Technical Report

UCAM-CL-TR-247
ISSN 1476-2986

Number 247

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Programming metalogics with a fixpoint type

Roy Luis Crole

February 1992

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted January 1992 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Churchill College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

This University of Cambridge Computer Laboratory Technical Report is a revised version of my Ph.D. thesis. The report is essentially the same as my original thesis which was completed in July 1991, with corrections and alterations as suggested by my Ph.D. examiners,

Dr. J.M.E. Hyland,
Department of Pure Mathematics and Mathematical Statistics,
Cambridge,
England

and

Prof. E. Moggi,
Dip. di Matematica,
Univ. di Genova,
Italy.

I have also made some modifications which I personally think improves the presentation of my original thesis. Any errors which remain are my sole responsibility.

Summary of Contents

A programming metalogic is a formal system into which programming languages can be translated and given meaning. The translation should both reflect the structure of the language and make it easy to prove properties of programs. This thesis develops certain metalogics using techniques of category theory and treats recursion in a new way.

The notion of a category with fixpoint object is defined. Corresponding to this categorical structure there are type theoretic equational rules which will be present in all of the metalogics considered. These rules define the fixpoint type which will allow the interpretation of recursive declarations. With these core notions FIX categories are defined. These are the categorical equivalent of an equational logic which can be viewed as a very basic programming metalogic. Recursion is treated both syntactically and categorically.

The expressive power of the equational logic is increased by embedding it in an intuitionistic predicate calculus, giving rise to the FIX logic. This contains propositions about the evaluation of computations to values and an induction principle which is

derived from the definition of a fixpoint object as an initial algebra. The categorical structure which accompanies the FIX logic is defined, called a FIX hyperdoctrine, and certain existence and disjunction properties of FIX are stated. A particular FIX hyperdoctrine is constructed and used in the proof of the above properties.

PCF-style languages are translated into the FIX logic and computational adequacy results are proved. Two languages are studied: Both are similar to PCF except one has call by value recursive function declarations and the other higher order conditionals.

A dependently typed equational logic containing a fixpoint type and a universal type is given together with its related categorical structure, namely a FIX category with attributes. A representation theorem for Scott predomains is proved, which gives rise to a concrete example of such a FIX category with attributes. Recursive domain equations give rise to endofunctions on the universal type; using the fixpoint type we may solve for fixpoints of such endofunctions and thus obtain a solution of the original domain equation as the type coded by the fixpoint.

Contents

Chapter 0
Categorical Logic in Computer Science

0.1 Introduction

During the late 1960's and early 1970's Scott and Strachey, researchers in the University of Oxford, became concerned with the methods used to define programming languages, or perhaps we should say the *lack* of methods. At the time, the description of a language was essentially operational in nature and to some extent language definitions were virtually synonymous with actual implementations. Very little research concerning underlying mathematical theories of programming had been undertaken. In the mid 1960's, Landin noted connections between the λ calculus, a formalism developed by logicians for representing functions, and certain constructs which appear in programming languages. Landin used the formal theory of the λ calculus to guide the construction of a machine for evaluating programs [Lan64]. It is interesting to note that Landin comments "This paper is a contribution to the 'theory' of the activity of using computers." Indeed, Landin observed that the reduction rules of the λ calculus resemble certain operational reductions of commands and expressions in programming languages and his work was one of the first formal treatments of the theory of *operational* specifications of programming languages.

Scott and Strachey wanted to move away from operational specifications of languages and instead attempt to develop a denotational approach which concentrated more on the intended *meaning* of a language. A simple example of their idea is illustrated by the concept of natural number. Different languages can be used to specify natural numbers, for example octal, decimal, roman, but in each case we are really concerned with what is *denoted* (a natural number) and not *how* the natural number is represented. The first examples of denotational specifications of program fragments were worked out by Scott and Strachey; see [Sco70b], [Sco70a] and [SS71]. In pursuing the notion of mathematical models of programming languages, the question of what constitutes such a model of the λ calculus arose. The λ calculus allows syntactic expressions (which represent functions) to be applied to themselves. It was clear that a mathematical model should interpret such function-representing expressions as some form of set-theoretic function; and to model self application certain sets would have to contain their own function space. Scott realised that such a construction was possible by imposing certain conditions on the kind of function which was allowed in the model; see [Sco69a] and [Sco71]. For further examples of denotational semantics see [Sco82], [Str74] and [SW74].

