{vm\} = \mathrm{supp}(\nu(m))$.

Hence $vm \in \|\nu(m) \in GD\|$ and so $r(vm) \in \|h(\nu(m)) \in GE\|$. But since $\nu(m)$ is finite we clearly have $h(\nu(m)) = f(\nu(m))$. Hence we have $\nu'(n) \sqsubseteq f(\nu(m))$ iff $b(r(vm))n\downarrow$, where b is as in Remark 7.2.2. Thus the relation $\nu'(n) \sqsubseteq f(\nu(m))$ is r.e. in m, n.

It remains to check that $Gf = h$. Clearly if $d \in D$ is finite then $f(d) = h(d)$. For a general element $x \in D_{re}$, note that $x = \bigsqcup F_x$ and so x is a Σ-limit of F_x by Lemma 7.2.10(ii); thus $h(x)$ is a Σ-limit of $h(F_x)$ and so by Lemma 7.2.10(ii) we have $h(x) = \bigsqcup h(F_x) = \bigsqcup f(F_x) = f(x)$ as required. $\Box$

Proposition 7.2.12 *G preserves exponentials.*

Proof Suppose (D, ν) and (E, ν') are effective domains. Recall that the r.e. elements of E^D are precisely the computable maps $D \to E$; thus since G is full and faithful there is a canonical bijection

$$|G(E^D)| \;=\; (E^D)_{re} \;\cong\; \mathrm{Hom}(GD, GE) \;=\; |GE^{GD}|.$$

If $f : D \to E$ is computable, let graph $f = \{(m; n) \mid \nu'(n) \sqsubseteq f(\nu(m))\}$. We will show that each of the sets $\|f \in G(E^D)\|$, $\|f \in GE^{GD}\|$ is recursively equivalent to ind(graph f) uniformly in f.

Note that $(m; n) \in$ graph f iff $(\nu(m) \Rightarrow \nu'(n)) \sqsubseteq f$ iff $[(m; n)] \in \mathrm{supp}(f)$, since if ρ is the enumeration given in the proof of Proposition 7.1.9 then $\rho([(m; n)]) = (\nu(m) \Rightarrow \nu'(n))$. Take $p, q \in K_1$ such that $pz = [z]$ and $q[z] = z$. If $r \in \|f \in G(E^D)\|$ then clearly $\lambda^* z.p(rz) \in$ ind(graph f), and if $e \in$ ind(graph f) then $\lambda^* z.q(ez) \in \|f \in G(E^D)\|$. So $\|f \in G(E^D)\|$ and ind(graph f) are recursively equivalent.

Next note that if $e \in$ ind(graph f) then from the proof of Proposition 7.2.4 we see that $\lambda^* rz.\, k\ (p_2(ez))\ ((br)(p_1(ez))$ tracks Gf. Conversely, if r tracks Gf then from the proof of Proposition 7.2.11 we see that $c(\lambda^* z.\, b(r(v(p_1 z)))(p_2 z)) \in$ ind(graph f), where b, c are as in Remark 7.2.2 and p_1, p_2 are codes for π_1, π_2. Thus ind(graph f) and $\|f \in GE^{GD}\|$ are recursively equivalent. Hence we see that $G(E^D) \cong GE^{GD}$, via an isomorphism that clearly identifies the evaluation morphisms. $\Box$

Finally we will show that G respects the fixed point operators. First recall from Proposition 5.3.2 that $\mathbf{Mod}(K_1)$ equipped with the r.e. dominance gives a model satisfying the completeness axiom; hence for every useful object X we obtain a morphism fix_X as in Section 5.4. Next note that since $G(N_\perp) \cong N_\perp$ and $G(E^D) \cong GE^{GD}$, for each type σ we have $G(\llbracket \sigma \rrbracket) \cong \llbracket \sigma \rrbracket_N$, where $\llbracket \sigma \rrbracket_N$ is defined as in Paragraph 6.2.1. Thus we may state the following proposition:

Proposition 7.2.13 *For each type σ we have $G(\mathit{fix}_{\llbracket \sigma \rrbracket}) \cong \mathit{fix}_{\llbracket \sigma \rrbracket_N}$ via the canonical isomorphisms $G(\llbracket \sigma \to \sigma \rrbracket) \cong \llbracket \sigma \to \sigma \rrbracket_N$ and $G(\llbracket \sigma \rrbracket) \cong \llbracket \sigma \rrbracket_N$.*

Proof Let $D = \llbracket \sigma \rrbracket$. Since $\mathbf{Mod}(K_1)$ is well-pointed and G is full, it suffices to check that $G(\mathit{fix}(f))$ agrees with $\mathit{fix}(Gf)$ for all $f : D \to D$ in **EDom**. It follows from Lemma 7.2.10 that $(|GD|, \sqsubseteq)$ and $(|\llbracket \sigma \rrbracket_N|, \preceq_\Sigma)$ are isomorphic as posets. Moreover, $G(\mathit{fix}(f))$ corresponds to the least fixed point of f with respect to $\sqsubseteq$; and it is easy to see that $\mathit{fix}(Gf)$ corresponds to the least fixed point of Gf with respect to $\preceq_\Sigma$. Thus the canonical isomorphism identifies $G(\mathit{fix}(f))$ with $\mathit{fix}(Gf)$. $\square$

The above six propositions may be summarized as follows:

Theorem 7.2.14 (McCarty) *The functor G : **EDom** $\to$ $\mathbf{Mod}(K_1)$ is full, faithful and cartesian-closed, and preserves the object $N_\perp$ and the morphisms $\mathit{fix}_{\llbracket \sigma \rrbracket}$ up to isomorphism.* $\square$

7.2.2 Interpreting PCF^{++} in $\mathbf{Mod}(K_1)$

We can now reap the benefits of all this work. First notice that our interpretation of PCF$^{++}_N$ in **EDom** is related via G to the interpretation in $\mathbf{Mod}(K_1)$ given by Chapter 6:

Proposition 7.2.15 *The PCF-model $\mathbf{Mod}(K_1)$ supports* PCF^{++}. *Moreover, for any* PCF$^{++}_N$ *term $M : \sigma$ in environment Γ, the morphism $G\llbracket M \rrbracket^\Gamma : G\llbracket \Gamma \rrbracket \to G\llbracket \sigma \rrbracket$ is canonically isomorphic to the morphism $\llbracket M \rrbracket^\Gamma_N : \llbracket \Gamma \rrbracket_N \to \llbracket \sigma \rrbracket_N$.*

PROOF To see that $\mathbf{Mod}(K_1)$ supports PCF^{++}, just take $por = G([\![\,\mathsf{por}\; xy\,]\!]^{\Gamma})$ (where $\Gamma = \langle x^{\iota}, y^{\iota}\rangle$) and $\exists = G([\![\,\lambda f^{\iota\to\iota}.\,\mathsf{exists}\;(\lambda g^{\iota\to\iota}.f(g\,0))\,]\!])$. Then it is easy to see that por and $\exists$ satisfy the equations given in Propositions 6.2.9 and 6.2.10. (This way of defining por and $\exists$ obviates the need to give explicit realizers for them.) The second clause is straightforward by induction on the structure of M, by comparing the definition of the interpretation in Section 6.2 with that in Paragraph 7.1.3 and noting that G preserves all relevant structure. □

Theorem 7.2.16 *(i) The interpretations of* PCF^+_N, PCF^+_V, PCF^+_L *in* $\mathbf{Mod}(K_1)$ *are fully abstract.*

(ii) The interpretations of PCF$^{++}_N$, PCF$^{++}_V$, PCF$^{++}_L$ *in* $\mathbf{Mod}(K_1)$ *are universal.*

PROOF (i) Suppose $M, N : \sigma$ are closed terms of PCF^+_N with $M \approx N$. Then by Theorem 7.1.11 we have $[\![\,M\,]\!] = [\![\,N\,]\!]$ in $\mathbf{EDom}$. Hence by the above proposition we have $[\![\,M\,]\!]_N = [\![\,N\,]\!]_N$ in $\mathbf{Mod}(K_1)$. So the interpretation of PCF^+_N in $\mathbf{Mod}(K_1)$ is fully abstract. The corresponding results for PCF^+_V and PCF^+_L are now immediate by Theorem 6.4.4(i).

(ii) Let h be any morphism $1 \to [\![\,\sigma\,]\!]_N$ in $\mathbf{Mod}(K_1)$. Since G is full we may take $f : 1 \to [\![\,\sigma\,]\!]$ in $\mathbf{EDom}$ with $Gf = h$. Now by Theorem 7.1.12 there exists a closed term $M : \sigma$ of PCF$^{++}_N$ with $[\![\,M\,]\!] = f$. So by the above proposition we have $[\![\,M\,]\!]_N = h$. Thus the interpretation of PCF$^{++}_N$ in $\mathbf{Mod}(K_1)$ is universal. The corresponding results for PCF$^{++}_V$ and PCF$^{++}_L$ follow by Theorem 6.4.4(ii). □

The second half of this theorem is particularly interesting, as it shows that two quite different styles of "computation at higher types" give rise to the same class of partial functionals. To make this precise, let us define a *(call-by-name) partial type structure (PTS)* to be a family of sets A_σ, one for each type σ, together with bijections $A_\iota \cong \mathrm{N}_\perp$ and $A_{\sigma\to\tau} \cong A'_{\sigma\to\tau} \subseteq [A_\sigma, A_\tau]$, where $[A_\sigma, A_\tau]$ is the set-theoretic function space. Then there are natural "application" functions $A_{\sigma\to\tau} \times A_\sigma \to A_\tau$ for each σ, τ. Moreover, Theorem 7.2.16(ii) immediately yields the following (similar results can also be formulated for PCF$^{++}_V$ and PCF$^{++}_L$.)

Corollary 7.2.17 *The PTS consisting of* PCF$^{++}_N$*-terms modulo $\approx$ is isomorphic to the PTS arising from global elements of the objects $[\![\,\sigma\,]\!]_N$ in* $\mathbf{Mod}(K_1)$. □

Surprisingly, this beautiful result does not seem to have appeared in the literature, although it follows immediately from the known theorems of Plotkin/Sazonov (Theorem 7.1.12) and McCarty (Theorem 7.2.14). Nonetheless, it seems to us to be even more significant from a computational point of view than either of these two results individually. The language PCF_N^{++} represents a very *extensional* style of computation in which functionals are treated as "oracles"—information about a functional can only be obtained by testing it on some argument. By contrast, the functionals arising from global elements in $\mathbf{Mod}(K_1)$ represent a highly *intensional* style—computations on functionals really work on their machine-level representation as natural numbers. A functional of type $\sigma \to \tau$ is thus given by a recursive index that simply "happens" to behave in an extensional way on codes for functionals of type σ, and *a priori* it is far from obvious that there is any "extensional" programming language in which all such functionals may be expressed. The above result thus has much the same flavour as the Myhill-Shepherdson Theorem.

The functionals arising from $\mathbf{Mod}(K_1)$ may be termed the "hereditarily partial effective operations" (HPEOs), since they are the natural "partial" analogue of the hereditarily effective operations (see e.g. [51,99,38]). In view of Church's Thesis, this PTS arguably has a certain philosophical significance in that it captures the most generous possible notion of "effectively computable functional". There are in fact many other characterizations of this PTS, contributing to the impression that it is in some sense fundamental. We conclude this paragraph by mentioning some of these, and other related work in the area.

In [20] Ershov defined the "effective finite-type functionals" to be the effectively given elements in the category of f_0-*spaces*, a category very similar to our **Dom**. As early as 1972 Ershov stated [21] that the category of effective f_0-spaces (essentially our **EDom**) could be fully embedded in the category **EN** of *enumerated sets* (a full subcategory of $\mathbf{Mod}(K_1)$). This result is sometimes known as the Generalized Myhill-Shepherdson Theorem (for a proof see e.g. [31].) Ershov could thus define the "partial computable functionals" to be the partial functionals existing in **EN**, and showed that they coincided with the effective finite-type functionals. This closely foreshadows McCarty's result above. However, this definition of the partial computable functionals is arguably less natural than that of the HPEOs, since it is not immediately obvious that **EN** has the required exponentials.

The category **EN** has poor closure properties in comparison with $\mathbf{Mod}(K_1)$. However, Mulry [64] showed that **EN** could be nicely embedded in the *recursive topos*—hence the above PTS may also be characterized as the one arising from this topos. Later, Longo and Moggi [54,62] gave another good category extending **EN**, the category of *generalized enumerated sets*.

Finally we mention the Hereditarily Partial Effective Functionals (HPEFs) introduced by Longo (see e.g. [55]). These can be defined either for the integer types or for the pure types—the definition is very ingenious, making use of an intermediate type $n.5$ in order to define type $n+1$ from type n. As is shown in [55], the HPEFs correspond precisely to the effective elements of certain f_0-spaces. It follows that, if one works with the pure type hierarchy, one obtains essentially the call-by-value type structure arising from $\mathbf{Mod}(K_1)$. One disadvantage of the HPEF definition is that again it is not immediate that it is well-defined: the definition makes use of certain "pairing" operations at higher types, and it is non-trivial to show that these exist.

7.3 The models $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$

In the last section we defined an embedding $\mathbf{EDom} \to \mathbf{Mod}(K_1)$, and used it to obtain full abstraction and universality results. In this section we carry out a precisely similar programme for the PCA $\mathcal{P}\omega_{re}$ (see Section 3.3): we construct a full embedding $I : \mathbf{EDom} \to \mathbf{Mod}(\mathcal{P}\omega_{re})$, and hence show that $\mathbf{Mod}(\mathcal{P}\omega_{re})$ provides a fully abstract model for PCF$^+$, and a universal model for PCF^{++}. Thus we obtain yet another characterization of the HPEOs as the partial type structure arising from $\mathbf{Mod}(\mathcal{P}\omega_{re})$. Somewhat incidentally, we also obtain a full embedding $H : \mathbf{Dom} \to \mathbf{Mod}(\mathcal{P}\omega)$, showing that $\mathbf{Mod}(\mathcal{P}\omega)$ provides a fully abstract model for PCF$^+$. (The line of investigation here follows a suggestion by Alex Simpson.)

Although these results hold less philosophical interest than those for K_1, mathematically they seem more elegant and their proofs are simpler. Moreover, as a model for PCF^{++}, $\mathbf{Mod}(\mathcal{P}\omega_{re})$ possesses some good mathematical properties not shared by $\mathbf{Mod}(K_1)$. In particular, we will see that all the objects corresponding to PCF types are *projective* (see Section 2.4).

For technical reasons, we will in fact work in this section with the category **NDom** of *enumerated domains* and continuous maps (see Definition 7.1.7(i)) rather than **Dom**. However, it is easy to see that $\mathbf{NDom} \simeq \mathbf{Dom}$, either by invoking the axiom of choice or by understanding the countability condition in Definition 7.1.3 in a constructive sense. If (D, ν) is an enumerated domain and $x \in D$, we write $\mathrm{supp}(x) = \{n \mid \nu(n) \sqsubseteq x\}$ (note that this extends Definition 7.1.8 for effective domains). We sometimes write $|D|$ for the underlying set of D.

Throughout this section we take $(-;-)$ and $[-,\ldots,-]$ to be fixed recursive and surjective codings for pairs and finite sets of natural numbers, and take $\mathcal{P}\omega, \mathcal{P}\omega_{re}$ to be the PCAs with the corresponding application operation (see Definition 3.3.1).

7.3.1 Embedding EDom in $\mathbf{Mod}(\mathcal{P}\omega_{re})$

We define embeddings $H : \mathbf{NDom} \to \mathbf{Mod}(\mathcal{P}\omega)$ and $I : \mathbf{EDom} \to \mathbf{Mod}(\mathcal{P}\omega_{re})$ as follows.

Definition 7.3.1 *(i) If D is an enumerated domain, let HD be the modest set over $\mathcal{P}\omega$ defined by $|HD| = |D|$, $\|x \in HD\| = \{\,\mathrm{supp}(x)\,\}$. If $f : D \to E$ is a continuous map between enumerated domains, let $Hf = f : |HD| \to |HE|$.*

(ii) If D is an effective domain, let ID be the modest set over $\mathcal{P}\omega_{re}$ defined by $|ID| = D_{re}$, $\|x \in ID\| = \{\,\mathrm{supp}(x)\,\}$. If $f : D \to E$ is a computable map between effective domains, let $If : |ID| \to |IE|$ be the restriction of f to D_{re}.

Note that if x, y are distinct elements of D then $\mathrm{supp}(x) \neq \mathrm{supp}(y)$, so HD is indeed a modest set. Also if $x \in D_{re}$ then $\mathrm{supp}(x)$ is r.e. so ID is a modest set over $\mathcal{P}\omega_{re}$. Moreover if f is computable then it restricts to a function $D_{re} \to E_{re}$.

Proposition 7.3.2 *The above defines functors $H : \mathbf{NDom} \to \mathbf{Mod}(\mathcal{P}\omega)$ and $I : \mathbf{EDom} \to \mathbf{Mod}(\mathcal{P}\omega_{re})$.*

PROOF Suppose $(D, \nu), (E, \nu')$ are enumerated domains and $f : D \to E$ is continuous. Let $A = \{([m]; n) \mid \nu'(n) \sqsubseteq f(\nu(m))\}$. Then A tracks Hf, since given $x \in D$ and $B \in \|x\|$ we have $B = \{m \mid \nu(m) \sqsubseteq x\}$ and so

$$A \cdot B \;=\; \bigcup_{\nu(m) \sqsubseteq x} \{n \mid \nu'(n) \sqsubseteq f(\nu(m))\} \;=\; \{n \mid \nu'(n) \sqsubseteq f(x)\} \;\in\; \|f(x)\|.$$

Thus Hf is indeed a morphism in $\mathbf{Mod}(\mathcal{P}\omega)$. Furthermore, if D, E are effective domains and f is computable then clearly A is r.e. and A tracks If; thus If is a morphism in $\mathbf{Mod}(\mathcal{P}\omega_{re})$. The fact that H, I preserve identities and composition is trivial. $\Box$

Note at once that for any D the objects HD, ID have the following property: for any element x, the set $\|x\|$ consists of just one realizer. We will return to this point in the next paragraph. Meanwhile we establish some of the preservation properties of H, I. The proofs here, unlike some of those in Paragraph 7.2.1, are entirely elementary.

Proposition 7.3.3 *The functors H, I are full and faithful.*

PROOF It is trivial that H is faithful, and the proof of Proposition 7.2.5 shows that I is faithful. To see that H is full, suppose $h : HD \to HE$ is a morphism tracked by $A \in \mathcal{P}\omega$. Define $f = h : |D| \to |E|$. Then f is monotone: if $x \sqsubseteq y \in D$ and $B \in \|x\|$, $C \in \|y\|$ then $B \subseteq C$ so $A \cdot B \subseteq A \cdot C$; hence $h(x) \sqsubseteq h(y)$ since $h(x) = \bigsqcup\{\nu'(n) \mid n \in A \cdot B\}$ and similarly for y. Also f is continuous: if $x = \bigsqcup S$, $B_x \in \|x\|$ and $B_y \in \|y\|$ for each $y \in S$ then $B_x = \bigcup_{y \in S} B_y$; hence $A \cdot B_x = \bigcup A \cdot B_y$ and so

$$h(x) \;=\; \bigsqcup \nu'(A \cdot B_x) \;=\; \bigsqcup_{y \in S} (\bigsqcup \nu'(A \cdot B_y)) \;=\; \bigsqcup_{y \in S} h(y).$$

Thus f is a morphism in $\mathbf{NDom}$, and trivially $Hf = h$.