Once the foundations of operational and denotational semantics had been laid down,

other researchers took up the task of attempting more formal treatments of programming semantics. It became clear that formal semantics would be necessary to ensure that programs really behaved as they were supposed to. The semantics could be used to reason about programs; in particular to prove that a program satisfies its specification. Plotkin investigated different kinds of operational semantics of the λ calculus [Plo75] and also the connections with denotational semantics [Plo77], and went on to clarify many issues which had arisen from both his work and the work of others: see [Plo80] and [Plo81].

As programs became larger and more intricate, the task of proving programs correct became much more difficult. One solution to this problem was the idea of a *programming metalogic* which is a formal logical system in which it is possible to give meaning to other programming languages. The programming metalogic will usually have a very rich type structure and powerful rules for reasoning and will be used to give meaning to a programming language by translating the source code into the metalogic. This translation should preserve the structure of the original language, thereby allowing properties of source programs to be proven by showing the property holds in the metalogic. A proof in the metalogic should be substantially easier than a direct proof using the semantics of the original language.

This thesis presents three such programming metalogics and describes some simple applications. Each of the metalogics deals with recursively defined declarations in a novel way, introducing a new type called the *fixpoint type*; see [CP90a]. The first of these metalogics is a simple equational logic which is an extension of the computational let calculus [Mog89b]; the latter is a formal system for programming semantics which separates the notions of computation and value. The second metalogic is a predicate logic which subsumes the original equational logic via an equality predicate. It is ideally suited to reasoning about languages presented in the natural semantics style [Kah88]. The third metalogic is a dependently typed equational logic containing a type universe: we use this to solve domain equations. The solution of a domain equation can be regarded as a recursively defined type; equivalently we may regard the equation as determining an endofunction on the type universe. We can find a fixpoint of such a function using the fixpoint type and the solution to the domain equation is the type represented by the fixpoint.

Categorical logic is the study of connections between formal logical systems and category theory. As we have remarked, logics can be used to give meaning to programming languages. Category theory can be used to guide the design of such metalogics and also to give a uniform presentation of their semantics. Often it is easier to prove a property of a metalogic by categorical means and then translate the results into statements about the metalogic. The techniques of categorical logic will be used throughout this thesis.

0.2 A Thesis Summary

This thesis divides into three parts.

- **Part I** A simple equational metalogic is described which builds upon the computational let calculus. This system will form a basic core for the metalogics described in Part II and Part III.
- **Part II** The equational metalogic of Part I is strengthened to a predicate metalogic and we give some simple applications.
- **Part III** A dependently typed equational logic containing a type universe is given; this is used to solve domain equations.

Part I

Chapter 1

We begin by giving a brief review of some basic category theory which will be used throughout this thesis, aiming to set up notation for monads, tensorial strengths and let categories. We also provide a few elementary examples of let categories which illustrate their use in computer science. For some background in categorical logic see [Law69] and [MR77].

Chapter 2

We review the computational let calculus and motivate both its uses and also the form in which it is usually presented. This leads to a discussion of suitable extensions and in particular to the fixpoint type: see [NPS90] for some background in type theory. In the presence of a fixpoint type certain endofunctions must have fixpoints. We introduce a categorical definition of the concept of fixpoint type, namely a fixpoint object in a let category. In such categories, morphisms of certain kinds are guaranteed to have fixpoints. This leads to the internal logic of fixpoint objects and from this we describe an extension of the computational λ calculus which, among other things, contains the logic of a fixpoint object. This system is referred to as the $\mathrm{FIX}_{=}$ logic.

Chapter 3

We define FIX categories and prove the usual categorical logic correspondence between such categorical structures and $\mathrm{FIX}_{=}$. We show that FIX categories are, in a precise sense, the most general structures for interpreting $\mathrm{FIX}_{=}$. We discuss fixpoints both in $\mathrm{FIX}_{=}$ and in FIX categories and move on to show that FIX categories are functionally complete. Finally we carry out a gluing construction for let categories which will be used to prove a result about equality of ground terms in $\mathrm{FIX}_{=}$.