To see that I is full, suppose $h : ID \to IE$ is a morphism tracked by $A \in \mathcal{P}\omega_{re}$. Define $f : |D| \to |E|$ by $f(x) = \bigsqcup h(F_x)$; then f is monotone and continuous exactly as in the proof of Proposition 7.2.11. Furthermore f is computable: note that $\nu'(n) \sqsubseteq f(\nu(m))$ iff $([m_1, \ldots, m_j]; n) \in A$ for some $m_1, \ldots, m_j \in \mathrm{supp}(\nu(m))$; but A is r.e. and the relation $m' \in \mathrm{supp}(\nu(m))$ is decidable, so the relation $\nu'(n) \sqsubseteq f(\nu(m))$ is r.e. in m, n. Thus f is a morphism in **EDom**. It remains to check that $If = h$. If $x \in D$ is finite then clearly $If(x) = \bigsqcup h(F_x) = h(x)$. More generally if $x \in D_{re}$, take $B_x \in \|x\|$ and $B_y \in \|y\|$ for each $y \in F_x$. Then $B_x = \bigcup B_y$ so

$$\mathrm{supp}(\bigsqcup h(F_x)) \;=\; \bigcup_{y \in F_x} \mathrm{supp}(h(y)) \;=\; \bigcup_{y \in F_x} A \cdot B_y \;=\; A \cdot B_x \;=\; \mathrm{supp}(h(x)),$$

and so $If(x) = \bigsqcup h(F_x) = h(x)$; thus $If = h$ as required. □

Proposition 7.3.4 *H, I preserve products.*

Proof Clearly $H(1) \cong 1$ and $I(1) \cong 1$. If D_1, D_2 are enumerated domains, clearly we have

$$\begin{aligned} |H(D_1 \times D_2)| &= |HD_1 \times HD_2| = |D_1| \times |D_2|, \\ \|(x_1, x_2) \in H(D_1 \times D_2)\| &= \{(n_1; n_2) \mid n_i \in \mathrm{supp}(x_i)\}, \\ \|(x_1, x_2) \in HD_1 \times HD_2\| &= \{\, \langle \mathrm{supp}(x_1), \mathrm{supp}(x_2) \rangle \,\}. \end{aligned}$$

Define pair', fst', $\mathit{snd}' \in \mathcal{P}\omega$ by

$$\begin{aligned} \mathit{pair}' &= \{([m]; ([n]; (m; n))) \mid m, n \in \omega\}, \\ \mathit{fst}' &= \{([(m; n)]; m) \mid m, n \in \omega\}, \\ \mathit{snd}' &= \{([(m; n)]; n) \mid m, n \in \omega\}. \end{aligned}$$

Then the evident morphism $H(D_1 \times D_2) \to HD_1 \times HD_2$ is tracked by the element $\lambda^* z.\, \mathit{pair}(\mathit{fst}'\, z)(\mathit{snd}'\, z)$, and its inverse is tracked by $\lambda^* z.\, \mathit{pair}'(\mathit{fst}\, z)(\mathit{snd}\, z)$. Thus H preserves binary products. For I, note that if D_1, D_2 are effective domains then

$$|I(D_1 \times D_2)| \;=\; |ID_1 \times ID_2| \;=\; D_{1re} \times D_{2re},$$

and that the realizers for (x_1, x_2) in $I(D_1 \times D_2)$ and $ID_1 \times ID_2$ are given by the same formulae as above. Moreover, the sets pair', fst', snd' are clearly r.e., and so just as above we have $I(D_1 \times D_2) \cong ID_1 \times ID_2$. □

Proposition 7.3.5 *H, I preserve exponentials.*

PROOF Suppose (D,ν) and (E,ν') are enumerated domains. Since H is full and faithful, there is a canonical bijection

$$|H(E^D)| \;=\; \mathrm{Hom}(D,E) \;\cong\; \mathrm{Hom}(HD,HE) \;=\; |HE^{HD}|.$$

If $f : D \to E$ is continuous, define graph $f = \{(m;n) \mid \nu'(n) \sqsubseteq f(\nu(m))\}$ as before. Then we have

$$\begin{aligned} \|f \in H(E^D)\| &= \{\{[(m_1;n_1),\ldots,(m_j;n_j)] \mid (m_i;n_i) \in \text{graph } f \text{ for each } i\}\}, \\ \|f \in HE^{HD}\| &= \{A \mid \forall x \in D.\ A \cdot \mathrm{supp}(x) = \mathrm{supp}(f(x))\}. \end{aligned}$$

Note that if $[(m_1;n_1),\ldots,(m_j;n_j)] \in \|f \in H(E^D)\|$ then we also have $[(m_i;n_i)] \in \|f \in H(E^D)\|$ for each i. Hence it is easy to see that the canonical morphism $H(E^D) \to HE^{HD}$ is tracked by $B = \{([[(m;n)]];([m];n)) \mid m,n \in \omega\}$. For the inverse morphism, suppose $A \in \|f \in HE^{HD}\|$. We first show how to extract graph f from A. Note that $(m;n) \in$ graph f iff $n \in A \cdot \mathrm{supp}(\nu(m))$ iff there exist $m_1,\ldots,m_j \in \mathrm{supp}(\nu(m))$ such that $([m_1,\ldots,m_j];n) \in A$. So consider the set

$$C \;=\; \{\,([([m_1,\ldots,m_j];n)]\,;\,(m;n)) \;\mid\; m,n \in \omega,\ m_1,\ldots,m_j \in \mathrm{supp}(\nu(m))\,\}.$$

Clearly $C \cdot A =$ graph f. Next, take $D = \{(p;p) \mid p \in \omega\}$; it is easy to check that $D \cdot (\text{graph } f) \in \|f \in H(E^D)\|$. Thus the canonical morphism $HE^{HD} \to H(E^D)$ is tracked by $\lambda^* z.\, D(Cz)$. Clearly this isomorphism identifies the evaluation morphisms. Thus H preserves exponentials.

For I, suppose (D,ν) and (E,ν') are effective domains. Since I is full and faithful, there is a canonical bijection

$$|I(E^D)| \;=\; (E^D)_{re} \;\cong\; \mathrm{Hom}(ID,IE) \;=\; |IE^{ID}|.$$

Moreover the sets B, C, D above are r.e., and so the canonical morphism $I(E^D) \to IE^{ID}$ is tracked by B, and its inverse by the element $\lambda^* z.\, D(Cz)$ of $\mathcal{P}\omega_{re}$. Hence I preserves exponentials. $\square$

In the next proposition we regard $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ as equipped with the dominances given by Example 4.3.5. We write $\perp$ for the lift functor arising from either of these dominances.

Proposition 7.3.6 *We have $H(N_\perp, \nu_0) \cong N_\perp$ in $\mathbf{Mod}(\mathcal{P}\omega)$, and $I(N_\perp, \nu_0) \cong N_\perp$ in $\mathbf{Mod}(\mathcal{P}\omega_{re})$.*

PROOF First note that we can conveniently define pairing in $\mathcal{P}\omega$ and $\mathcal{P}\omega_{re}$ by $\langle A, B\rangle = \{2n \mid n \in A\} \cup \{2n+1 \mid n \in B\}$; it is easy to see that this is realizably equivalent to the pairing operation given in Proposition 1.1.10(i). Next, by Proposition 3.3.6 we can define the natural number object in $\mathbf{Mod}(\mathcal{P}\omega)$ or $\mathbf{Mod}(\mathcal{P}\omega_{re})$ by $\|n \in N\| = \{\{n\}\}$. Hence by Remark 4.2.5(ii) the object $N_\perp$ in $\mathbf{Mod}(\mathcal{P}\omega)$ may be defined by

$$\|n \in N_\perp\| \;=\; \{\{0\}\} \wedge \{\{n\}\} \;=\; \{\{0, 2n+1\}\}, \qquad \|\perp \in N_\perp\| \;=\; \{\emptyset\} \wedge \mathcal{P}\omega.$$

Moreover the object $H(N_\perp, \nu_0)$ is given by $\|n\| = \{\{0, n+1\}\}$, $\|\perp\| = \{\{0\}\}$. To see that $H(N_\perp) \cong N_\perp$, note that the evident morphism $H(N_\perp) \to N_\perp$ is tracked by $\{\, ([n+1]; 0),\ ([n+1]; 2n+1) \mid n \in \omega\}$ since $\emptyset \in (\{\emptyset\} \wedge \mathcal{P}\omega)$; and its inverse is tracked by $\{\, ([\,]; 0)\,\} \cup \{\, ([0, 2n+1]; n+1) \mid n \in \omega\}$. Since these realizers are both r.e. sets, similarly we have $I(N_\perp, \nu_0) \cong N_\perp$. $\Box$

To show that H, I respect the fixed point operators we need the following lemma. Its proof is very similar to that of Lemma 7.2.10.

Lemma 7.3.7 *(i) If D is an enumerated domain, the order $\sqsubseteq$ on D coincides with the Σ-order on HD. Likewise if D is an effective domain, the order $\sqsubseteq$ on D_{re} coincides with the Σ-order on ID.*

(ii) If $S \subseteq D$ is directed, then S has a least upper bound x iff $S \subseteq |HD|$ has a Σ-limit x', and then $x = x'$. Likewise if $S \subseteq D_{re}$ is directed, then S has a least upper bound x in D_{re} iff $S \subseteq |ID|$ has a Σ-limit x', and then $x = x'$.

PROOF (i) Suppose $x \sqsubseteq y$ in D. Then if $A \in \|x \in HD\|$ and $B \in \|y \in HD\|$ then $A \subseteq B$. Hence if $\phi : HD \to \Sigma$ is tracked by C then $C \cdot A \subseteq C \cdot B$ and so $\phi(x) \leq \phi(y)$ in Σ; thus $x \preceq_\Sigma y$ in HD. Conversely, if $x \not\sqsubseteq y$ then take $n \in \mathrm{supp}(x)$, $n \notin \mathrm{supp}(y)$, and define $\phi : HD \to \Sigma$ by $\phi(z) = \top$ iff $n \in \mathrm{supp}(z)$ (it is easy to give a realizer for ϕ). Then $\phi(x) = \top$ and $\phi(y) = \perp$, so $x \not\preceq_\Sigma y$. The proof for ID is exactly similar—just observe that the morphism ϕ defined above is tracked by an element of $\mathcal{P}\omega_{re}$.

(ii) Suppose S has a least upper bound x in D. If $A_x \in \|x\|$ and $A_y \in \|y\|$ for each $y \in S$, then $A_x = \bigcup A_y$; hence if $\phi : HD \to \Sigma$ is tracked by C then $C \cdot A_x = \bigcup C \cdot A_y$ and so $\phi(x) = \top$ iff $\phi(y) = \top$ for some $y \in S$; thus x is a Σ-limit of S. Conversely, if x' is a Σ-limit of S in HD then for all $y \in S$ we have $y \preceq_\Sigma x'$ and so $y \sqsubseteq x$ by (i); thus $x \sqsubseteq x'$ where $x = \bigsqcup S$. If $x \neq x'$ then take $n \in \mathrm{supp}(x)$, $n \notin \mathrm{supp}(x)$ and construct ϕ as above. Then $\phi(x') = \top$ but $\phi(y) = \bot$ for all $y \in S$, a contradiction. Thus $x = x'$. The proof for ID is precisely similar. $\Box$

Now recall from Proposition 5.3.7 that our dominances on $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ satisfy the Completeness Axiom, so we obtain a morphism fix_X for every useful X. Next, since H, I both preserve $N_\bot$ and exponentials, for any PCF type σ we have $H([\![\sigma]\!]) \cong [\![\sigma]\!]_N$ in $\mathbf{Mod}(\mathcal{P}\omega)$, and $I([\![\sigma]\!]) \cong [\![\sigma]\!]_N$ in $\mathbf{Mod}(\mathcal{P}\omega_{re})$. This enables us to state the following:

Proposition 7.3.8 *For each type σ we have $H(\mathit{fix}_{[\![\sigma]\!]}) \cong \mathit{fix}_{[\![\sigma]\!]_N}$ and $I(\mathit{fix}_{[\![\sigma]\!]}) \cong \mathit{fix}_{[\![\sigma]\!]_N}$ modulo the canonical isomorphisms.*

Proof The proof is precisely similar to that of Proposition 7.2.13, and uses the above lemma. $\Box$

The above propositions may be summarized as follows:

Theorem 7.3.9 *The functors $H : \mathbf{NDom} \to \mathbf{Mod}(\mathcal{P}\omega)$ and $I : \mathbf{EDom} \to \mathbf{Mod}(\mathcal{P}\omega_{re})$ are full, faithful and cartesian-closed, and preserve the object $N_\bot$ and the morphisms $\mathit{fix}_{[\![\sigma]\!]}$ up to isomorphism.* $\Box$

Remark 7.3.10 Although our main interest is in $\mathbf{Mod}(\mathcal{P}\omega_{re})$, it is a pleasing observation that the category **Dom** of Scott domains arises as a full sub-CCC of the topos $\mathbf{RT}(\mathcal{P}\omega_{re})$. In other words, classical domains may legitimately be thought of as "sets" in this realizability topos. This provides a non-trivial instance of how one kind of classical domain theory may be recovered as a special case of synthetic domain theory. It would be interesting to know whether this result extends further: for instance, does the embedding H preserve initial solutions of recursive domain equations?

7.3.2 Interpreting PCF^{++} in $\mathbf{Mod}(\mathcal{P}\omega_{re})$

We are now in a position to give interpretations of PCF^{++} in $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ exactly as in Paragraph 7.2.2:

Proposition 7.3.11 *The PCF-models $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ both support* PCF^{++}. *Moreover, for any* PCF_N^{++} *term $M : \sigma$ in environment Γ, the morphisms $H[\![M]\!]'^{\Gamma} : H[\![\Gamma]\!] \to H[\![\sigma]\!]$ and $[\![M]\!]_N : [\![\Gamma]\!]_N \to [\![\sigma]\!]_N$ are canonically isomorphic in $\mathbf{Mod}(\mathcal{P}\omega)$. Likewise, the morphisms $I[\![M]\!]^{\Gamma}$ and $[\![M]\!]_N$ are canonically isomorphic in $\mathbf{Mod}(\mathcal{P}\omega_{re})$.*

Proof Precisely similar to the proof of Proposition 7.2.15. □

Theorem 7.3.12 *(i) The interpretations of* PCF_N^{+}, PCF_V^{+}, PCF_L^{+} *in $\mathbf{Mod}(\mathcal{P}\omega)$ and in $\mathbf{Mod}(\mathcal{P}\omega_{re})$ are fully abstract.*

(ii) The interpretations of PCF_N^{++}, PCF_V^{++}, PCF_L^{++} *in $\mathbf{Mod}(\mathcal{P}\omega_{re})$ are universal.*

Proof Precisely similar to Theorem 7.2.16. □

Remarks 7.3.13 (i) From Theorem 7.3.12 we thus obtain yet another characterization of the HPEOs (cf. Paragraph 7.2.2) as the partial functionals over $N_\perp$ in the category $\mathbf{Mod}(\mathcal{P}\omega_{re})$.

(ii) It similarly follows from Theorem 7.3.9 that the hierarchy of partial functionals over N in $\mathbf{Mod}(\mathcal{P}\omega)$ coincides with that in **Dom**. It is an easy exercise in classical domain theory to check that this in turn coincides with the hierarchy of partial functionals in the category of (chain-)CPOs. This answers affirmatively a question posed by Phoa in [75, Section 4].

(iii) As an aside, we note that the hierarchy of *total* functionals over N in $\mathbf{Mod}(\mathcal{P}\omega_{re})$ also coincides with that in $\mathbf{Mod}(K_1)$. Beeson in [10, Section VI.8] mentions that the total functionals in $\mathbf{Mod}(\mathcal{P}\omega_{re})$ are precisely Kreisel's "hereditarily recursively continuous functionals", also known as ECF(Rec); and it is proved in [99, Section 2.6] that these are equivalent to the hereditarily effective operations. Likewise, the total functionals in $\mathbf{Mod}(\mathcal{P}\omega)$ are precisely Kreisel's "hereditarily continuous functionals", also known as ECF (this is proved in [12, Chapter 7]). These are in fact the same as Kleene's "countable functionals"—for a detailed study of this type structure see [37].

We now turn our attention to a property of $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ not shared by $\mathbf{Mod}(K_1)$: all objects corresponding to PCF types are *projective* in the sense of Definition 2.4.8. Recall from Proposition 2.4.9 and Remark 2.4.11 that for a modest set X the notions of projectivity in $\mathbf{Mod}(A)$, $\mathbf{Ass}(A)$ and $\mathbf{RT}(A)$ all coincide.

Proposition 7.3.14 *Suppose E is one of the three evaluation mechanisms N, V, L (cf. Section 6.3). Then for any type σ, the objects C_E^σ, V_E^σ (in both $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$) are projective.*

Proof We have noted already that all objects of the form HD, ID have the "singleton property", and hence are projective by Proposition 2.4.10. Thus all the objects $C_N^\sigma = V_N^\sigma$ are projective. Now recall from Proposition 6.3.6 that each object C_L^σ is a retract of some C_N^τ, and hence is a retract of some object X the singleton property. It is thus easy to see that C_L^σ is isomorphic to an evident subobject of X, and hence that $C_L^\sigma = V_L^\sigma$ is projective. Similarly, by Proposition 6.3.3 each object C_V^σ is a retract of some C_L^τ, whence C_V^σ is projective. Finally, it is easy to see that V_V^σ is a regular subobject of $C_V^\sigma = (V_V^\sigma)_\perp$, and so V_V^σ is also projective. $\square$

Experts will recognize that the projective objects in $\mathbf{RT}(A)$ are precisely those for which (a form of) the *axiom of choice* holds in the internal logic. From an external point of view, the force of this is as follows. Suppose X, Y are assemblies where X has the singleton property, and suppose $r \in A$ is such that whenever $a \in \|x\|$ for some $x \in X$ we have $ra \in \|y\|$ for some $y \in |Y|$. Then clearly r realizes a morphism $X \to Y$, since the requirement that r should behave "extensionally" on realizers for the same element $x \in X$ is vacuous. (We will return to this point in Section 8.2.)

It is worth pausing to see why Proposition 7.3.14 fails for $\mathbf{Mod}(K_1)$. We will show in particular that in this model the object $N_\perp^N$ is not projective. Clearly the object $N_\perp^N$ may be presented as follows:

$$|N_\perp^N| \;=\; \{f : \mathrm{N} \rightharpoonup \mathrm{N} \mid f \text{ is partial recursive}\}, \qquad \|f \in N_\perp^N\| \;=\; \{e \mid \{e\} = f\}.$$

Now suppose we have an isomorphism $N_\perp^N \to X$ tracked by t, where X has the singleton property. Then e, e' code the same partial recursive function iff $te = te'$. Hence equality between partial recursive functions is decidable—a contradiction.

Remark 7.3.15 Curiously, although in $\mathbf{Mod}(\mathcal{P}\omega)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$ all the objects in the partial type hierarchy over N are projective, this is not true for the *total* type hierarchy. It can be shown that though the object N^N is projective, the object $N^{(N^N)}$ is not.

7.4 Term models and sequential PCF

In Chapter 3 we gave a proof that the PCAs $\mathcal{P}\omega_{re}$ and Λ^0/T are inequivalent whenever T is a semi-sensible λ-theory, i.e. T does not equate a solvable and an unsolvable term (see Theorem 3.3.10). The proof depended on the absence of a certain "parallel" operation in Λ^0/T, and at the end of the chapter we suggested that one might expect the toposes $\mathbf{RT}(\Lambda^0/T)$ to provide good models for sequential programming languages. In this section we investigate this idea in more detail.

Let us consider categories of the form $\mathbf{Mod}(\Lambda^0/T)$ for T semi-sensible. By considering the set of unsolvable terms as a divergence we obtain natural dominances in these categories (see Example 4.3.5(iii)); moreover, in Proposition 5.3.4 we saw that these dominances satisfy the Completeness Axiom. Hence by the results of Chapter 6 we have an adequate interpretation of sequential PCF in these models. Encouraged by our success in obtaining universality theorems for parallel languages (Theorems 7.2.16 and 7.3.12 above), we are naturally led to conjecture the following:

Conjecture 7.4.1 *For any semi-sensible theory T, the interpretation of PCF in* $\mathbf{Mod}(\Lambda^0/T)$ *is universal.*

This result was (implicitly) conjectured by Phoa in [72] for a particular theory T. We strongly believe the conjecture to be true, though so far we have been unable to prove it. We also believe that a similar result should hold for many other term models, for instance for those obtained from the call-by-value λ-calculus (see Paragraph 3.2.4). We have concentrated mainly on the term models Λ^0/T because there is a particularly well developed and widely known body of theory for the classical λ-calculus (see [8]). However, our attempts at proving the conjecture have raised some interesting questions about the λ-calculus that do not seem to be addressed by this theory.

In Paragraph 7.4.1 we give a rather loose assortment of some partial results in the direction of the main conjecture, culminating in a proof that the interpretation of PCF_V in these models is fully abstract at type 3. In Paragraph 7.4.2 we describe a strategy for proving the main conjecture itself, and discuss some of the hard questions that arise from it.

7.4.1 Some partial results

In [72], Phoa considered the interpretation of PCF_N in the model $\mathbf{Mod}(\Lambda^0/\mathcal{B})$, where $\mathcal{B}$ is the theory that equates terms iff they have the same Böhm tree. We will first show that a universality result for this particular model would immediately imply a similar result for a large class of other semi-sensible theories. The proof is a simple but pleasing application of the theory developed in Chapter 2.

Proposition 7.4.2 *Suppose T is any λ-theory such that $T \subseteq \mathcal{B}$. Then*

(i) $\mathbf{Mod}(\Lambda^0/T)$ *is a universal model for PCF iff* $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ *is so;*

(ii) $\mathbf{Mod}(\Lambda^0/T)$ *is a fully abstract model for PCF iff* $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ *is so.*

Proof First note that if $T \subseteq \mathcal{B}$ then T is semi-sensible. By Proposition 3.2.4 we have a canonical applicative inclusion $(\gamma \dashv \delta) : \Lambda^0/\mathcal{B} \to \Lambda^0/T$; moreover by Proposition 3.2.5 the applicative morphism $\delta : \Lambda^0/\mathcal{B} \longrightarrow\!\!\!\!\!\to \Lambda^0/T$ is decidable. Hence by

Propositions 2.4.19 and 2.5.11(i) the functor $\delta_* : \mathbf{Ass}(\Lambda^0/\mathcal{B}) \to \mathbf{Ass}(\Lambda^0/T)$ is full and faithful, and preserves exponentials and the natural number object. Moreover, for our choice of dominances on $\Lambda^0/\mathcal{B}$ and Λ^0/T, it is easy to see that δ is a *domin-ical morphism* in the sense of Definition 4.2.10; hence by Proposition 4.2.11 the functor δ_* commutes with lifting up to isomorphism. Thus $\delta_*([\![\sigma]\!]_N) = [\![\sigma]\!]_N$ for any PCF type σ. Finally, it is an easy exercise to check that $\delta_*(\mathit{fix}_{[\![\sigma]\!]})$ is identified with $\mathit{fix}_{[\![\sigma]\!]}$ by the canonical isomorphisms. Thus δ_* preserves the interpretations of terms of PCF_N (cf. Proposition 7.2.15). Both parts of the Proposition are now straightforward using the fact that δ_* is full and faithful. $\Box$

Note that this argument breaks down if $T \not\subseteq \mathcal{B}$. Although one always obtains an applicative morphism $\epsilon : \Lambda^0/T \longrightarrow\!\!\!\!\mapsto \Lambda^0/\beta$ such that ϵ_* preserves exponentials, this morphism will not in general be decidable and so ϵ_* may fail to preserve the natural number object. Nevertheless, the objects $\epsilon_*(N)$ and N will look quite similar, and it is hard to believe that the partial type structures arising from $\mathbf{Mod}(T)$ and $\mathbf{Mod}(\mathcal{B})$ could be different!

Although we cannot prove Conjecture 7.4.1 even for some particular theory T, we can obtain universality and full abstraction results at certain "low" types. For example, the "first-order" functions $(N_\perp)^k \to N_\perp$ representable by morphisms of $\mathbf{Mod}(\Lambda^0/T)$ are precisely the PCF-definable ones—here we will prove this result for the case $T \subseteq \mathcal{B}$. First we fix on a convenient presentation of the object $N_\perp$. Define $\overline{n} = \lambda x f. f^n x$; note that if $T \vdash M = \overline{n}$ then $M =_\beta \overline{n}$ since $T \subseteq \mathcal{B}$. Moreover the elements of Λ^0/T given by $\overline{0}, \overline{1}, \ldots$ are equivalent to the standard choice of numerals by Proposition 1.1.15. Hence the following is an acceptable presentation of $N_\perp$:

$$|n \in N_\perp| \;=\; \{\overline{n}\}, \qquad |\perp \in N_\perp| \;=\; \{M \mid M \text{ unsolvable}\}$$

(here we confuse λ-terms with their equivalence classes modulo T).

Theorem 7.4.3 *Suppose $T \subseteq \mathcal{B}$ is some λ-theory. Then a set-theoretic function $f : |(N_\perp)^k| \to |N_\perp|$ gives a morphism in $\mathbf{Mod}(\Lambda^0/T)$ iff there is a closed term $F : \underbrace{\iota \to \cdots \to \iota}_{k} \to \iota$ of PCF_N such that $\mathit{uncurry}([\![F]\!]) = f$.*

PROOF The right-to-left implication is trivial, since if such a term F exists then $uncurry([\![F]\!])$ is indeed a morphism $(N_\perp)^k \to N_\perp$ in $\mathbf{Mod}(\Lambda^0/T)$. For the converse implication, suppose $f : (N_\perp)^k \to N_\perp$ is a morphism in $\mathbf{Mod}(\Lambda^0/T)$. Then from a realizer for f it is easy to construct a term $R \in \Lambda^0$ such that whenever $T_i \in \|m_i \in N_\perp\|$ for $1 \leq i \leq k$, we have $RT_1 \ldots T_k \in \|f(m_1, \ldots, m_k) \in N_\perp\|$. Hence if $f(m_1, \ldots, m_k) = n$ then $RT_1 \ldots T_k \twoheadrightarrow \overline{n}$, where $\twoheadrightarrow$ denotes many-step leftmost reduction; and if $f(m_1, \ldots, m_k) = \perp$ then $RT_1 \ldots T_k$ has an infinite leftmost reduction sequence (see [8, Section 13.2]).

The basic strategy now is as follows: we write a PCF program F which, given arguments $m_1, \ldots, m_k$, simulates the leftmost reduction of the term $RT_1 \ldots T_k$ using a Gödel-numbering for untyped λ-terms. If this ever terminates, it does so with (a code for) a term of the form $\overline{n}$, and this can be "decoded" to obtain the result n. This strategy depends on the fact that from the argument m_i we can effectively obtain (a code for) the realizer T_i. Of course, one must be careful to do all this in a "lazy" way, only evaluating each m_i when it is needed, since one cannot hope to obtain a code for T_i when m_i is undefined.

More formally, let $\Lambda^0(z_1, \ldots, z_k)$ be the set of terms $M \in \Lambda$ such that $\mathrm{FV}(M) \subseteq \{z_1, \ldots, z_k\}$. (Here the z_i are formal symbols used to stand for the realizers T_i.) For any $T_1, \ldots, T_k \in \Lambda^0$ there is an evident notion of $\beta\delta_{T_1 \ldots T_k}$-*reduction* on $\Lambda^0(\vec{z})$ given by the δ-rules $z_1 \to T_1, \ldots, z_k \to T_k$. Given $M \in \Lambda^0(\vec{z})$, the *leftmost* $\beta\delta$-redex L_M in M (if any) is defined to be the leftmost-outermost subterm of M of either of the forms $(\lambda x.N)P$ or z_i (i.e. the subterm of either form whose leftmost character appears leftmost in M). We write $\twoheadrightarrow_{T_1 \ldots T_k}$ (or just $\twoheadrightarrow_T$) for the evident many-step leftmost $\beta\delta_{T_1 \ldots T_k}$-reduction relation. We also say M is in $\beta\delta$-*normal form* if it is in β-nf and contains no z_i free. The following fact is now evident:

Lemma 7.4.4 *Suppose $M, M' \in \Lambda^0(\vec{z})$ and M' is in $\beta\delta$-nf. Then $M \twoheadrightarrow_T M'$ iff $M[T_1/z_1, \ldots, T_k/z_k] \twoheadrightarrow M'$. Hence $Rz_1 \ldots z_k \twoheadrightarrow_T \overline{n}$ iff $RT_1 \ldots T_k \twoheadrightarrow \overline{n}$.* $\Box$

Now suppose $\ulcorner - \urcorner$ is an effective Gödel-numbering for elements of $\Lambda^0(\vec{z})$. Since all partial recursive functions are expressible in PCF_N, it is clear that one can construct terms $Encode$, $Decode$, $Normal : \iota \to \iota$ of PCF_N satisfying the following specifications:

$$\begin{array}{rclcrcl} Encode\; n & \Downarrow & \ulcorner \overline{n} \urcorner, & & Decode\; \ulcorner \overline{n} \urcorner & \Downarrow & n, \\ Normal\; \ulcorner M \urcorner & \Downarrow & 0 \;\; (M \text{ in } \beta\delta\text{-nf}), & & Normal\; \ulcorner M \urcorner & \Downarrow & 1 \;\; (M \text{ not in } \beta\delta\text{-nf}). \end{array}$$

With a little more effort, one can also construct a term $OneStep : \iota \to \iota \to \cdots \to \iota$ satisfying the following for every $M \in \Lambda^0(\vec{z})$, $P_1, \ldots, P_k : \iota$.

- If L_M is a β-redex then $OneStep\; \ulcorner M \urcorner P_1 \ldots P_k \Downarrow \ulcorner M' \urcorner$, where M' is obtained from M by contracting L_M. (Thus $OneStep\; \ulcorner M \urcorner$ is not strict in any of its k arguments.)
- If $L_M = z_i$ and $P_i \Downarrow \ulcorner T_i \urcorner$ then $OneStep\; \ulcorner M \urcorner P_1 \ldots P_k \Downarrow \ulcorner M' \urcorner$, where M' is obtained from M by replacing the subterm L_M with T_i. (Thus $OneStep\; \ulcorner M \urcorner$ is strict in its ith argument, but not in any of the others.)

Next we define a PCF-term $Reduce : \underbrace{\iota \to \cdots \to \iota}_{k+1} \to \iota$ by

$$Reduce \;=\; \lambda y_1^\iota \ldots y_k^\iota.\, \mu g^{\iota\to\iota}.\, \lambda x^\iota.\, \mathsf{cond}\; (Normal\; x)\; x\; (g(OneStep\; x y_1 \ldots y_k)).$$

Let us say that an index i is *needed* for M with respect to $T_1, \ldots, T_k$ if there exists M' such that $M \twoheadrightarrow_T M'$ and $L_{M'} = z_i$ (i.e. the δ-rule $z_i \to T_i$ is used in the leftmost reduction of M). Then we can easily prove the following properties for $Reduce$ (we omit the boring details):

- Suppose $P_i \Downarrow \ulcorner T_i \urcorner$ whenever i is needed for M w.r.t. $T_1, \ldots, T_k$. Then $Reduce\; P_1 \ldots P_k \ulcorner M \urcorner \Downarrow \ulcorner M' \urcorner$ if $M \twoheadrightarrow_T M'$ where M' is in $\beta\delta$-nf; and $Reduce\; P_1 \ldots P_k \ulcorner M \urcorner \Uparrow$ if M has no $\beta\delta$-nf.
- Suppose for some i needed for M w.r.t. $T_1, \ldots, T_k$ we have $P_i \Uparrow$. Then $Reduce\; P_1 \ldots P_k \ulcorner M \urcorner \Uparrow$.

Finally we define

$$F \;=\; \lambda m_1^\iota \ldots m_k^\iota.\, Decode(Reduce(Encode\; m_1) \ldots (Encode\; m_k)\, \ulcorner R z_1 \ldots z_k \urcorner).$$

We will show that $uncurry([\![F]\!]) = f$ pointwise. Suppose $f(m_1, \ldots, m_k) = n$ where $m_1, \ldots, m_k, n \in \mathrm{N}_\perp$ and $m_i = [\![M_i]\!]$ for each i. We wish to check that $[\![FM_1 \ldots M_k]\!] = n$. We write $T_i = \overline{m_i}$ for each i, where by convention $\overline{\perp} = \Omega$. We also write R' for $Rz_1 \ldots z_k$, and *Reduce*$'$ for *Reduce*(*Encode* M_1)$\ldots$(*Encode* M_k).

First suppose $n \in \mathrm{N}$. Then $RT_1 \ldots T_k \twoheadrightarrow \overline{n}$, so $R' \twoheadrightarrow_T \overline{n}$. In particular if i is needed for R' then T_i is solvable (otherwise the head reduction sequence for T_i would yield an infinite leftmost reduction sequence for R'); hence $m_i \in \mathrm{N}$ and so *Encode* $P_i \Downarrow \lceil T_i \rceil$. Thus by the above properties of *Reduce* we have *Reduce*$'$ $\lceil R' \rceil \Downarrow \lceil \overline{n} \rceil$; hence $FM_1 \ldots M_k \Downarrow n$ and so $[\![FM_1 \ldots M_k]\!] = n$.

Now suppose $n = \perp$; then $RT_1 \ldots T_k$ is unsolvable and so R' has no $\beta\delta$-nf. If $m_i \in \mathrm{N}$ whenever i is needed for R' then *Encode* $M_i \Downarrow \lceil T_i \rceil$ for all such i; thus by the above properties of *Reduce* we have *Reduce*$'$ $\lceil R' \rceil \Uparrow$. Hence $FM_1 \ldots M_k \Uparrow$ and so $[\![FM_1 \ldots M_k]\!] = \perp$ by the Adequacy Theorem for PCF_N. Otherwise there exist i, R'' such that $R' \twoheadrightarrow_T R''$, $L_{R''} = z_i$ and $m_i = \perp$. Consider the pair (i, R'') with this property for which the length of the reduction $R' \twoheadrightarrow_T R''$ is minimal. It is easy to see that *Reduce*$'$ $\lceil R' \rceil \Downarrow p$ iff *Reduce*$'$ $\lceil R'' \rceil \Downarrow p$. But *Encode* $M_i \Uparrow$ and so *Reduce*$'$ $\lceil R'' \rceil \Uparrow$. Hence we again have $FM_1 \ldots M_k \Uparrow$ and so $[\![FM_1 \ldots M_k]\!] = \perp$.
$\square$

Remarks 7.4.5 (i) In fact the condition $T \subseteq \mathcal{B}$ in the above theorem can be relaxed to the requirement that T is semi-sensible (i.e. $T \subseteq \mathcal{K}^*$; see [8, Section 17.1]). In this case we may have $T \vdash M = \overline{n}$ but not $M \twoheadrightarrow \overline{n}$, so instead of simulating the reduction of M to normal form we just have to reduce M until enough of the Böhm tree of M is "visible" to allow us to extract the answer n. However, the extra effort involved in proving this stronger result is considerable, so we have contented ourselves with giving a proof for the case $T \subseteq \mathcal{B}$.

(ii) It follows easily from the above theorem that all morphisms $f : (N_\perp)^k \to N_\perp$ in our model are denotable by k-ary term contexts of *call-by-value* PCF. For if F is a PCF_N term denoting f, by the translation given in Definition 6.3.10 we obtain a PCF_V term $\widehat{F} : (\iota \to \iota) \to \cdots \to (\iota \to \iota)$. Now let $G[-, \ldots, -]$ be the

PCF_V term context defined by

$$G[x_1^\iota, \ldots, x_k^\iota] \;=\; \widehat{F}\,(\lambda z^\iota.x_1)\cdots(\lambda z^\iota.x_k)\,0.$$

Then from Proposition 6.3.12 it is clear that $[\![\,G\,]\!] = f$.

(iii) It seems worth pointing out that, for computable functions $f : (\mathrm{N}_\perp)^k \to \mathrm{N}_\perp$, PCF-definability is actually a stronger condition than the Milner-Vuillemin notion of sequentiality (see e.g. [61]). This was pointed out by M.B. Trakhtenbrot in [98], who gave an example of a three-argument function that is computable and sequential but not PCF-definable. In fact one can give examples with just two arguments, e.g.:

$$f(x,y) \;=\; \begin{cases} 0 & \text{if } x \Downarrow \text{ and } (x \in K \text{ or } y \Downarrow) \\ \perp & \text{otherwise} \end{cases}$$

where K is the halting set $\{n \mid \{n\}(0)\downarrow\}$. (This example is similar to one given in Section 2 of [86].)

We now turn our attention to the functionals of higher type in $\mathbf{Mod}(\Lambda^0/T)$. Although one can easily show the absence of certain particular "parallel" functionals such as the $\exists$ functional (cf. Proposition 6.2.10), the problem of proving universality even at type 2 seems to present prodigious difficulties. In the next paragraph we will discuss a strategy for proving the following:

Conjecture 7.4.6 *Every morphism $(N_\perp)^N \to N_\perp$ in $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is denotable by a term of* PCF_V *of type $(\iota \to \iota) \to \iota$.*

(This seems slightly weaker—hence perhaps easier to prove—than the corresponding fact for call-by-name PCF.) Although we have not managed to prove this conjecture, we give below a proof of a closely related result, namely that the interpretation of PCF_V in $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is *fully abstract at type 3*. The proof uses the fact that the object $(N_\perp)^N$ has a particularly nice presentation—this gives us a very good grasp of the properties of this object.