Part II

Chapter 4

The language $\mathrm{FIX}_{=}$ captures certain computational features through an equational theory: we increase its expressiveness by embedding $\mathrm{FIX}_{=}$ in a predicate logic called FIX. We define the notion of a FIX theory and show that the pure FIX logical system has a formulation in which some of the rules assume an adjoint form. The chapter finishes with a number of results about FIX, some of which will be put to use later on, together with the statements of three theorems concerning the metalogical properties of FIX.

Chapter 5

We define a FIX hyperdoctrine, which is the categorical counterpart of the FIX logic, and give a concrete example. The next task is to give a formal statement of the categorical logic correspondence for FIX hyperdoctrines and the FIX logic. With a view to using this correspondence to prove the metalogical properties stated in Chapter 4, we define the "logical relations hyperdoctrine;" the proofs make use of its internal logic.

Chapter 6

We investigate how well suited the FIX logic is for analysing two small programming languages. Both of these languages are based on Plotkin's PCF. We give translations of both static and dynamic semantics into the pure FIX logic and prove adequacy results which show that the translations we give preserve the structure and properties of the source languages.

Part III

Chapter 7

We extend the notion of information system to provide a representation theorem for Scott predomains; the latter have properties similar to Scott domains but do not necessarily possess a least element. To effect this, the classical definition of information system is altered in a simple way, giving rise to preinformation systems. We present canonical constructions of products, coproducts and lifting, along with partial function space, in the category of preinformation systems.

Chapter 8

We aim to develop a logic in which there is both a universal type and a fixpoint type. Domain equations can then be solved by considering the endofunctions which will be induced on the universal type. The $\mathrm{FIX}_{=}$ equational logic forms the backbone of

the system introduced here, namely $\mathrm{FIX}^{\mu}_{=}$. We describe the syntax and logical rules of $\mathrm{FIX}^{\mu}_{=}$ which is a dependently typed equational logic and introduce the notion of a $\mathrm{FIX}^{\mu}_{=}$ theory.

Chapter 9

A general categorical structure for dependently typed equational theories is reviewed, namely categories with attributes. We then define a FIX category with attributes; such structures will be used to model $\mathrm{FIX}^{\mu}_{=}$ theories. Having done this, we give a concrete example of a FIX category with attributes and move on to consider the general categorical semantics of $\mathrm{FIX}^{\mu}_{=}$. We finish this chapter with some basic results about such semantics.

Chapter 10

We consider what has been achieved, the prospects for further research and draw our conclusions together.

0.3 Foundation and Notation

Foundations of Category Theory

Throughout this thesis, if we say "let $\mathcal{C}$ be a category" we shall mean that we have specified interpretations of the statements "A is an object of $\mathcal{C}$" and "f is a morphism of $\mathcal{C}$," along with interpretations of "identity morphism," "domain," "codomain" and "composition." There will be no assumptions as to whether the collections of objects or morphisms of $\mathcal{C}$ form a set. We shall only impose restrictions on collections when it is prudent to do so. In such cases, we shall refer to the notions of small and locally small categories as appropriate and think of such concepts within a given model of set theory.

Martin-Löf's Theory of Arities and Expressions

We shall use the theory of arities and expressions due to Martin-Löf to present the object level syntax in this thesis; to do this we make the following definition:

Definition 0.3.1 An *abstract syntax signature* Σ is a pair of sets (GAr, Con) where the elements of GAr are called *ground arities* and the elements of Con are called *constants*. Regarding the ground arities as the ground types for a simply typed λ calculus, we refer to the simple types as *arities*. The constants are assumed to be tagged with an arity.

With these data, we can regard the abstract syntax signature Σ as a signature in the conventional sense for a (type tagged) simply typed λ calculus with constants.

Definition 0.3.2 Given an abstract syntax signature Σ, an *abstract syntax* is the collection of $\alpha\beta\eta$ equivalence classes of terms of the simply typed λ calculus generated from Σ and we shall refer to this calculus as the *meta λ calculus*. We shall call individual classes *expressions* of the abstract syntax and refer to the variables of this simply typed λ calculus as *metavariables*. Abstraction of an expression e will be denoted by $u.e$ where u is a metavariable, substitution for a metavariable by $e[e'/u]$ and application by $e(e')$. A multiple application $e_1(e_2)\ldots(e_n)$ will be denoted by $e_1(e_2,\ldots,e_n)$ and $\mathrm{FV}(e)$ is the set of free metavariables of e.

Roughly, we shall view the syntax of object level languages as certain expressions of an abstract syntax. Usually, the set GAr will consist of elements such as TERM or TYPE. The constants Con will consist of the function symbols arising from a given signature for an object level language, together with a countably infinite set of object level variables. The arity of the function symbols is specified for the particular object level language being considered; likewise for the object level variables. Then the raw syntax of an object level language will be defined as closed expressions of an abstract syntax. Thus variable binding will take place only in the meta λ calculus and substitution of object level terms will become application in the meta λ calculus. An exposition of the theory of arities and expressions can be found in [Cro90] or [NPS90].

Notational Conventions

(1) As a general convention, we shall omit typing information from morphisms, functors etc. For example, if we speak of a natural transformation τ (between a pair of bifunctors) which has components $\tau_{(A,B)}$ with domain $A\times TB$ and codomain $T(A\times B)$, then we shall often denote these data by

$$\tau\colon A\times TB\to T(A\times B)$$

rather than

$$\tau_{(A,B)}\colon A\times TB\to T(A\times B).$$

When using commutative diagrams, we shall place just enough typing information on the morphisms to make the picture unambiguous.

(2) Throughout this document, we shall use rules of the form

$$\frac{\text{Hypothesis}}{\text{Conclusion}}$$

where "Hypothesis" and "Conclusion" are syntactic expressions. In any such rule we assume that both "Hypothesis" and "Conclusion" are well formed.

(3) If we are given a list of expressions $E_1,\ldots,E_n$, we will often abbreviate such a list by $\vec{E}$.

(4) When discussing object level languages presented using an abstract syntax, we shall not introduce formal syntactic classes to distinguish between object level

variables and metavariables, but instead rely on the context of usage to make the distinction. For example, if M and N are expressions of an abstract syntax, for an expression of the form $M[N/x]$ to be meaningful, x must be a metavariable.

Chapter 1
Strong Monads and Let Categories

1.1 Introduction

Chapter 1 contains a review of some of the basic category theory which will be used throughout this thesis. The account is by no means comprehensive; it is not intended to be. We simply define some of the fundamental concepts which are relatively new to computer science and which form the backbone of almost all of the categorical structures which arise in this work. A standard reference for basic category theory is [Mac71]; additional material on monads which is particularly relevant can be found in [Man76] and [Kel82]. Those readers who are familiar with the original works on monads, strengths and enriched category theory will see that some of our basic notation is different from that used in the references. However, as our story unfolds, we hope that the choice of notation will be seen to be both appropriate and useful.

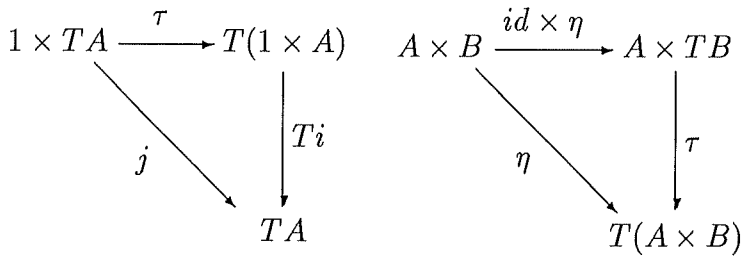
1.2 Monads and Tensorial Strengths

Definition 1.2.1 Let $\mathcal{C}$ be a category. Recall that a *monad* over $\mathcal{C}$ is a triple (T, η, μ) where $T{:}\mathcal{C} \to \mathcal{C}$ is an endofunctor, $\eta{:}\, Id_{\mathcal{C}} \to T$ is a natural transformation called the *unit* of the monad, $\mu{:}\, T^2 \to T$ is a natural transformation called the *multiplication* of the monad and these data satisfy the equations $Id_T = \mu \circ \eta_T$, $Id_T = \mu \circ T\eta$ (referred to by $\texttt{Monad}\eta$) and $\mu \circ T\mu = \mu \circ \mu_T$ (referred to by $\texttt{Monad}\mu$). We shall often speak just of the monad T.