The crucial observation is that any morphism $f : N \to N_\perp$ $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ has a *canonical* realizer R_f in $\Lambda^0/\mathcal{B}$. The Böhm tree corresponding to R_f may be schematically depicted as follows:

$$\begin{tikzcd}[column sep=small]
 & \lambda x.\, \mathit{if}\ x=\overline{0} \arrow[dl, no head] \arrow[dr, no head] & & \\
\overline{f0} & & \mathit{if}\ x=\overline{1} \arrow[dl, no head] \arrow[dr, no head] & \\
 & \overline{f1} & & \ddots
\end{tikzcd}$$

(where again we take $\overline{\perp} = \Omega$). Note that if f is partial recursive then this tree is r.e. and has no free variables; hence by [8, Theorem 10.1.23] it is indeed the Böhm tree of some term R_f. Intuitively $\mathrm{BT}(R_f)$ embodies the infinite "look-up table" for f. Moreover, as we shall shortly see, there is a realizer T that transforms any realizer for a morphism f into this canonical realizer. It may seem surprising that such a canonical choice of realizer exists at all, in view of the fact that there is no canonical "algorithm" for computing f. One way of explaining this apparent paradox is to say that by identifying terms with the same Böhm tree we abstract away from the "algorithmic" process by which f is actually computed.

The following proposition expresses these ideas slightly more formally:

Proposition 7.4.7 *We may assign to every partial recursive* $f : \mathrm{N} \to \mathrm{N}_\perp$ *an element* $R_f \in \Lambda^0/\mathcal{B}$*, in such a way that*

(i) $f \subseteq g$ *iff* $\mathrm{BT}(R_f) \subseteq \mathrm{BT}(R_g)$*;*

(ii) if S *is a set of partial recursive functions such that* $f \subseteq g$ *for each* $f \in S$*, then* $g = \bigsqcup S$ *iff* $\mathrm{BT}(R_g) = \bigcup_{f \in S} \mathrm{BT}(R_f)$*;*

(iii) the object $(N_\perp)^N$ *in* $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ *may be presented by* $\|f \in (N_\perp)^N\| = \{R_f\}$*. Thus* $(N_\perp)^N$ *is projective.*

Proof To define R_f formally we need to clarify the meaning of the labels $\mathit{if}\ x = \overline{n}$ in the picture above. In particular, to know that this picture does define a Böhm tree, we need to replace these labels with expressions that begin with a head variable. From Theorem 1.1.16 it is clear that there exists an element $Q \in \Lambda^0/BB$ such that $\mathcal{B} \vdash Q\,\overline{m}\,\overline{n} = \lambda xy.x$ if $m = n$, and $\mathcal{B} \vdash Q\,\overline{m}\,\overline{n} = \lambda xy.y$ if $m \neq n$. Rather than explicitly calculate Q we can use the following trick. Since $\overline{m}ii = i$ for any $m \in \mathrm{N}$ (where $i = \lambda y.y$), we have

$$\overline{m}ii\,(Q\,\overline{m}\,\overline{n})\,P_1P_2 \;=\; \begin{cases} P_1 & \text{if } m = n \\ P_2 & \text{if } m \neq n. \end{cases}$$

Thus we may define R_f to be the (unique) element of $\Lambda^0/\mathcal{B}$ whose Böhm tree is

$$\begin{tikzcd}[arrows=dash]
& \lambda x.\, xii(Qx\overline{0}) \arrow[dl] \arrow[dr] & & \\
\mathrm{BT}(\overline{f0}) & & xii(Qx\overline{1}) \arrow[dl] \arrow[dr] & \\
& \mathrm{BT}(\overline{f1}) & & \ddots
\end{tikzcd}$$

where of course $xii(Qx\overline{n})$ really stands for the tree fragment

$$\begin{tikzcd}[arrows=dash]
& x \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{BT}(i) & \mathrm{BT}(i) & \mathrm{BT}(Qx\overline{n}).
\end{tikzcd}$$

(As noted above, such an element R_f exists because this tree is closed and r.e.) Properties (i) and (ii) are now immediate since $\mathrm{BT}(\overline{\bot}) = \bot$. For (iii), first note that for any $n \in \mathrm{N}$ we have

$$R_f\,\overline{n} \;=\; \overline{n}ii\,(Q\,\overline{n}\,\overline{0})\,\overline{f0}\,(\ldots(\overline{n}ii\,(Q\,\overline{n}\,\overline{n})\,\overline{fn}\,U)\ldots)$$

for some term U; hence $R_f\,\overline{n} = \overline{fn}$. Thus R_f tracks f relative to the presentations of $N, N_\bot$ given by $\|n \in N\| = \|n \in N_\bot\| = \{\overline{n}\}$, $\|\bot \in N_\bot\| = \{\Omega\}$. Now let T be the unique element of $\Lambda^0/\mathcal{B}$ whose Böhm tree is

$$\begin{tikzcd}[arrows=dash]
& \lambda fx.\, xii(Qx\overline{0}) \arrow[dl] \arrow[dr] & & \\
f \arrow[d] & & xii(Qx\overline{1}) \arrow[dl] \arrow[dr] & \\
\mathrm{BT}(\overline{0}) & f \arrow[d] & & \ddots \\
& \mathrm{BT}(\overline{1}) & &
\end{tikzcd}$$

(again such a T exists since this tree is r.e.). Suppose $M \in \Lambda^0/\mathcal{B}$ tracks f relative to the above presentations of $N, N_\perp$; we claim that $TM = R_f$ in $\Lambda^0/\mathcal{B}$. To see this, for each $k > 0$ define $T^k, R_f^k \in \Lambda^0/\mathcal{B}$ by

$$\begin{array}{rcl} T^k & = & \lambda fx.\, xii\,(Qx\overline{0})\,(f\overline{0})\,(\ldots(xii\,(Qx\overline{k-1})\,(f\overline{k-1})\,\Omega)\ldots) \\ R_f^k & = & \lambda x.\, xii\,(Qx\overline{0})\;\overline{f0}\;(\ldots(xii\,(Qx\overline{k-1})\;\overline{f(k-1)}\;\Omega)\ldots). \end{array}$$

Clearly $T^k M = R_f^k$ for each k. But $\mathrm{BT}(T) = \bigcup \mathrm{BT}(T^k)$ and $\mathrm{BT}(R_f) = \bigcup \mathrm{BT}(R_f^k)$, so by [8, Theorem 14.3.22] we have $TM = R_f$ in $\Lambda^0/\mathcal{B}$. Property (iii) now follows easily: if X is the object $(N_\perp)^N$ defined as in Proposition 1.2.4 and Y is the object defined by $|Y| = |X|$, $\|f \in Y\| = \{R_f\}$, then the canonical morphism $Y \to X$ is tracked by i, and its inverse is tracked by T. Finally, Y has the singleton property and so is projective by Proposition 2.4.10. $\Box$

Using this proposition, it is easy to show that both the Σ-order and the notion of Σ-limit in $(N_\perp)^N$ coincide with the pointwise notions:

Proposition 7.4.8 *The following hold for the model* $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ *equipped with the dominance given by the divergence* $\{\Omega\}$*:*

(i) If $f, g \in |(N_\perp)^N|$*, then* $f \preceq_\Sigma g$ *iff* $f \subseteq g$*.*

(ii) If $g \in |(N_\perp)^N|$*,* $S \subseteq |(N_\perp)^N|$ *and* $f \subseteq g$ *for each* $f \in S$*, then* g *is the* Σ*-limit of* S *iff* $g = \bigcup S$*.*

Proof (i) Suppose $f \subseteq g$, and $\phi : (N_\perp)^N \to \Sigma$ is any morphism, tracked by P say. Then $\mathrm{BT}(R_f) \subseteq \mathrm{BT}(R_g)$ by the above proposition, hence $\mathrm{BT}(PR_f) \subseteq \mathrm{BT}(PR_g)$ by [8, Corollary 14.3.20] and so $\phi(f) \leq \phi(g)$. Thus $f \preceq_\Sigma g$. Conversely, suppose $f \not\subseteq g$; then for some n we have $\perp \neq f(n) \neq g(n)$. Using the "evaluate at n" morphism $(N_\perp)^N \to N_\perp$ it is easy to construct a morphism $(N_\perp)^N \to \Sigma$ refuting $f \preceq_\Sigma g$.

(ii) Suppose $g = \bigcup S$, and P tracks ϕ as in (i). Then $\mathrm{BT}(R_g) = \bigcup_{f \in S} \mathrm{BT}(R_f)$ by the above proposition, hence $\mathrm{BT}(PR_g) = \bigcup_{f \in S} \mathrm{BT}(PR_f)$ by [8, Theorem 14.3.22]. Thus $\phi(g) = \top$ iff $\phi(f) = \top$ for some $f \in S$. Conversely, if $g \neq \bigcup S$ then for some n we have $g(n) \neq \perp$ but $f(n) = \perp$ for all $f \in S$; hence we may easily construct ϕ such that $\phi(g) = \top$ but $\phi(f) = \perp$ for all $f \in S$. $\Box$

These results for $(N_\perp)^N$ seem to be something of a flash in the pan—there seems to be no easy way of obtaining similar results for $(N_\perp)^{N_\perp}$, for instance. (In fact, we have not even been able to prove that the Σ-order on Σ^Σ coincides with the pointwise order!) However, it would follow easily from Conjecture 7.4.1 that for all objects $[\![\,\sigma\,]\!]$ the Σ-order coincides with the pointwise order, and Σ-limits coincide with least upper bounds.

Proposition 7.4.8 seems quite interesting in its own right. However, in order to prove our full abstraction result we need a slightly different though related fact:

Lemma 7.4.9 *For each $k > 0$, let $D_k[-]$ be the* PCF$_V$ *term context defined by*

$$D_k[f^{\iota\to\iota}] \;=\; \lambda x^\iota.\,\mathsf{cond}\;(x < k)\,(f\,x)\,\Omega^\iota$$

where $\Omega^\iota = \mu y^\iota.y$ and $<$ is some (infix) PCF$_V$ *term implementing the order relation. Let $\theta_k = [\![\,D_k\,]\!]_V : (N_\perp)^N \to (N_\perp)^N$. Then the Σ-limit of the morphisms θ_k exists and is the identity morphism.*

Proof We work with the presentation of $(N_\perp)^N$ given by $\|f\| = \{R_f\}$. It is easy to see that each morphism θ_k is tracked by the term T^k appearing in the proof of Proposition 7.4.7, and that the identity on $(N_\perp)^N$ is tracked by T. But $\mathrm{BT}(T) = \bigcup_k \mathrm{BT}(T_k)$, and the lemma follows easily by the continuity of application. $\Box$

Theorem 7.4.10 *The interpretation of* PCF$_V$ *in* $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ *is fully abstract at type 3: given closed terms $M, N : ((\iota \to \iota) \to \iota) \to \iota$, if $M \approx N$ then $[\![\,M\,]\!]_V = [\![\,N\,]\!]_V$.*

Proof Suppose $[\![\,M\,]\!]_V \neq [\![\,N\,]\!]_V$; then there exists $\phi \in |(N_\perp)^N \to N_\perp|$ such that $[\![\,M\,]\!](\phi) \neq [\![\,N\,]\!](\phi)$ in $N_\perp$. We will construct elements $\phi_1 \preceq_\Sigma \phi_2 \preceq_\Sigma \ldots$ of $|(N_\perp)^N \to N_\perp|$ whose Σ-limit is ϕ, and such that each ϕ_k is denotable by some term $P_k : (\iota \to \iota) \to \iota$. This will suffice, since for some k we then have $[\![\,M\,]\!](\phi_k) \neq [\![\,N\,]\!](\phi_k)$ and so $[\![\,MP_k\,]\!] \neq [\![\,NP_k\,]\!]$, whence $M \not\approx N$ via the Adequacy Theorem.

For each $k > 0$ let θ_k be as in the above lemma, and define $\phi_k = \phi \circ \theta_k$. Then by the lemma we clearly have that ϕ is the Σ-limit of the ϕ_k. It remains to show that the ϕ_k are denotable. The informal idea is that for any f the value of $\phi_k(f)$ depends only on the k values of $f(0), \ldots, f(k-1)$, so ϕ_k may be regarded as essentially a function of k arguments of type $N_\perp$; and we already know that all morphisms $(N_\perp)^k \to N_\perp$ are denotable in PCF. More formally, for each k we may define a k-ary term context H_k of PCF_V by

$$H_k[x_0^\iota, \ldots, x_{k-1}^\iota] \;=\; \lambda y^\iota.\,\mathsf{cond}\ (y=0)\ x_0\ (\ldots(\mathsf{cond}\ (y=k-1)\ x_{k-1}\ \Omega^\iota)\ldots))$$

where $=$ implements equality testing. Let $h_k = [\![\, H_k \,]\!] : (N_\perp)^k \to ((N_\perp)^N)_\perp$, and let $j_k = \phi \circ \alpha \circ h_k : (N_\perp)^k \to N_\perp$. Then by Remark 7.4.5(ii) there is a k-ary context J_k of PCF_V such that $[\![\, J_k \,]\!] = j_k$. Now let $P_k \equiv \lambda f^{\iota \to \iota}.\, J_k[f0, \ldots, f(k-1)]$. We claim that $\phi_k = [\![\, P_k \,]\!]$. For given any $f \in |(N_\perp)^N|$, for each $n \in |N|$ we clearly have

$$\theta_k(f)(n) \;=\; (\alpha \circ h_k)(f(0), \ldots, f(k-1))(n) \;=\; \begin{cases} f(n) & \text{if } n < k \\ \perp & \text{if } n \geq k \end{cases}$$

and so $\theta_k(f) = (\alpha \circ h_k)(f(0), \ldots, f(k-1))$ by well-pointedness. Thus

$$\phi_k(f) \;=\; (\phi \circ \alpha \circ h_k)(f(0), \ldots, f(k-1)) \;=\; j_k(f(0), \ldots, f(k-1)) \;=\; [\![\, P_k \,]\!](f),$$

and so by well-pointedness $\phi_k = [\![\, P_k \,]\!]$. This completes the proof. $\Box$

Remark 7.4.11 An easy extension of this argument shows that in fact the interpretation of PCF_V in $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is fully abstract at any type σ of level 3 or less, where $\mathrm{level}(\iota) = 0$ and $\mathrm{level}(\sigma_1 \to \cdots \to \iota) = \max(\mathrm{level}(\sigma_i)) + 1$. By the argument used in the proof of Proposition 7.4.2 the result also extends to the models $\mathbf{Mod}(\Lambda^0/T)$ for all semi-sensible T. Note, however, that we have been unable to obtain the corresponding theorem for call-by-name PCF, since we have a much poorer grasp of the object $(N_\perp)^{N_\perp}$ than of $(N_\perp)^N$.

7.4.2 A strategy for proving universality

We end this chapter with a very informal discussion of a possible strategy for proving Conjecture 7.4.1 and some of the difficulties it raises. In the light of

Theorem 6.4.4 it does not matter which version of sequential PCF we work with—our experience suggests that PCF_V is probably the easiest, since in PCF_N and PCF_L it is harder to determine statically which subterms of a term are actually evaluated. Also it suffices to consider universality for the *pure* types ι_n defined by $\iota_0 = \iota$, $\iota_{n+1} = \iota_n \to \iota$, since it can be shown that any type σ can be "embedded" in a pure type in a suitable sense.

Our basic idea is to try to extend the technique used to prove Theorem 7.4.3 as follows. For each pure type ι_n we construct a PCF_V term $I_n : \iota \to \iota_n$, called an *interpreter* for type n, and try to prove that it has the following property: for any element y of $[\![\, \iota_n \,]\!]_V$ and any realizer $M \in \|y\|$ we have $[\![\, I_n \lceil M \rceil \,]\!] = y$ (where $\lceil M \rceil$ is a Gödel-number for M). This would suffice to establish universality at type ι_n. It is a rather trivial exercise to construct interpreters I_0, I_1 with the above property, and so we concentrate here on trying to prove universality at type 2. If the proof could be made to work in this case, we are optimistic that the same ideas would (in principle) allow us to prove the result for all pure types.

The idea behind the construction of I_2 is similar to that used in Theorem 7.4.3. Given a morphism $\phi : (N_\perp)^N \to N_\perp$, a realizer $M \in \Lambda^0/\mathcal{B}$ for ϕ and a PCF term $F : \iota \to \iota$, the strategy used by to compute $I_2 \lceil M \rceil\, F$ would be as follows: we simulate the leftmost reduction of the untyped term Mz, where z is a formal symbol standing for the function F. If this terminates, we then "decode" the result to give a natural number. The problem is that in doing this we only have access to the "extension" of the function F, i.e. its values on particular arguments—we have no way of accessing a realizer for the morphism $f = [\![\, F \,]\!]$. So if in the course of reducing Mz we are confronted with a subterm zP, we will first need to reduce and decode P itself to yield a natural number p say, then feed this to the function F, re-encode the result as $\lceil \overline{Fp} \rceil$ and resume the leftmost reduction of Mz. Since the subterm P itself may well contain occurrences of z, all this reduction and decoding must of course be done in a highly recursive fashion.

There are several problems with this strategy as we have outlined it. First, the right notion of reduction in Λ^0 is the call-by-name one, whereas the above strategy gives us a very "call-by-value" treatment of subterms of the form zP.

More specifically, if N realizes f and we wish to compute the normal form (if any) of MN, then when confronted with $NP[N/z]$ we will reduce this subterm before reducing $P[N/z]$ itself. However, according to the above strategy, when we confronted with zP we must first reduce P to normal form! So one might be worried that even if the reduction of MN terminates, the reduction of Mz could be side-tracked indefinitely. Fortunately, this problem can be overcome by exploiting the fact that M behaves "extensionally" on codes for f: if $N, N' \in \|f\|$ and $MN \twoheadrightarrow \overline{n}$ then $MN' \twoheadrightarrow \overline{n}$. We use a trick similar to that employed in the proof of Proposition 7.4.7: let N be any realizer for f and take $N' = \lambda x.\, xii\,(Nx)$. Then clearly N' also realizes f, but N' is always forced to "evaluate its argument": the reduction of $N'P$ begins (essentially) with a reduction of P to normal form. By keeping track of the lengths of the various reduction sequences, one can show that (for otherwise well-behaved M) if $MN' \twoheadrightarrow \overline{n}$ under call-by-name reduction then $Mz \twoheadrightarrow \overline{n}$ under the strategy outlined above.

A second problem is that if we are trying to reduce a term M' in which z occurs to the left of any β-redexes, there is no guarantee that this occurrence of z will be applied to a well-behaved term P. Indeed, it need not be applied to anything, as in the term $M' = \lambda x.z$. Alternatively, if the subterm zP occurs underneath an abstraction, then P might contain extraneous free variables, as in the term $M' = \lambda x.zx$. However, it seems that both of these rather annoying situations can be disposed of via technical arguments. The most interesting (and worst) case is when P has no free variables other than z, but for some (or all) realizers $N \in \|f\|$ the term that $P[N/z]$ is not a numeral, i.e. we do not have $P[N/z] \twoheadrightarrow \overline{p}$ for any p.