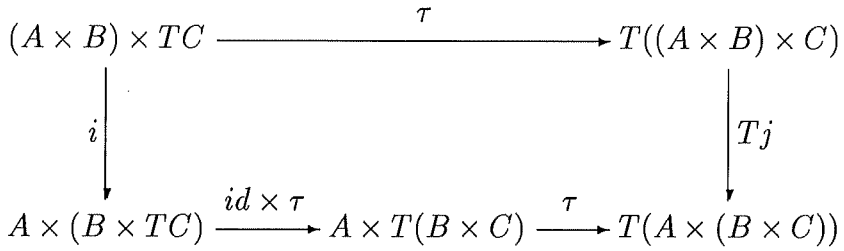
Definition 1.2.2 Let $\mathcal{C}$ be a category with finite products and let (T, η, μ) be a monad over $\mathcal{C}$. The monad T is said to possess a *tensorial strength* if there is a natural transformation τ with components

$$\tau_{(A,B)}{:}\, A \times TB \longrightarrow T(A \times B)$$

which satisfy the equational identities represented by the following commutative diagrams, where i and j denote obvious canonical isomorphisms:

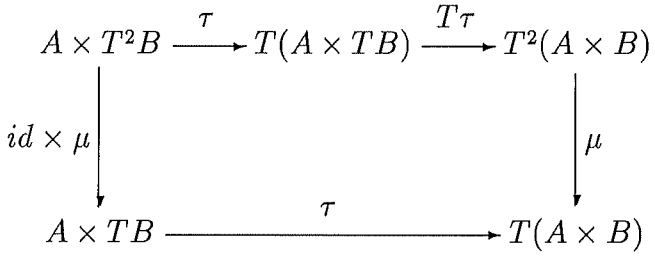


Tensor1



Tensor2

Tensor3



Tensor4

A *strong monad* (T, τ) is a monad T for which there is a given choice of tensorial strength τ. When no confusion can arise we refer to a strong monad T.

1.3 The Definition of a Let Category

Two Definitions of a Let Category

We begin by making the following definition:

Definition 1.3.1 A *let category* is specified by a category $\mathcal{C}$ with finite products, over which there is a strong monad T.

We can give an alternative definition of let category, which will prove to be of great value to us when we study the equational logic of these categories.

Definition 1.3.2 A *let category* is specified by a category $\mathcal{C}$ with finite products which enjoys the following properties:

- For each object A in $\mathcal{C}$, there is an object TA,
- for each object A in $\mathcal{C}$, there is a morphism $\eta_A: A \to TA$, and
- for each morphism $f: A \times B \to TC$, there is a morphism $\mathit{lift}(f): A \times TB \to TC$,

such that the following conditions are satisfied:

$$\mathit{lift}(g(f \times id_B)) = \mathit{lift}(g)(f \times id_{TB}) \qquad \texttt{LiftS}$$

where $f: A \to A'$, $g: A' \times B \to TC$,

$$\mathit{lift}(f)(id_A \times \eta_B) = f \qquad \texttt{LiftB}$$

where $f: A \times B \to TC$,

$$\mathit{lift}(\eta_B \pi_2) = \pi_2' \qquad \texttt{LiftH}$$

where $\pi_2: A \times B \to B$, and

$$\mathit{lift}(\mathit{lift}(g)\langle \pi_1, f \rangle) = \mathit{lift}(g)\langle \pi_1', \mathit{lift}(f) \rangle \qquad \texttt{LiftA}$$

where $f: A \times B \to TC$, $g: A \times C \to TD$.
We shall refer to the operation $f \mapsto \mathit{lift}(f)$ as *lifting*.