We may respond to this problem by appealing again to the "extensional" behaviour of M. Suppose that we are confronted with a subterm zP such that $P[N/z]$ is not a numeral for some $N \in \|f\|$; suppose moreover that this is the first time in the reduction of Mz that this particular problem has arisen. One idea is to construct N' that agrees with N on all *bona fide* numerals $\overline{p}$, but such that $N'P$ "diverges". So our simulation of the reduction of Mz will diverge because we will fail to reduce P to a numeral; but this will be the right thing to do anyway, since the reduction of MN' in Λ^0 diverges and so $\phi(f) = \bot$. By using (and extending) the Böhm tree "separation" technology in [8, Section 10.4], it turns out that we can do exactly this kind of thing: given any term $Q \in \Lambda^0$ such that $Q \not\approx_\eta \overline{p}$ for any p, we can indeed construct N' such that $N'\overline{p} = N\overline{p}$ for all p but $N'Q = \Omega$.

We seem to be almost home and dry. However there remain two subtle problems. The first of these concerns the gap between the theories $\mathcal{B}$ and $\mathcal{K}^*$: we have not said what we do if $Q \approx_\eta \overline{p}$ but $\mathcal{B} \not\vdash Q = \overline{p}$. This subtlety spawns another host of technical difficulties which we do not discuss here; but we believe that these can be overcome. The second problem seems at first sight to be very slight. Suppose as above we have P such that $P[N/z]$ is a numeral, and we wish to construct N' such that $N'P$ diverges. Using our separation techniques we can obtain N' such that $N'P[N/z] = \Omega$ in $\mathcal{B}$; however, what we really need is that $N'P[N'/z] = \Omega$. In fact there is an infinite supply of realizers N' such that $N'P[N/z] = \Omega$, and for each of these there is an infinite supply of realizers N'' such that $N''P[N'/z] = \Omega$, so it seems almost inconceivable that there is *no* $N' \in \|f\|$ such that $N'P[N'/z] = \Omega$! However, we have been unable to find any way of constructing such an N', and appear to have reached an impasse.

We therefore propose the following conjecture as a "challenge problem". Although the conjecture itself is rather weaker than the property we would need to solve our difficulties, it is easy to present as a simple self-contained problem, and it seems likely that a proof of it would provide the key to solving the problem described above.

Conjecture 7.4.12 *Let $\mathcal{S} = \{N \in \Lambda^0 \mid Ni =_\beta i\}$, and suppose $P \in \Lambda^0(z)$ is such that $NP[N/z] =_\beta i$ for all $N \in \mathcal{S}$. Then $P[N/z] \approx_\eta i$ for all $N \in \mathcal{S}$.*

We believe that if one could prove this kind of assertion, one would be able to complete the whole of the proof of universality at type 2 outlined above. But even if this were possible, the technical difficulties involved in extending this result to all pure types (or even to type 3) would probably still be appalling!

One might ask whether the problem could be made easier by choosing a different term model in place of $\Lambda^0/\mathcal{B}$, for instance a call-by-value term model. So far we have not found a model for which the problem seems to be any easier, though some of the difficulties that arise for other models seem to be different.

One final remark: it is also natural to ask what hierarchy of *total* functionals over N arises from $\Lambda^0/\mathcal{B}$ (or other term models). Nobody seems to know the answer to this question (cf. [10, VI.8]), although Martin Hyland has suggested that the answer is probably *not* the HEOs, as it seems that operations definable in the λ-calculus would be likely to have strong continuity properties, whereas not all the HEOs are continuous (see e.g. [30]).

Chapter 8

Program logics

Traditionally, one of the main motivations for denotational semantics has been to gain insight into the logic of programming languages, and to provide conceptual and mathematical tools for reasoning about the languages themselves and about programs written in them. We end the main body of this thesis with a brief discussion of how our models can be used to give logics for reasoning about programs in PCF-like languages. This chapter is an account of "work in progress" and the material is not in a definitive state, so rather than attempt a detailed mathematical exposition we content ourselves with a rough outline of the main ideas. Nevertheless, we hope that the ideas sketched here will be of some help in explaining the motivations behind the work in previous chapters.

Denotational semantics can be used to establish relationships between a programming language $\mathcal{L}$ and a logic $\mathbf{J}$. For instance, by giving interpretations of both $\mathcal{L}$ and $\mathbf{J}$ in some common mathematical structure $\mathcal{M}$, we may be able to show that if $\vdash e \Downarrow$ is provable in $\mathbf{J}$ then e really does terminate. Typically, then, certain simple formulae of the logic (such as $e \Downarrow$) will have not only a denotational interpretation in $\mathcal{M}$ but also an interpretation as a statement about programs in the real world (such as the statement that e terminates). Often this situation breaks down for more complicated formulae—for example, there may be no simple "real-world" explanation of the meaning of the quantifiers. However, here we are interested in finding program logics for which we do have such a naive real-world interpretation: in other words, logics in which every sentence has a "meaning" that can be explained in terms of the programming language itself, without reference to the model $\mathcal{M}$. We might call this the *operational* interpretation of φ, in contrast to its denotational interpretation in $\mathcal{M}$. Our contention is that a logic will be more easily grasped and used by a non-specialist if it has a good operational interpretation.

Given a logic $\mathbf{J}$ with both an operational and a denotational interpretation, it is natural to ask whether these "agree", i.e. whether a sentence is satisfied by the model $\mathcal{M}$ precisely when it is true under the operational interpretation. In this kind of situation, we will say that the model $\mathcal{M}$ is *logically fully abstract* for $\mathcal{L}$ and $\mathbf{J}$. A logically fully abstract model can be used to show that the operational interpretation of the logic is legitimate: if a sentence φ is provable in the logic then φ expresses a true fact about the real world. Notice that if $\mathbf{J}$ is a logic whose formulae are just equations between terms of $\mathcal{L}$, then logical full abstraction corresponds to ordinary (equational) full abstraction if we take the operational interpretation of these equations to be "observational equivalence". Thus logical full abstraction may be seen as generalizing the ordinary notion. Both the concept and the name "logical full abstraction" are due to Gordon Plotkin.

In this chapter we will investigate the idea of logical full abstraction for two different kinds of program logic. In Section 8.1 we consider a simple *classical* logic $\mathbf{K}(\mathcal{L})$ where $\mathcal{L}$ is a PCF-like language. This is the kind of logic we might expect to be useful in practice for specifying and proving interesting properties of programs. We give a very naive operational interpretation of sentences of this logic, and show how a denotational interpretation can be given in any PCF-model $\mathcal{C}$ that supports $\mathcal{L}$. In this case we show that $\mathcal{C}$ is logically fully abstract provided it is a *universal* model for $\mathcal{L}$; hence some of the realizability models discussed in Chapter 7 provide examples of logically fully abstract models. The material in this section is in a reasonably satisfactory state, although there are some details still to be filled in.

The material in Section 8.2 is much less well-developed. We consider a simple *constructive* logic $\mathbf{J}(\mathcal{L})$. This logic has a well-known denotational interpretation as the internal logic of a topos, and so it is natural to ask whether one can give an operational interpretation to match this. We tentatively suggest an operational interpretation based on *typed realizability*, and hence define a notion of "logical full abstraction" for $\mathbf{J}(\mathcal{L})$. This turns out to be a very strong condition indeed, and in fact we cannot show that it holds for any of the models considered in Chapter 7 (though some come closer than others). However, by modifying the definition of one of these models slightly we obtain an example of a logically fully abstract model for $\mathbf{J}(\mathrm{PCF}_N^{++})$. All the material in this section is exploratory in nature and we do not work out any of the details; thus this section may be read as a prospectus for future work.

8.1 Classical logic

We start by considering a simple classical logic for reasoning about programs. This logic has many features in common with LCF [34], and in other respects resembles Extended ML (see [47]). We believe that this is the kind of logic that would in principle be useful for specifying and proving the kinds of properties of programs that programmers are interested in. (We do not claim, however, that the *particular* logic presented here is especially practical or useful—our aim has been to make our logic as simple as possible for the purposes of illustrating the ideas. There are many delicate decisions involved in the design of a logic that may affect its usability, and it is not our purpose to discuss these here.)

For each of our languages $\mathcal{L}$, we can define the syntax of a many-sorted logic $\mathbf{K}(\mathcal{L})$ as follows. The *sorts* of $\mathbf{K}(\mathcal{L})$ are precisely the types of $\mathcal{L}$; expressions of sort σ in $\mathbf{K}(\mathcal{L})$ are precisely terms of type σ in $\mathcal{L}$; and logical variables of sort σ are just term variables of type σ. The syntactic class of *formulae* of $\mathbf{K}(\mathcal{L})$ is distinct from that of expressions; we use φ, ψ to range over formulae and $M, N, P, \ldots$ to range over expressions. The syntax of formulae of $\mathbf{K}(\mathcal{L})$ is as follows:

$$\begin{aligned} \varphi \ ::= \ & \mathit{false} \mid M \simeq N \mid M \Downarrow \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\ & \mid \varphi_1 \Rightarrow \varphi_2 \mid \forall x^\sigma.\varphi_1 \mid \exists x^\sigma.\varphi_1. \end{aligned}$$

(Note that $\mathbf{K}(\mathcal{L})$ is really a many-sorted *first-order* logic rather than a higher-order logic—we have a separate ground sort for each type σ.)

Informally, our intention is that $\simeq$ denotes Kleene equality— if either expression terminates then so does the other and they are equal in some sense—and that both free and bound variables range over expressions definable in $\mathcal{L}$, terminating or not. (We will make these intentions more explicit when we give the semantics of $\mathbf{K}(\mathcal{L})$ below.) Many other ways of treating non-terminating expressions in a logic are possible (see [16] for a good survey and discussion)— the choice we have made here is in some sense maximally expressive. For practical convenience we may wish to enrich the syntax with derived constructs such as the following:

$$\begin{array}{rcl}
(M = N) & \equiv & (M \Downarrow) \wedge (M \simeq N) \\
\neg\varphi & \equiv & \varphi \Rightarrow \mathit{false} \\
\mathit{true} & \equiv & \neg\mathit{false} \\
\forall' x^\sigma.\varphi & \equiv & \forall x^\sigma.(x \Downarrow) \Rightarrow \varphi \\
\exists' x^\sigma.\varphi & \equiv & \exists x^\sigma.(x \Downarrow) \wedge \varphi
\end{array}$$

The axioms and inference rules of $\mathbf{K}(\mathcal{L})$ are essentially those of classical first-order logic. For simplicity we consider here a Hilbert-style presentation of the logic: all judgements are of the form $\vdash \varphi$. The axioms of are taken to be all tautologies of classical propositional logic, plus all judgements of the following forms:

$$\begin{array}{ll}
\vdash M \simeq M & \vdash \varphi[M/x] \wedge (M \simeq N) \Rightarrow \varphi[N/x] \\
\vdash (M \simeq N) \Rightarrow (N \simeq M) & \vdash (\forall x.\varphi) \Rightarrow \varphi[M/x] \\
\vdash (M \simeq N) \wedge (N \simeq P) \Rightarrow (M \simeq P) & \vdash \varphi[M/x] \Rightarrow (\exists x.\varphi).
\end{array}$$

The rules of inference of $\mathbf{K}(\mathcal{L})$ are taken to be

$$\frac{\vdash \varphi \Rightarrow \psi \qquad \vdash \varphi}{\vdash \psi} \qquad \frac{\vdash \varphi \Rightarrow \psi \qquad x \notin \mathrm{FV}(\varphi)}{\vdash \varphi \Rightarrow (\forall x.\psi)} \qquad \frac{\vdash \psi \Rightarrow \varphi \qquad x \notin \mathrm{FV}(\varphi)}{\vdash (\exists x.\psi) \Rightarrow \varphi}$$

We now give a very naive *operational interpretation* of $\mathbf{K}(\mathcal{L})$—essentially, we give a way of translating formulae of $\mathbf{K}(\mathcal{L})$ into English statements about programs of $\mathcal{L}$. First, we need to clarify what we mean by *termination*. For lazy and call-by-value languages, we will say that a closed term M of $\mathcal{L}$ terminates iff $M \Downarrow V$ for some V. For call-by-name languages, however, things are more subtle, since officially we are not allowed to observe termination at higher types. For the purposes of this chapter, then, we adopt the convention that *all* closed terms of higher type terminate. More precisely, we will say that a closed term $M : \sigma$ *terminates* iff either $M \Downarrow V$ for some V or $\sigma \neq \iota$.

Having established this piece of terminology, we can now give a rather Tarskian definition of our operational interpretation of $\mathbf{K}(\mathcal{L})$:

Definition 8.1.1 (Operational truth) *We define a relation $\models_{op} \varphi$ (read "φ is operationally true") on closed formulae φ of $\mathbf{K}(\mathcal{L})$ as follows.*

- $\models_{op}$ *false doesn't hold;*
- $\models_{op} (M \simeq N)$ *iff M, N are observationally equivalent;*
- $\models_{op} (M \Downarrow)$ *iff M terminates;*
- $\models_{op} \varphi \wedge \psi$ *iff* $\models_{op} \varphi$ *and* $\models_{op} \psi$*;*
- $\models_{op} \varphi \vee \psi$ *iff* $\models_{op} \varphi$ *or* $\models_{op} \psi$*;*
- $\models_{op} \varphi \Rightarrow \psi$ *iff either* $\not\models_{op} \varphi$ *or* $\models_{op} \psi$*;*
- $\models_{op} \forall x^\sigma.\varphi$ *iff* $\models_{op} \varphi[M/x]$ *for all closed $M : \sigma$;*
- $\models_{op} \exists x^\sigma.\varphi$ *iff* $\models_{op} \varphi[M/x]$ *for some closed $M : \sigma$.*

We may extend the relation $\models_{op}$ to open formulae as follows: if φ is an open formula with free variables $x_1, \ldots, x_n$, then $\models_{op} \varphi$ iff $\models_{op} \varphi[\vec{M}/\vec{x}]$ for all closed terms $M_1, \ldots, M_n$ of the appropriate types.

(We apologize for the slight mismatch between the use of the symbol $\Downarrow$ here and that in Chapter 6!) The above definition extends in a natural way to the derived constructs: for instance, we have $\models_{op} (M = N)$ iff M, N are observationally equivalent *terminating* expressions. Notice that the notion of operational truth only requires concepts relating to the language $\mathcal{L}$ itself—there is no reference to a denotational semantics. We are thus hopeful that this interpretation of the logic would be readily understood by programmers.

Next we give a simple way of interpreting $\mathbf{K}(\mathcal{L})$ denotationally. Let $\mathcal{C}$ be any PCF-model supporting $\mathcal{L}$. For each σ let $S^\sigma_{\mathcal{L}} = \mathrm{Hom}(1, C^\sigma_{\mathcal{L}})$. The idea is to interpret the sort σ by the set S^σ, and a formula with free variables $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ by a subset of $S^{\sigma_1} \times \cdots \times S^{\sigma_n}$. To do this, we need subsets $T^\sigma \subseteq S^\sigma$ corresponding to the "terminating" elements. We first define subobjects $D^\sigma \rightarrowtail C^\sigma$ as follows:

$$\begin{array}{llllll} D_N^\iota & = N, & D_L^\iota & = N, & D_V^\iota & = N, \\ D_N^{\sigma\to\tau} & = V_N^{\sigma\to\tau}, & D_L^{\sigma\to\tau} & = (V_L^\sigma \to V_L^\tau), & D_V^{\sigma\to\tau} & = V_V^{\sigma\to\tau}. \end{array}$$

For each σ and $\mathcal{L}$ there is an evident morphism $D_\mathcal{L}^\sigma \rightarrowtail C_\mathcal{L}^\sigma$. We then define $T_\mathcal{L}^\sigma = \mathrm{Hom}(1, D_\mathcal{L}^\sigma)$; we regard T^σ as a subset of S^σ.

Definition 8.1.2 (External validity) *For an environment $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\rangle$, define $[\![\,\Gamma\,]\!]_K = S^{\sigma_1} \times \cdots \times S^{\sigma_n}$. For any term $M : \tau$ in environment Γ, let $[\![\,M\,]\!]_K^\Gamma : [\![\,\Gamma\,]\!]_K \to S^\tau$ be the set-theoretic function determined by composition with $[M]^\Gamma$. For any formula φ of $\mathbf{K}(\mathcal{L})$ with $\mathrm{FV}(\varphi) \subseteq \Gamma$, we define the set $[\![\,\varphi\,]\!]_K^\Gamma \subseteq [\![\,\Gamma\,]\!]_K$ as follows:*

$$\begin{array}{lll} z \in [\![\,\mathit{false}\,]\!]_K^\Gamma & \textit{never;} & \\ z \in [\![\,M \simeq N\,]\!]_K^\Gamma & \textit{iff} & [\![\,M\,]\!]_K^\Gamma(z) = [\![\,N\,]\!]_K^\Gamma(z); \\ z \in [\![\,M\Downarrow\,]\!]_K^\Gamma & \textit{iff} & [\![\,M\,]\!]_K^\Gamma(z) \in T^\tau; \\ z \in [\![\,\varphi \wedge \psi\,]\!]_K^\Gamma & \textit{iff} & z \in [\![\,\varphi\,]\!]_K^\Gamma \textit{ and } z \in [\![\,\psi\,]\!]_K^\Gamma; \\ z \in [\![\,\varphi \vee \psi\,]\!]_K^\Gamma & \textit{iff} & z \in [\![\,\varphi\,]\!]_K^\Gamma \textit{ or } z \in [\![\,\psi\,]\!]_K^\Gamma; \\ z \in [\![\,\varphi \Rightarrow \psi\,]\!]_K^\Gamma & \textit{iff} & z \notin [\![\,\varphi\,]\!]_K^\Gamma \textit{ or } z \in [\![\,\psi\,]\!]_K^\Gamma; \\ z \in [\![\,\forall x^\sigma.\varphi\,]\!]_K^\Gamma & \textit{iff} & (z,w) \in [\![\,\varphi\,]\!]_K^{\Gamma,x^\sigma} \textit{ for all } w \in S^\sigma; \\ z \in [\![\,\exists x^\sigma.\varphi\,]\!]_K^\Gamma & \textit{iff} & (z,w) \in [\![\,\varphi\,]\!]_K^{\Gamma,x^\sigma} \textit{ for some } w \in S^\sigma. \end{array}$$

We say that φ is externally valid *in $\mathcal{C}$, and write $\mathcal{C} \models_K \varphi$, if $[\![\,\varphi\,]\!]_K^\Gamma = [\![\,\Gamma\,]\!]_K$ where $\mathrm{FV}(\varphi) \subseteq \Gamma$. In particular, if φ is closed then $\mathcal{C} \models_K \varphi$ iff $[\![\,\varphi\,]\!]_K$ is inhabited.*

It is trivial to check that the axioms and inference rules of $\mathbf{K}(\mathcal{L})$ are sound with respect to this interpretation. Note that by considering only global elements we are throwing away most of the interesting categorical structure of $\mathcal{C}$. In particular, the functor Γ drastically fails to preserve exponentials, and so our interpretation of sorts as sets is no longer "compositional".

Now that we have given both an operational and a denotational interpretation of $\mathbf{K}(\mathcal{L})$, there is a natural notion of logical full abstraction:

Definition 8.1.3 *The interpretation of a language $\mathcal{L}$ in a PCF-model $\mathcal{C}$ is* logically fully abstract for $\mathbf{K}(\mathcal{L})$ *(or* classically l.f.a.*) if, for every formula φ of $\mathbf{K}(\mathcal{L})$, $\mathcal{C} \models_K \varphi$ iff $\models_{op} \varphi$.*

Theorem 8.1.4 *Suppose $\mathcal{C}$ is a well-pointed PCF-model supporting $\mathcal{L}$. If the interpretation of $\mathcal{L}$ in $\mathcal{C}$ is universal, then it is logically fully abstract for $\mathbf{K}(\mathcal{L})$.*

Proof Suppose $\mathcal{C}$ is a universal model for $\mathcal{L}$; then it is also equationally fully abstract by Proposition 6.4.3. We first show by induction that $\mathcal{C} \models_K \varphi$ iff $\models_{op} \varphi$ for all *closed* formulae φ. For formulae of the form $M \simeq N$, clearly $M \approx N$ iff $[\![M]\!] = [\![N]\!]$ by full abstraction; hence $\models_{op} (M \simeq N)$ iff $\mathcal{C} \models_K (M \simeq N)$. For formulae of the form $M \Downarrow$, note that M terminates iff $[\![M]\!] \in T^\sigma$ by adequacy; hence $\models_{op} (M \Downarrow)$ iff $\mathcal{C} \models_K (M \Downarrow)$. The cases for the propositional connectives are all trivial. For the quantifiers, suppose first that $\mathcal{C} \models_K \forall x^\sigma.\varphi$. Then for any closed $M : \sigma$ we have $[\![M]\!] \in S^\sigma$, and so $\mathcal{C} \models_K \varphi[M/x]$ by the definition of $[\![\forall x.\varphi]\!]_K$; hence $\models_{op} \varphi[M/x]$ by the induction hypothesis. Thus $\models_{op} \forall x^\sigma.\varphi$. Conversely, suppose $\models_{op} \forall x^\sigma.\varphi$. For any $w \in S^\sigma$, by universality we have $w = [\![M]\!]$ for some closed $M : \sigma$. But we have $\models_{op} \varphi[M/x]$ and so $\mathcal{C} \models_K \varphi[M/x]$ by the induction hypothesis; hence $w \in [\![\varphi]\!]^x_K$. Thus $[\![\varphi]\!]^x_K = S^\sigma$, so $\mathcal{C} \models \forall x^\sigma.\varphi$. The case for $\exists$ is similar; this completes the proof for closed formulae. To see that the result extends to open formulae, it is enough to observe that if $\forall \vec{x}.\varphi$ is the universal closure of φ then $\models_{op} \varphi$ iff $\models_{op} \forall \vec{x}.\varphi$, and $\mathcal{C} \models \varphi$ iff $\mathcal{C} \models \forall \vec{x}.\varphi$. $\square$

Examples 8.1.5 By the results of Chapter 7, the models $\mathbf{Mod}(K_1)$, $\mathbf{Mod}(\mathcal{P}\omega_{re})$ are logically fully abstract for (any version of) $\mathbf{K}(\mathrm{PCF}^{++})$. If Conjecture 7.4.1 is true then the model $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is logically fully abstract for $\mathbf{K}(\mathrm{PCF})$. From the results in Section 7.4 we can see that the $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is logically fully abstract at least for a significant fragment of $\mathbf{K}(\mathrm{PCF}_V)$, namely that in which all equations are at type level 3 or less, and all quantifications are at type level 1 or less.

It seems that the converse of Theorem 8.1.4 also holds: any logically fully abstract model is universal. To prove this, one could define an effective Gödel-numbering $\ulcorner - \urcorner$ of PCF terms as natural numbers, and for each type σ construct an "interpreter" $K^\sigma : \iota \to \sigma$ such that $K^\sigma\lceil M\rceil \approx M$ for all closed $M : \sigma$. We would then have $\models_{op} \forall x^\sigma . \exists n^\sigma . \, x = K^\sigma n$; hence if the model $\mathcal{C}$ were logically fully abstract we would have that every element $z \in S^\sigma$ was denoted by a term $K^\sigma n$. In general, the construction of suitable terms K^σ seems harder than one might expect, and we have not worked out the details. However, in the case of PCF^{++} we have a cheap proof that such terms exist:

Proposition 8.1.6 *Suppose $\mathcal{L}$ is a version of* PCF^{++}, *and $\mathcal{C}$ is a logically fully abstract model of $\mathcal{L}$. Then $\mathcal{C}$ is universal for $\mathcal{L}$.*

Proof (Sketch.) We do the proof for PCF$_V^{++}$; the proof is easily adapted to work for the other variants. Let $G : \mathbf{EDom} \to \mathbf{Mod}(K_1)$ be as in Section 7.2. An easy exercise in recursion theory shows that each object GD is isomorphic to some object X such that $\bigcup_{x \in |X|} \|x\| = \mathrm{N}$ (cf. [31, Lemma 1.16]). For each σ, let $X^\sigma \cong C_V^\sigma$ be some object with this property; then there is a morphism $N \to X^\sigma$ tracked by i that is surjective on the underlying sets. But $\mathbf{Mod}(K_1)$ is universal for PCF$_V^{++}$, so there exists a term $K^\sigma : \iota \to \sigma$ that denotes the corresponding morphism $N \to C_V^\sigma$. The rest of the proof proceeds as outlined above. $\square$

The axioms and rules of logic $\mathbf{K}(\mathcal{L})$ as we have presented it only allow us to deduce tautologies of predicate logic—it does not allow us to prove any interesting properties of programs. To turn $\mathbf{K}(\mathcal{L})$ into a useful program logic, we would have to add further axioms that gave information about the values of terms of $\mathcal{L}$. For instance, we could "axiomatize" the fixed point operator by including as axioms all judgements of the forms

$$\vdash \mu x^\sigma . M \simeq M[\mu x^\sigma . M/x], \qquad \vdash (N \simeq M[N/x]) \Rightarrow ((\mu x^\sigma . M) \sqsubseteq N),$$

where $M \sqsubseteq N$ is syntactic sugar for $\forall f^{\sigma \to \iota} . (fM \Downarrow) \Rightarrow (fN \Downarrow)$. We would then check that all these axioms were true in some universal model $\mathcal{C}$ under our denotational interpretation; Theorem 8.1.4 would then assure us that if we can prove $\vdash \varphi$ from the axioms then φ is operationally true. This would show that our operational interpretation was legitimate. It would be interesting to work out the details of some good sets of axioms for particular languages $\mathcal{L}$. It seems that the logics obtained would be very much in the spirit of LCF, but with some extra "effectivity" principles (such as *Church's Thesis*) arising from the fact that variables range over computable rather than continuous elements.

8.2 Constructive logic

Many people have claimed that an advantage of synthetic domain theory is that one can use the *internal logic* of a topos to reason about domains in a clean and familiar "set-theoretic" way (see e.g. [90,59,63]). Some of these people have also suggested that the internal logic might provide a convenient setting for "naive" reasoning about programs (see e.g. [25]). We are therefore prompted to ask: Can we give a naive operational interpretation of the internal logic, so that the meaning of the logical formulae can be made intelligible to programmers? In this section we discuss this question for the case of realizability models. The ideas here are at present rather tentative, and we do not attempt much more than a broad philosophical overview, omitting all the proofs. (We hope that these will appear elsewhere.)

The internal logic of $\mathbf{RT}(A)$ corresponds to (a generalization of) Kleene's realizability interpretation of constructive logic (see the Introduction). Already this can be seen as having some operational content in that it refers to "computations" in the abstract machine A. However, from a programmer's point of view this kind of operational meaning is not very satisfactory, since it is dependent on the way in which our high-level language $\mathcal{L}$ is implemented in terms of a low-level "machine language"—something that ought to be (and in practice is) hidden from the programmer. We would prefer an interpretation that made reference only to the operational semantics of $\mathcal{L}$. Moreover, if such an interpretation is to agree with the internal logic then it should have some "constructive" content. This leads us to propose the notion of *typed realizability*: a formula will be operationally true iff it is "realized" by a term of the typed language $\mathcal{L}$.

We will sketch the idea in the case that $\mathcal{L}$ is a *call-by-name* language. In fact we will need to extend $\mathcal{L}$ by adding a unit type υ and product types $\sigma \times \tau$; we also add a constant $* : \upsilon$ and term constructors $\langle -, - \rangle$, fst, snd. The term formation and evaluation rules may be given in the obvious way; we write $\mathcal{L}^\times$ for the extended language so obtained. We consider a constructive many-sorted logic $\mathbf{J}(\mathcal{L}^\times)$, whose sorts are the types of $\mathcal{L}^\times$ and whose expressions are the terms of $\mathcal{L}^\times$. The syntax of formulae of $\mathbf{J}(\mathcal{L}^\times)$ is defined exactly as for $\mathbf{K}(\mathcal{L})$. The axioms and inference rules of $\mathbf{J}(\mathcal{L}^\times)$ (if we want them) are exactly like those of $\mathbf{K}(\mathcal{L})$, except that in place of the classical propositional tautologies we just take the intuitionistic ones.

We will define what it means for a term P of $\mathcal{L}^\times$ to realize a sentence φ of $\mathbf{J}(\mathcal{L}^\times)$. To do this, we first associate with each formula φ a type $\rho(\varphi)$; realizers for φ (if any) will then be certain terms $P : \rho(\varphi)$. The type $\rho(\varphi)$ is determined by the syntactic shape of φ as follows:

$$\rho(\mathit{false}) \;=\; \rho(M \simeq N) \;=\; \rho(M \Downarrow) \;=\; \upsilon$$
$$\begin{array}{rclcrcl} \rho(\varphi \wedge \psi) & = & \rho(\varphi) \times \rho(\psi) & \qquad & \rho(\varphi \Rightarrow \psi) & = & \rho(\varphi) \to \rho(\psi) \\ \rho(\exists x^\sigma . \varphi) & = & \sigma \times \rho(\varphi) & & \rho(\forall x^\sigma . \varphi) & = & \sigma \to \rho(\varphi) \end{array}$$

We can now define a relation $P \;\mathbf{r}\; \varphi$ between *closed* formulae φ and closed terms $P : \rho(\varphi)$ of $\mathcal{L}^\times$:

- $P \;\mathbf{r}\; \mathit{false}$ never holds;
- $P \;\mathbf{r}\; (M \simeq N)$ iff $M \approx N$ and $P \Downarrow$;
- $P \;\mathbf{r}\; (M \Downarrow)$ iff M terminates and $P \Downarrow$;
- $P \;\mathbf{r}\; (\varphi \wedge \psi)$ iff $(\mathsf{fst}\ P) \;\mathbf{r}\; \varphi$ and $(\mathsf{snd}\ P) \;\mathbf{r}\; \psi$;
- $P \;\mathbf{r}\; (\varphi \Rightarrow \psi)$ iff $(PQ) \;\mathbf{r}\; \psi$ whenever $Q \;\mathbf{r}\; \varphi$;
- $P \;\mathbf{r}\; (\forall x^\sigma . \varphi)$ iff $(PQ) \;\mathbf{r}\; \varphi[Q/x]$ for all closed $Q : \sigma$;
- $P \;\mathbf{r}\; (\exists x^\sigma . \varphi)$ iff $(\mathsf{snd}\ P) \;\mathbf{r}\; \varphi[\mathsf{fst}\ P/x]$.

We say that a closed formula φ is *typed-realizable* (and write $\models_{tr} \varphi$) if $P \;\mathbf{r}\; \varphi$ for some $P : \rho(\varphi)$. (Note that we have not considered disjunction here: it is not clear what $\rho(\varphi \vee \psi)$ should be since in general $\rho(\varphi)$ and $\rho(\psi)$ are different. The easiest solution seems to be to drop disjunction from the logic and translate $\varphi \vee \psi$ by the formula

$$\exists n^{\iota}.\, (n \Downarrow) \wedge ((n = 0) \Rightarrow \varphi) \wedge (\neg(n = 0) \Rightarrow \psi).$$

Another solution might be by adding *sum types* to $\mathcal{L}^{\times}$.)

As an easy exercise, the reader might like to check that the *axiom of choice* is typed-realizable, i.e. for any two-place formula $\varphi(x^{\sigma}, y^{\tau})$, we have

$$\models_{tr}\ (\forall x^{\sigma}.\exists y^{\tau}.\,\varphi(x,y)) \ \Rightarrow\ (\exists f^{\sigma\to\tau}.\forall x^{\sigma}.\,\varphi(x, fx)).$$

The definition of typed realizability has an obvious affinity with the "propositions as types" idea in type theory, though we have not explored this connection. It is not clear whether the notion of typed realizability is of any practical use—conceivably one could use it as a basis for some kind of extraction of programs from proofs. From our point of view, however, its interest lies in the fact that it is defined using purely "operational" concepts, and makes no reference to a denotational model.

It is also interesting to note that typed realizability is related to the notion of operational truth (Definition 8.1.1) via the well-known *double-negation translation*. More precisely, for any closed formula φ of $\mathbf{K}(\mathcal{L}^{\times})$ we have $\models_{op} \varphi$ iff $\models_{tr} \varphi^{\neg\neg}$, where $\varphi^{\neg\neg}$ is the double-negation translation of φ. This is analogous to a more familiar fact for Kleene realizability: a formula φ of arithmetic is classically true iff $\varphi^{\neg\neg}$ has a realizer in K_1.

How well does the typed realizability interpretation of $\mathbf{J}(\mathcal{L}^{\times})$ agree with its interpretation as the internal logic of a topos? More precisely, can we find a topos $\mathcal{C}$ such that a sentence φ is typed-realizable iff it is internally true in $\mathcal{C}$? In order to address this question, we will first specify how we interpret $\mathbf{J}(\mathcal{L}^{\times})$ internally in $\mathcal{C}$, using standard ideas from categorical logic. In fact, since the logic $\mathbf{J}(\mathcal{L}^{\times})$ contains no impredicativity, we do not even require a topos; we therefore assume only that $\mathcal{C}$ is a PCF-model supporting $\mathcal{L}^{\times}$, and is *regular* and *locally cartesian-closed* (cf. Paragraphs 1.2.2 and 4.1.2). Concretely, this means that if $\mathbf{Mod}(A)$ supports $\mathcal{L}^{\times}$, we can model $\mathbf{J}(\mathcal{L}^{\times})$ entirely within $\mathbf{Mod}(A)$ without reference to the topos $\mathbf{RT}(A)$. (Since the inclusion $\mathbf{Mod}(A) \hookrightarrow \mathbf{RT}(A)$ is locally cartesian-closed and exact, we of course get the same interpretation by working in $\mathbf{Mod}(A)$ as in $\mathbf{RT}(A)$.)

Recall from Chapter 6 that if $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \rangle$ then $[\![\, \Gamma \,]\!] = C_N^{\sigma_1} \times \cdots \times C_N^{\sigma_n}$. A formula φ of $\mathbf{J}(\mathcal{L}^\times)$ whose free variables are among those in Γ will be interpreted by a subobject $[\![\, \varphi \,]\!]_J^\Gamma$ of $[\![\, \Gamma \,]\!]$ (i.e. an isomorphism class of monos with codomain $[\![\, \Gamma \,]\!]$)—we may think of this subobject as the *extension* of the predicate φ. We may define $[\![\, \varphi \,]\!]_J^\Gamma$ as follows:

- $[\![\, M \simeq N \,]\!]_J^\Gamma$ is the equalizer of $[\![\, M \,]\!]^\Gamma, [\![\, N \,]\!]^\Gamma : [\![\, \Gamma \,]\!] \rightrightarrows C^\sigma$.
- $[\![\, M \Downarrow \,]\!]_J^\Gamma$ is the pullback of $D^\sigma \rightarrowtail C^\sigma$ along $[\![\, M \,]\!]^\Gamma$.
- $[\![\, \mathit{false} \,]\!]_J^\Gamma$ may be obtained as $[\![\, 0 \simeq 1 \,]\!]_J^\Gamma$.
- $[\![\, \varphi \wedge \psi \,]\!]_J^\Gamma$ is the intersection (given by pullback) of $[\![\, \varphi \,]\!]_J^\Gamma$ and $[\![\, \psi \,]\!]_J^\Gamma$.
- $[\![\, \varphi \vee \psi \,]\!]_J^\Gamma$ is the subobject given by the image factorization of $[[\![\, \varphi \,]\!]_J^\Gamma, [\![\, \psi \,]\!]_J^\Gamma]$.
- $[\![\, \varphi \Rightarrow \psi \,]\!]_J^\Gamma = \Pi_f(f^*([\![\, \psi \,]\!]_J^\Gamma))$, where $f = [\![\, \varphi \,]\!]_J^\Gamma$.
- $[\![\, \forall x^\sigma . \varphi \,]\!]_J^\Gamma = \Pi_{\pi^\Gamma}([\![\, \varphi \,]\!]_J^{\Gamma, x^\sigma})$, where π^Γ is the projection $[\![\, \Gamma, x^\sigma \,]\!] \to [\![\, \Gamma \,]\!]$.
- $[\![\, \exists x^\sigma . \varphi \,]\!]_J^\Gamma$ is the subobject given by the image factorization of $\pi^\Gamma \circ [\![\, \varphi \,]\!]_J^{\Gamma, x^\sigma}$.

It can be shown that the axioms and inference rules of $\mathbf{J}(\mathcal{L}^\times)$ are sound with respect to this interpretation. We say φ is *internally valid* in $\mathcal{C}$, and write $\mathcal{C} \models_J \varphi$, if the subobject $[\![\, \varphi \,]\!]_J^\Gamma$ is *entire* (i.e. is represented by $\mathrm{id}_{[\![\Gamma]\!]}$). In the case $\mathcal{C} = \mathbf{Mod}(A)$, it can be shown that this interpretation corresponds to (a generalization of) Kleene's realizability: a closed formula φ is internally valid in $\mathbf{Mod}(A)$ iff φ is realizable by an (untyped) element of A. For such models one can also show that the notions of external and internal validity are related via the double-negation translation: for closed formulae φ of $\mathbf{K}(\mathcal{L}^\times)$ we have $\mathcal{C} \models_K \varphi$ iff $\mathcal{C} \models_J \varphi^{\neg\neg}$.

We declare a model $\mathcal{C}$ to be *logically fully abstract for* $\mathbf{J}(\mathcal{L})$, or *constructively l.f.a.*, if for every closed formula φ of $\mathbf{J}(\mathcal{L})$ we have $\mathcal{C} \models_J \varphi$ iff $\models_{tr} \varphi$. This is evidently a very strong condition indeed, so we need to ask whether we have any examples at all of constructively l.f.a. models. From the above remarks it follows that a model is logically fully abstract for $\mathbf{K}(\mathcal{L}^\times)$ iff it is logically fully abstract for the double-negation fragment of $\mathbf{J}(\mathcal{L}^\times)$; hence any constructively l.f.a. model would also be classically l.f.a. This prompts us to look at our two known examples of classically l.f.a. models for $\mathcal{L}^\times$ where $\mathcal{L}$ is PCF_N^{++}, namely $\mathbf{Mod}(K_1)$ and $\mathbf{Mod}(\mathcal{P}\omega_{re})$. (Officially we have shown only that these models are classically l.f.a. for $\mathcal{L}$, but the results extend easily to $\mathcal{L}^\times$.)

It is easy to see that the model $\mathbf{Mod}(K_1)$ is *not* constructively l.f.a. for $\mathcal{L}^\times$. In particular, every instance of the axiom of choice is typed-realizable (see above), though not every instance is internally valid in $\mathbf{Mod}(K_1)$. For example, suppose $\varphi(f^{\iota\to\iota}, e^\iota)$ is a formula expressing "e tracks f"; then we have

$$\mathbf{Mod}(K_1) \models_J \forall f.\exists e.\, \varphi(f,e) \quad \text{but not} \quad \mathbf{Mod}(K_1) \models_J \exists G.\forall f.\, \varphi(f, Gf).$$

Intuitively, this is because although we can (trivially) transform codes for f into realizers for f, we cannot do this in an extensional way—there is no way to extract a *canonical* realizer for each f. This is exactly the same argument as that used in Paragraph 7.3.2 to show that $N_\perp^N$ is not projective in $\mathbf{Mod}(K_1)$.

For $\mathbf{Mod}(\mathcal{P}\omega_{re})$ the situation seems more promising. Here all objects C^σ are projective (see Proposition 7.3.14). In fact every instance of the axiom of choice is internally valid in $\mathbf{Mod}(\mathcal{P}\omega_{re})$, and more generally if $\forall x^\sigma.\psi$ is internally valid then it has an (untyped) realizer that behaves extensionally on elements of C^σ. Thus it is reasonable to ask whether this model is logically fully abstract for $\mathbf{J}(\mathcal{L}^\times)$. (In a talk given at the CLiCS II workshop in Paris I claimed this was true.) Curiously, this result holds for the fragment of $\mathbf{J}(\mathcal{L})$ without implication, though not for the whole of $\mathbf{J}(\mathcal{L})$. A counterexample is the following:

$$\forall x^\upsilon y^\upsilon.\, \neg(x\Downarrow \wedge\, y \Downarrow) \;\Rightarrow\; \exists n^\iota.\, (x \Downarrow \Rightarrow n = 0) \wedge (y \Downarrow \Rightarrow n = 1).$$

This formula is internally valid in $\mathbf{Mod}(\mathcal{P}\omega_{re})$. To see this, let S be the regular subobject of $\Sigma \times \Sigma$ determined by the predicate $\neg(x \Downarrow \wedge\, y \Downarrow)$. Intuitively, given a realizer for an element of S we can obtain a realizer for a suitable element n of $N_\perp$. In fact S is projective, so we even obtain a morphism $S \to N_\perp$. However, the above formula is *not* typed-realizable. For if P were a typed realizer and Q

were any term of type $\rho(\neg(x \Downarrow \wedge y \Downarrow))$, then $PXYQ$ would yield an element of $N_\perp$ for all $X, Y : \upsilon$, *whether or not* these satisfied $\neg(X \Downarrow \wedge Y \Downarrow)$. But there is no monotone way to extend the above morphism $S \to N_\perp$ to the whole of $\Sigma \times \Sigma$.

It seems that we can resolve this difficulty by switching to a slightly different model, one based on Kreisel's *modified realizability*. The idea behind Kreisel's notion is that for each formula φ we have not only a collection of "actual" realizers R, but also a larger domain of "potential" realizers $P \supseteq R$. The roles of R and P are different: the elements of R behave much like Kleene realizers and provide "computational evidence" for φ, whereas P is determined by the syntactic shape of φ and embodies "type information" about realizers. The modified realizability interpretation of implication is different from the standard one: a realizer for $\varphi \Rightarrow \psi$ must not only map actual realizers for φ to actual realizers for ψ, but also map potential realizers for φ to potential realizers for ψ:

$$R_{\varphi \Rightarrow \psi} \;=\; (R_\varphi \Rightarrow R_\psi) \cap (P_\varphi \Rightarrow P_\psi), \qquad P_{\varphi \Rightarrow \psi} \;=\; (P_\varphi \Rightarrow P_\psi).$$

Thus realizers for implications have a larger domain of definition than in standard realizability. So it seems at least plausible that modified realizability can help us to overcome the above deficiency of the model $\mathbf{Mod}(\mathcal{P}\omega_{re})$.

Fortunately, it appears that we can obtain the result we want without reworking the whole of this thesis for the case of modified realizability! We give here a rough outline of the story—we hope the details will appear elsewhere. For any PCA A, we can define a category $\mathbf{mMod}(A)$ of *modified modest sets* as follows: The objects of $\mathbf{mMod}(A)$ are pairs (X, P_X), where X is an object of $\mathbf{Mod}(A)$ and $P_X \subseteq A$ is such that $\|x\| \subseteq P_X$ for each $x \in |X|$. (We regard P_X as the set of potential realizers for elements of X.) A morphism $(X, P_X) \to (Y, P_Y)$ is a function $f : |X| \to |Y|$ for which there exists $r \in A$ such that r tracks f in the usual sense and additionally $a \in P_X \Rightarrow ra \in P_Y$. (This category in fact sits as a subcategory inside a *modified realizability topos* $\mathbf{mRT}(A)$, though we do not need this here.)

Let us say that a modified modest set (X, P_X) is *tight* if $P_X = \bigcup_{x \in |X|} \|x\|$. The full subcategory of tight objects is clearly isomorphic to $\mathbf{Mod}(A)$, so we have a full and faithful embedding $E : \mathbf{Mod}(A) \to \mathbf{mMod}(A)$, left adjoint to the forgetful functor. It is easy to see that $\mathbf{mMod}(A)$ is cartesian-closed, and that E preserves the cartesian-closed structure. One can also check that if Σ is a dominance in $\mathbf{Mod}(A)$ then $E\Sigma$ is a dominance in $\mathbf{mMod}(A)$ and E commutes with lifting. Moreover, if $(\mathbf{Mod}(A), \Sigma)$ is a PCF-model then so is $(\mathbf{mMod}(A), E\Sigma)$, and E preserves the PCF-model structure. Thus the results of Section 7.3 transfer directly from $\mathbf{Mod}(\mathcal{P}\omega_{re})$ to $\mathbf{mMod}(\mathcal{P}\omega_{re})$: the latter is a universal model for $\mathcal{L} = \mathrm{PCF}_N^{++}$, and every object C_N^σ in $\mathbf{mMod}(\mathcal{P}\omega_{re})$ is isomorphic to one with the "singleton property". Furthermore, $\mathbf{mMod}(\mathcal{P}\omega_{re})$ is locally cartesian-closed and regular, and so we can interpret the logic $\mathbf{J}(\mathcal{L}^\times)$ in $\mathbf{mMod}(\mathcal{P}\omega_{re})$ as above. However, the embedding E crucially does *not* preserve this structure, and so the interpretation of $\mathbf{J}(\mathcal{L}^\times)$ in $\mathbf{mMod}(\mathcal{P}\omega_{re})$ is different from that in $\mathbf{Mod}(\mathcal{P}\omega_{re})$.

We believe that $\mathbf{mMod}(\mathcal{P}\omega_{re})$ is indeed logically fully abstract for $\mathbf{J}(\mathcal{L}^\times)$. The idea of the proof is as follows: one first translates the above definition of the interpretation in $\mathbf{mMod}(\mathcal{P}\omega_{re})$ into an equivalent "concrete" definition involving realizers—this turns out to correspond precisely to Kreisel's modified realizability interpretation. One then checks that the potential (untyped) realizers for any formula φ are precisely the potential realizers for elements of the object $C^{\rho(\varphi)}$—these correspond to the codes for potential *typed* realizers. Hence one can show that the actual untyped realizers for φ correspond precisely to codes for actual typed realizers of φ. Thus φ has a typed realizer iff it has an untyped one. (Although this proof seems to work, we have not yet checked all the details.)

Assuming our result is true, we have a very strong connection indeed between the model $\mathbf{mMod}(\mathcal{P}\omega_{re})$ and the language $(\mathrm{PCF}_N^{++})^\times$. It would be interesting to know whether one could obtain similar results for the call-by-value and lazy variants of this language.

We end with an intriguing question: Is it plausible that the model $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ is constructively l.f.a. for $(\mathrm{PCF}_N)^\times$? We have argued in Section 7.4 that this model is likely to be universal (and hence classically l.f.a.) for PCF_N. It also seems plausible that all the required instances of the axiom of choice are internally valid in $\mathbf{Mod}(\Lambda^0/\mathcal{B})$, even though in general the objects C_N^σ are not projective. Moreover, the formula given above as a counterexample for $\mathbf{Mod}(\mathcal{P}\omega_{re})$ is *not* internally valid in the model $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ (the Böhm tree enthusiast may enjoy checking this). However, these observations are not convincing evidence that constructive l.f.a. holds in general, and on this question I have no strong feelings either way! But it may be that many questions of this kind will remain out of reach until we have a much deeper understanding of the nature of sequential computation.

Chapter 9

Conclusions and further work

We conclude with some philosophical remarks on the material in this thesis, and some suggestions for further research. Many of the opinions expressed in this chapter are quite personal, so it should be read with a degree of scepticism!

9.1 Conclusions

There are several philosophical points we wish to make about the material in this thesis. It seems best to discuss these under various headings.

Notions of computability

Throughout this thesis, our emphasis has been on studying PCAs and realizability models for their *recursion-theoretic* interest. We have regarded PCAs as models for a kind of abstract recursion theory, with different PCAs embodying different "notions of computability". Many aspects of these notions of computability become more clearly visible when we construct the realizability models (for instance, we find that different PCAs give rise to different type structures). Moreover, some of our results on applicative morphisms in Chapters 2 and 3 indicate that the realizability models are "more abstract" than the PCAs, in the sense that non-isomorphic PCAs may give rise to "the same" realizability models. The realizability models thus seem to be the more natural setting for studying the notions of computability, in that they abstract away from uninteresting "coding" details associated with particular PCAs.

The PCF-like typed languages that we have considered can also be thought of as defining notions of computability at higher types. It appears that work in recursion theory at higher types has tended to concentrate more on *total* functionals than on partial ones (see e.g. [49,30]). However, to me it seems more natural to consider the partial functionals—not least because one can give languages with a decidable syntax (such as PCF^{++}) in which all the partial functionals of some natural hierarchy (and only they) are definable. (Contrast this with the fact that there is no recursive enumeration for the total recursive functions even at type 1.) Of course, there are choices to be made in the design of such languages, in particular the choice of evaluation mechanism—however, the results of Chapter 6 show that different choices give rise to languages that embody the same notion of computability in some deeper sense. The main theorems (and conjectures) of Chapter 7 can thus be viewed as recursion-theoretic facts expressing the harmony between certain "typed" and "untyped" notions of computation.

From a computer science point of view, the recursion-theoretic nature of realizability models has the benefit that one can obtain universality and full abstraction results for a variety of languages. Indeed, the fact that we have a choice of universal models for PCF^{++} prompts us to consider even stronger connections between languages and models (for instance, constructive logical full abstraction). We believe that the results of Chapter 8, although rather tentative, indicate that there are some interesting fine distinctions between different notions of computation that do not show up as differences in the corresponding type structures.

There are many other features of realizability toposes that spring from their recursion-theoretic content: for instance, the existence of internally complete subcategories, which allows us to solve domain equations and model strongly polymorphic languages. Although these have not been considered in this thesis, I hope that what I have done illustrates that realizability toposes are attractive objects of study from a recursion-theoretic point of view. It is in the recursion-theoretic aspects of realizability models that I believe their main interest lies.

The "domains as sets" philosophy

We now turn to consider a different (but related) aspect of realizability toposes: the fact that they contain categories of domains as full subcategories, and hence in principle allow one to treat domains simply as "sets". It was first suggested by Scott [90] that because cartesian-closed categories could be embedded in toposes, one could use full higher-order intuitionistic logic for reasoning about the corresponding typed λ-calculi. This idea has been developed further by synthetic domain theory, although here the approach is usually slightly different: we start with a topos and then try to identify a full subcategory of "domains". It is perhaps possible to separate out three distinct claims that are sometimes made in connection with this approach:

- It is intuitive: it allows us to think of a domain simply as a set.
- It saves us work: for instance, we never need to check that a function is continuous, as this is automatic.
- It is powerful: we use the higher-order logic of a topos to reason about domains and elements of them.

The first of these points is a conceptual rather than a technical one; it rests on the idea that a topos can be viewed as a universe of constructive set theory. However, one very soon discovers counter-classical features of synthetic domain theory which betray that we are not dealing with sets in the "usual" sense (for instance, the fact that there is no function interchanging the two points of Σ), and so it seems debatable whether this naive intuition is really helpful. Besides, some people (including myself) are left dissatisfied by an abstract high-level "naive" theory that hides the concrete details of the models—they feel a need to understand what is inside the "black box".

The second point has been strongly advocated by McCarty [59], who cites several examples of "tiresome calculations" (e.g. the check that λ-terms define computable maps) that become superfluous if one believes that all functions are continuous and computable. To me it seems a moot point whether any of these calculations were ever particularly onerous anyway. But in any case, the benefit of saved labour is only enjoyed by theoreticians and semanticists, not by programmers or program verifiers. It seems that in the course of proving properties of programs one would rarely need to check that some function was continuous, once a theoretician had shown that all syntactically definable functions were so.

The last point concerns the internal logic of toposes. It seems clear that in the hands of specialists this is a powerful and effective weapon for proving mathematical facts about the toposes and their internal subcategories. Many of the proofs in [71] and [40], for example, would be much more cumbersome without the use of the internal logic, and it seems that some of our own proofs in Chapter 5 could be simplified by using it. Recently Reus and Streicher [81] have shown how the basic results of synthetic domain theory can be developed axiomatically by working *entirely* in the internal logic. So there does seem to be some substantial evidence for this last claim.

However, while the internal logic may be extremely useful as a tool for *theoretical* investigation, I personally believe that it is wrong to view it as the basis for a usable *program logic*. In Chapter 8 I suggested that a good program logic should not be simply a piece of formal machinery but should have have a clear and intuitive operational interpretation—not least so that the program verifier can understand (and explain) the meaning of the correctness property he has proved! The results in Chapter 8 show that a very simple-minded classical logic can be given a good operational interpretation of the kind that is wanted; and whilst in certain models the internal logic also seems to have an operational interpretation (typed realizability), it is rather esoteric and not obviously useful for proving the correctness of programs. Besides, whatever the *philosophical* arguments in favour of constructive logic may be, there seem to be good *pragmatic* reasons for preferring classical logics: for example, larger fragments of the logic are decidable, and decision algorithms for decidable fragments are often of a lower inherent complexity.

In defence of the internal logic, tt might be argued that one only really needs a good operational interpretation for particularly simple formulae of the logic—for example, for equalities between programs. But even if this is so, I do not know of any practical reason for wanting to use more than the double-negation fragment of the internal logic. (I reached this conclusion after carrying out the correctness proofs for some simple ML programs "in the internal logic"—I then noticed that exactly the same proofs could be much more naturally understood as classical proofs via the notion of operational truth given in Section 8.1.)

I also believe that the results in Chapter 8 illustrate another point: in order to have a good internal logic it is less essential to work in a *topos* than is sometimes assumed. Everything in this thesis from Chapter 4 onwards really takes place within $\mathbf{Ass}(A)$, including the "internal" interpretation of the logic $\mathbf{J}(\mathcal{L}^{\times})$ in Section 8.2. (It seems that for our purposes the structure of a *logos* suffices—see [29].) The topos $\mathbf{RT}(A)$ only plays a background role, and at present we do not have a real use for it from the point of view of programming languages. It would be interesting to know whether for the study of programming languages there is any real advantage in working in $\mathbf{RT}(A)$.

Semantic versus syntactic methods

Next we discuss a question that seems to apply quite generally to much work in denotational semantics. We have seen how our denotational models can be used to obtain program logics for our languages and prove that they are "sound". We end up with a result that can be seen as purely "syntactic": all the judgements provable in some logic are true of the programming language under some operational interpretation. So we should ask: do our *semantic* (i.e. denotational) methods have any significant advantage over purely *syntactic* (i.e. operational) ones? This question seems particularly pertinent to our set-theoretic models of the classical logics $\mathbf{K}(\mathcal{L})$ in Section 8.1, since here the denotation of a type is (up to isomorphism) nothing other than the set of closed terms of that type modulo observational equivalence. Given that such a direct syntactic construction of this model is possible, one may reasonably ask whether one really learns anything new from more roundabout semantic constructions. In particular, could the soundness of our program logic not be proved just as simply by purely operational means?

I do not know the answer to this question, but there are two tentative suggestions I would like to make. The first is that for the simple PCF-like languages we have considered, the honest answer is probably that the semantic methods have no technical advantage over the syntactic ones. Even the validity of logical axioms expressing the minimality property of the fixed point operator could probably be proved operationally without too much effort. But it does seem at least plausible that for larger and more complex programming languages the syntactic approach would be far less feasible, but that the semantic approach would still be tractable and relatively clean. However, it should be stressed that this is speculation on my part—in this thesis we have not even worked out the details of logical axioms for our PCF-like languages!

My second suggestion is that even if there is no *technical* advantage in the use of semantic models, they may still be useful conceptually and methodologically. The fact that certain results *in retrospect* could have been proved syntactically does not mean that they could have been arrived at just as easily by syntactic considerations. Indeed, a consideration of the semantic models could guide us in the design of a good program logic—one for which we knew we would be able to obtain reasonable proof rules.

It would be interesting to know whether our models could shed any useful light on formalisms such as Extended ML [47], the design of which seems to have proceeded along much more syntactic lines. My personal feeling is that it will be very difficult to take an already complicated language definition and "add on" proof rules—in principle it would seem better to take a logic that arises "naturally" from some semantic setting. Of course, the fragment of ML treated in [47] is far more complex than the languages we have considered, and the technology needed for modelling it denotationally cannot yet be said to be fully-fledged!

Limitations of this thesis

We now point out a few of the limitations and shortcomings of the material presented in this thesis.

(1). Our account of realizability toposes has been somewhat one-sided: in particular, we have taken no account of the construction of $\mathbf{RT}(A)$ via *tripos theory* [76,41]. A tripos is essentially an indexed preorder with enough structure to allow us to model higher-order intuitionistic logic; every PCA gives rise to a "realizability tripos", and every tripos gives rise to a topos. Since we know that our applicative morphisms correspond precisely to certain functors between the toposes, it is natural to ask what they correspond to at the level of triposes. For the benefit of the specialist we may simply state the answer here: they correspond to the indexed functors that preserve "regular logic" (i.e. that preserve finite limits in each fibre and commute with the functors $\exists_f$ between fibres). The details may appear elsewhere.

(2). So far we have used the theory of applicative morphisms principally to obtain negative results, in particular the inequivalence of various toposes. It would be nice, however, if one could find some more positive applications, i.e. some good consequences of the *existence* of certain applicative morphisms. It is frustrating (and somewhat surprising) that we do not seem to obtain much interesting information from them about the relationships between the (total or partial) *type structures* in different models. We saw in Proposition 7.4.2 how in certain circumstances the existence of an applicative inclusion can be used to show that two models possess isomorphic type structures, but it seems that we can only obtain results of this kind in rather trivial cases when one of the PCAs is a homomorphic quotient of the other. None of our examples of applicative *retractions* seem to tell us anything interesting at all beyond the mere existence of the corresponding functors.

(3). We have not told the full story about the category of well-complete objects in Chapter 5. In particular, we have not shown that this category is internally complete—we would need to show this in order to justify our claim that it compares favourably with other proposed categories of domains in our models. An account of the details is in preparation.

(4). Finally, we note that with hindsight our exposition would have been cleaner if in Chapter 6 we had included product types and a unit type in the definition of PCF. Not only would this give greater continuity between Section 8.2 and the rest of the thesis, but it would also have allowed us to define the embedding of PCF_L in PCF_N in a more perspicuous way, since we could have taken $\widetilde{\sigma \to \tau}$ to be $\upsilon \times (\tilde{\sigma} \to \tilde{\tau})$ rather than $\iota \to \tilde{\sigma} \to \tilde{\tau}$. This in turn would have greatly simplified the rather messy proofs of the definability lemmas in Section 6.4.

9.2 Directions for further work

Other notions of realizability

In this thesis we have concentrated almost entirely on models based on so-called *standard* realizability—essentially the generalization of Kleene's original notion [48] to an arbitrary PCA. One obvious line of inquiry would be to ask whether the ideas and results can be carried over to other notions of realizability. There is a rather bewildering array of "other realizabilities" that one could explore: a very useful survey is given in Chapter 1 of [68]. Perhaps the most natural one to consider next would be the notion of *modified realizability*—we touched on this in Section 8.2. Modified realizability toposes have already been studied in [68] and [42]. Another direction would be to consider the toposes based on van Oosten's "$\leq$-PCAs" [69], an attractive generalization of ordinary PCAs admitting some interesting examples.

For any of these notions, one can ask the following questions. Can we find a notion corresponding to applicative morphisms and prove a result analogous to Theorem 2.3.7? Can we find good "categories of predomains" in the corresponding toposes—in particular does an idea analogous to our notion of well-complete object work well? Can we obtain universality and full abstraction results for these models analogous to those in Chapter 7? What about constructive logical full abstraction? (In Chapter 8 we briefly discussed the last two of these questions for the case of modified realizability.)

Models for sequential computation

The general problem of understanding the nature of "sequential" computation is the subject of a great deal of current research. This problem is far broader than just the celebrated full abstraction problem for PCF, and would seem to be relevant to the study of all kinds of sequential programming languages. Some very interesting advances have recently been made (see e.g. [3,43,65]), although several open questions remain. It would therefore be very interesting to find realizability models that corresponded well to sequential notions of computation.

In Chapter 7 we conjectured that the category $\mathbf{Mod}(\Lambda^0/\mathcal{B})$ provides a universal model for sequential PCF. Although at present this particular conjecture seems (to me) rather intractable, it seems very likely that the problem could be made much easier by a different choice of PCA. None of the natural "sequential" PCAs that we have considered so far seem to fit the bill, but it does seem that for slightly artificial PCAs the situation is more hopeful. (It might not matter too much if the PCA underlying our model was slightly ugly, since we would be interested mainly in its abstract categorical properties.) It is also conceivable that the problem might be easier for some model based on a "non-standard" notion of realizability.

Once we had constructed such a model for PCF, we would want to ask whether it provided a good setting for modelling sequential computation more generally (whatever that means). It would also be interesting to see how it compared with other known fully abstract models for PCF. The most obvious difference would appear to be that our model would naturally live inside a topos, and this might be an advantage in some respects.

It would also be quite interesting to know whether there are natural examples of realizability models giving partial type structures *in between* those corresponding to PCF and PCF^{++}—in other words, models whose "degree of parallelism" lies between these (see [86]). A very strong conjecture would be that for *any* degree of parallelism there is a realizability model having the corresponding partial type structure.

Other programming language features

The programming languages that we have considered in this thesis have been particularly simple. It is therefore natural to ask what happens when we try to interpret richer languages in our models.

One obvious way of extending our PCF-languages would be to add *recursive types*. We can interpret these in our models using the fact that the category of well-complete objects (or some other category of predomains) is internally complete. It would be interesting to give an interpretation for a language such as Plotkin's FPC in our models and try to obtain an adequacy theorem, perhaps along the lines of [23]. It would also be interesting to try and formulate an appropriate version of the limit-colimit coincidence for our models. On a related note, one could settle the question whether the embeddings of Sections 7.2 and 7.3 preserve initial solutions of domain equations. Finally, one could try to obtain universality and (equational or logical) full abstraction results for recursively-typed languages with respect to particular realizability models. We conjecture that a recursively-typed version of PCF^{++} has a universal interpretation in $\mathbf{Mod}(K_1)$ (some work of Streicher [96] seems to be relevant here).

Another way of extending our programming languages would be to incorporate *polymorphism*. It is known that the existence of internally complete subcategories in our models can be exploited to give interpretations of strongly polymorphic languages. An alternative approach—albeit applicable only to first-order (i.e. "ML-style") polymorphism—is via *polynomial* categories (see [25]). This latter approach seems to have the advantage that we can naturally extend our notion of "operational truth" to polymorphic formulae and maintain a "logical full abstraction" property: a polymorphic formula is taken to be operationally true precisely when all ground instances of it are operationally true. This means we can prove polymorphic properties about polymorphic programs and then instantiate them as required—thus polymorphism gives us reusable proofs as well as reusable programs. It would be nice to work out these ideas in more detail.

One could easily imagine adding more and more language features until one attained the complexity of a "real" programming language. As our languages get larger, one challenging problem is to design good program logics that remain reasonably intuitive and tractable—it is here that we feel a semantic understanding of the languages becomes indispensable. Moreover, the concept of logical full abstraction might be useful in providing a "guiding philosophy" for the design of such logics: we try to find logical constructs for which we can give a simple operational interpretation, and then hope that we can match it by a denotational interpretation in some model. As we have seen, realizability models seem to be good candidates because of their inherently recursion-theoretic nature. And although notions of realizability by no means provide the *only* good source of suitable models, we believe that they should continue to be studied alongside other approaches to denotational semantics.

Bibliography

[1] M. Abadi and G. Plotkin. A PER model of polymorphism and recursive types. In *Proc. of 5th Annual Symposium on Logic in Computer Science*, 1990.

[2] S. Abramsky. The lazy lambda-calculus. In D.L. Turner, editor, *Research Topics in Functional Programming*. Addison-Wesley, 1989.

[3] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF (Extended abstract). In *Proceedings of TACS '94*. Springer LNCS 789, 1994.

[4] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105, 1993.

[5] R. Amadio. Recursion over realizability structures. *Information and Computation*, 91, 1991.

[6] R. Amadio. On the adequacy of PER models. Technical Report 1579, INRIA, 1992.

[7] H.P. Barendregt. The lambda calculus. In J. Barwise, editor, *Handbook of Mathematical Logic*. North-Holland, 1977.

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

[9] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer, 1985.

[10] M. Beeson. *Foundations of Constructive Mathematics*. Springer, 1985.

[11] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3), 1982.

[12] I. Bethke. *Notes on Partial Combinatory Algebras.* PhD thesis, Universiteit van Amsterdam, 1988.

[13] B. Bloom and J.G. Riecke. LCF should be lifted. In *Proc. Conf. Algebraic Methodology and Software Technology.* Dept. Comp. Sci., University of Iowa, 1989.

[14] A. Carboni. Some free constructions in realizability and proof theory. Preprint, 1993.

[15] A. Carboni, P. Freyd, and A. Scedrov. A categorical approach to realizability and polymorphic types. In *Lecture Notes in Computer Science 298.* Springer, 1988.

[16] J.H. Cheng and C.B. Jones. On the usability of logics which handle partial functions. In C. Morgan and J. Woodcock, editors, *Proc. Third Refinement Workshop.* Springer, 1990.

[17] N.J. Cutland. *Computability.* Cambridge University Press, 1980.

[18] L. Egidi, F. Honsell, and S. Ronchi della Rocca. The lazy call-by-value λ-calculus. Preprint.

[19] T. Ehrhard. A categorical semantics of constructions. In *Proc. of 3rd Annual Symposium on Logic in Computer Science*, 1988.

[20] Yu.L. Ershov. Computable functionals of finite type. *Algebra i Logika*, 11(4), 1972.

[21] Yu.L. Ershov. The theory of A-spaces. *Algebra i Logika*, 12(4), 1972.

[22] Yu.L. Ershov. Theorie der Numerierungen I. *Zeit. Math. Logik*, 19, 1973.

[23] M.P. Fiore and G.D. Plotkin. An axiomatisation of computationally adequate domain theoretic models of FPC. In *Proc. of 9th Annual Symposium on Logic in Computer Science*, 1994.

[24] M.P. Fourman. The logic of topoi. In J. Barwise, editor, *Handbook of Mathematical Logic*. North-Holland, 1977.

[25] M.P. Fourman and W.K.-S. Phoa. A proposed categorical semantics for pure ML. Technical Report ECS-LFCS-92-213, Department of Computer Science, University of Edinburgh, 1992.

[26] P. Freyd, P. Mulry, G. Rosolini, and D.S. Scott. Extensional PERs. In *Proc. of 5th Annual Symposium on Logic in Computer Science*, 1990.

[27] P.J. Freyd. Aspects of topoi. *Bull. Austral. Math. Soc.*, 7, 1972.

[28] P.J. Freyd. Algebraically complete categories. In *Category Theory, Proceedings, Como 1990*. Springer LNCS 1488, 1990.

[29] P.J. Freyd and A. Scedrov. *Categories, Allegories*. North-Holland, 1990.

[30] R.O. Gandy and J.M.E. Hyland. Computable and recursively countable functions of higher type. In *Logic Colloquium '76*. North-Holland, 1977.

[31] P. Giannini and G. Longo. Effectively given domains and lambda-calculus models. *Information and Control*, 62, 1984.

[32] J.-Y. Girard. *Interprétation functionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Paris, 1972.

[33] R. Goldblatt. *Topoi: the Categorial Analysis of Logic*. North-Holland, 1979.

[34] M. Gordon, R. Milner, and C. Wadsworth. *LCF: A Mechanised Logic of Computation*. Springer, 1978.

[35] C.A. Gunter. *Semantics of Programming Languages*. MIT Press, 1992.

[36] J.R. Hindley, B. Lercher, and J.P. Seldin. *Introduction to Combinatory Logic*. Cambridge University Press, 1972.

[37] J.M.E. Hyland. Filter spaces and continuous functionals. *Ann. Math. Logic*, 16, 1979.

[38] J.M.E. Hyland. The effective topos. In *The L.E.J. Brouwer Centenary Symposium*. North-Holland, 1982.

[39] J.M.E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40, 1988.

[40] J.M.E. Hyland. First steps in synthetic domain theory. In *Category Theory, Proceedings, Como 1990*. Springer LNM 1488, 1990.

[41] J.M.E. Hyland, P.T. Johnstone, and A.M. Pitts. Tripos theory. *Math. Proc. Camb. Phil. Soc.*, 88, 1980.

[42] J.M.E. Hyland and C.-H. L. Ong. Modified realizability toposes and strong normalization proofs. In J.F. Groote and M. Bezem, editors, *Typed Lambda Calculi and Applications*. Springer LNCS 664, 1993.

[43] J.M.E. Hyland and C.-H. L. Ong. Observational full abstraction for PCF: dialogue games and innocent strategies. Draft, 1994.

[44] J.M.E. Hyland, E.P. Robinson, and G. Rosolini. The discrete objects in the effective topos. *Proc. Lond. Math. Soc.*, 3(60), 1990.

[45] P.T. Johnstone. *Topos Theory*. Academic Press, 1977.

[46] P.T. Johnstone and E.P. Robinson. A note on inequivalence of realizability toposes. *Math. Proc. Camb. Phil. Soc*, 105, 1989.

[47] S. Kahrs, D. Sannella, and A. Tarlecki. The definition of Extended ML. Technical Report ECS-LFCS-94-300, Department of Computer Science, University of Edinburgh, 1994.

[48] S.C. Kleene. On the interpretation of intuitionistic number theory. *J. Symb. Logic*, 10, 1945.

[49] S.C. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. Amer. Math. Soc*, 91, 1959.

[50] S.C. Kleene and R.E. Vesley. *The Foundations of Intuitionistic Mathematics*. North-Holland, 1965.

[51] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In A. Heyting, editor, *Constructivity in Mathematics*. North-Holland, 1959.

[52] J. Lambek and P.J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, 1986.

[53] G. Longo. Set-theoretical models of lambda calculus: Theories, expansions and isomorphisms. *Ann. Pure Appl. Logic*, 24, 1983.

[54] G. Longo and E. Moggi. Cartesian closed categories of enumerations for effective type structures. In G. Kahn, D. MacQueen, and G. Plopkin, editors, *Semantics of Data Types*. Springer, 1984.

[55] G. Longo and E. Moggi. The hereditary partial functionals and recursion theory in higher types. *J. Symb. Logic*, 49(4), 1984.

[56] G. Longo and E. Moggi. Constructive natural deduction and its 'ω-set' interpretation. *Math. Structures in Computer Science*, 1, 1991.

[57] S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

[58] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. Springer, 1992.

[59] D.C. McCarty. Information systems, continuity and realizability. In E. Clarke and D. Cozen, editors, *Logics of Programs, Lecture Notes in Computer Science 164*. Springer, 1984.

[60] D.C. McCarty. *Realizability and Recursive Mathematics.* PhD thesis, Oxford, 1984.

[61] R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4, 1977.

[62] E. Moggi. Partial morphisms in categories of effective objects. *Information and Computation*, 76, 1988.

[63] E. Moggi. Computational lambda calculus and monads. In *Proc. of 5th Annual Symposium on Logic in Computer Science*, 1989.

[64] P.S. Mulry. Generalized Banach-Mazur functionals in the topos of recursive sets. *J. Pure Appl. Algebra*, 26, 1982.

[65] P.W. O'Hearn and J.G. Riecke. Kripke logical relations and PCF. Preprint, 1994.

[66] P.W. O'Hearn and R.D. Tennent. Semantics of local variables. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Categories in Computer Science.* CUP, 1992.

[67] C.-H. L. Ong. *The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming.* PhD thesis, Imperial College, University of London, 1988.

[68] J. van Oosten. *Exercises in realizability.* PhD thesis, Universiteit van Amsterdam, 1991.

[69] J. van Oosten. Extensional realizability. Technical Report ML-93-18, University of Amsterdam, ILLC, 1993.

[70] W.K.-S. Phoa. Relative computability in the effective topos. *Math. Proc. Camb. Phil. Soc*, 106, 1989.

[71] W.K.-S. Phoa. *Domain Theory in Realizability Toposes.* PhD thesis, University of Cambridge, 1990. Available as CST-82-91, Department of Computer Science, University of Edinburgh.

[72] W.K.-S. Phoa. From term models to domains. In *Proc. of Theoretical Aspects of Computer Software, Sendai.* Springer LNCS 526, 1991.

[73] W.K.-S. Phoa. Building domains from graph models. *Math. Structures in Computer Science*, 2, 1992.

[74] W.K.-S. Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. Technical Report ECS-LFCS-92-208, Department of Computer Science, University of Edinburgh, 1992.

[75] W.K.-S. Phoa. Adequacy for untyped translations of typed λ-calculi. In *Proc. of 8th Annual Symposium on Logic in Computer Science*, 1993.

[76] A.M. Pitts. *The theory of triposes.* PhD thesis, University of Cambridge, 1980.

[77] A.M. Pitts. Polymorphism is set-theoretic, constructively. In *Category Theory and Computer Science, Edinburgh 1987.* Springer, 1987.

[78] G.D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1, 1975.

[79] G.D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5, 1977.

[80] G.D. Plotkin. Domains. Technical report, Dept. Comp. Sci., University of Edinburgh, 1983.

[81] B. Reus and T. Streicher. Naive synthetic domain theory—a logical approach. Preprint, 1994.

[82] J.C. Reynolds. Polymorphism is not set-theoretic. In G. Kahn, D. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*. Springer, 1984.

[83] J.G. Riecke. Fully abstract translations between functional languages. *Math. Struct. in Comp. Science*, 3, 1993.

[84] E. Robinson and G. Rosolini. Colimit completions and the effective topos. *J. Symb. Logic*, 55, 1990.

[85] G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, Oxford; Carnegie-Mellon, 1986.

[86] V.Yu. Sazonov. Degrees of parallelism in computations. In *Mathematical Foundations of Computer Science 1976*. Springer LNCS 45, 1976.

[87] H. Schellinx. Isomorphisms and non-isomorphisms of graph models. *J. Symb. Logic*, 56, 1991.

[88] D.S. Scott. Continuous lattices. In F.W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*. Springer, 1972.

[89] D.S. Scott. The language LAMBDA (abstract). *J. Symb. Logic*, 39, 1974.

[90] D.S. Scott. Relating theories of the λ-calculus. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[91] D.S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. In J. Fox, editor, *Computers and Automata*. Polytechnic Institute of Brooklyn Press, 1971.

[92] K. Sieber. Relating full abstraction results for different programming languages. In *Proc. 10th Conference on Foundations of Software Technology and Theoretical Computer Science, Bangalore*. Springer LNCS 472, 1990.

[93] D. Sitaram and M. Felleisen. Reasoning with continuations II: Full abstraction for models of control. In *Conference on Lisp and Functional Programming*. ACM, 1990.

[94] A. Stoughton. Interdefinability of parallel operations in PCF. *Theoretical Computer Science*, 79, 1991.

[95] T. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions*. PhD thesis, University of Passau, 1988.

[96] T. Streicher. A universality theorem for PCF with recursive types, parallel-or and $\exists$. *Math. Struct. in Comp. Science*, 4, 1994.

[97] P. Taylor. The fixed point property in synthetic domain theory. In *Proc. of 6th Annual Symposium on Logic in Computer Science*, 1991.

[98] M.B. Trakhtenbrot. On representation of sequential and parallel functions. In *Proc. 4th Symposium on Mathematical Foundations of Computer Science*. Springer LNCS 32, 1975.

[99] A.S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. Springer, 1973.

[100] A.S. Troelstra. Realizability. Technical Report ML-92-09, University of Amsterdam, ILLC, 1992.