It will be useful to work with a derived operation on morphisms of $\mathcal{C}$ which we shall refer to as the *let* operation. We shall write $f \mapsto \mathit{let}(f)$ for this. Take projections

$$\pi_1: 1 \times A \to A \qquad \pi_2: 1 \times TA \to TA.$$

Then the let operation is specified by

$$\frac{A \xrightarrow{f} TB}{TA \xrightarrow{\pi_2^{-1}} 1 \times TA \xrightarrow{\mathit{lift}(f\pi_1)} TB}$$

This definition leads immediately to the following lemmas:

Lemma 1.3.3 Suppose we are given a let category in the sense of Definition 1.3.2. Then given morphisms $f: A \to TB$ and $g: B \to TC$ the following equational identities hold:

$$\begin{aligned} \mathit{let}(f)\eta &= f && \texttt{LetB} \\ \mathit{let}(\eta) &= id && \texttt{LetH} \\ \mathit{let}(\mathit{let}(g)f) &= \mathit{let}(g)\mathit{let}(f) && \texttt{LetA} \end{aligned}$$

Proof A trivial calculation. □

Lemma 1.3.4 Suppose we are given a let category in the sense of Definition 1.3.2. Given morphisms $f: A \to TB$ and $g: C \times B \to TD$, then

$$\mathit{lift}(g)(id \times \mathit{let}(f)) = \mathit{lift}(\mathit{lift}(g)(id \times f))$$

Proof Immediate from the definitions. □

Equivalence of the Definitions

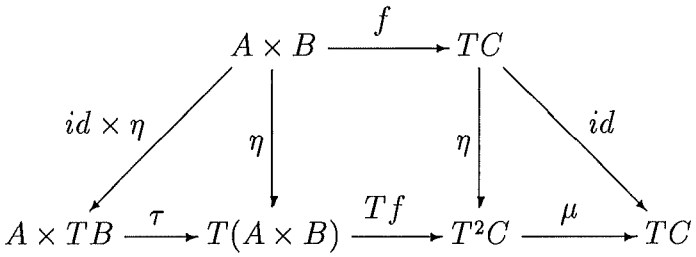
The previous definitions of let categories are equivalent in the following way:

Lemma 1.3.5 Given a let category $\mathcal{C}$ according to Definition 1.3.1 (Definition 1.3.2), then $\mathcal{C}$ has the structure of Definition 1.3.2 (Definition 1.3.1).

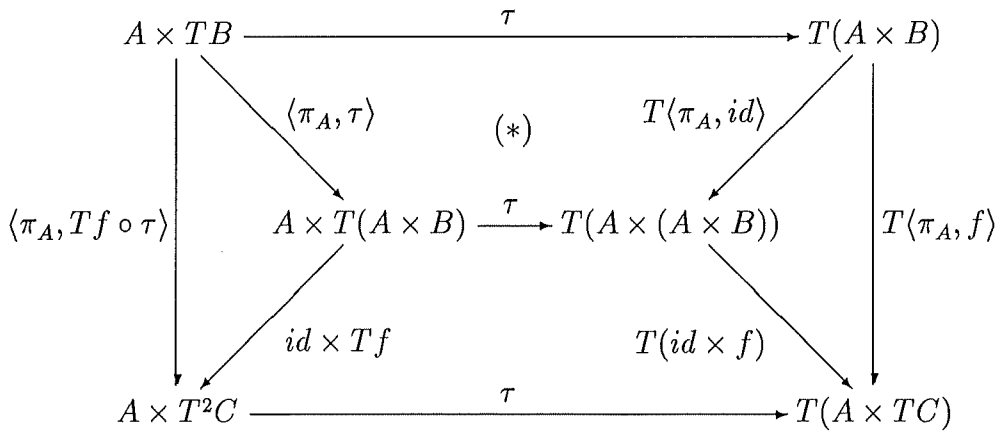
Proof Suppose we are given a category $\mathcal{C}$ with the structure of Definition 1.3.1. We show $\mathcal{C}$ has the structure of Definition 1.3.2. Definitions of the operation T and morphisms η_A are clear. The lifting operation is defined by

$$\frac{A \times B \xrightarrow{f} TC}{A \times TB \xrightarrow{\tau} T(A \times B) \xrightarrow{Tf} T^2C \xrightarrow{\mu} TC}$$

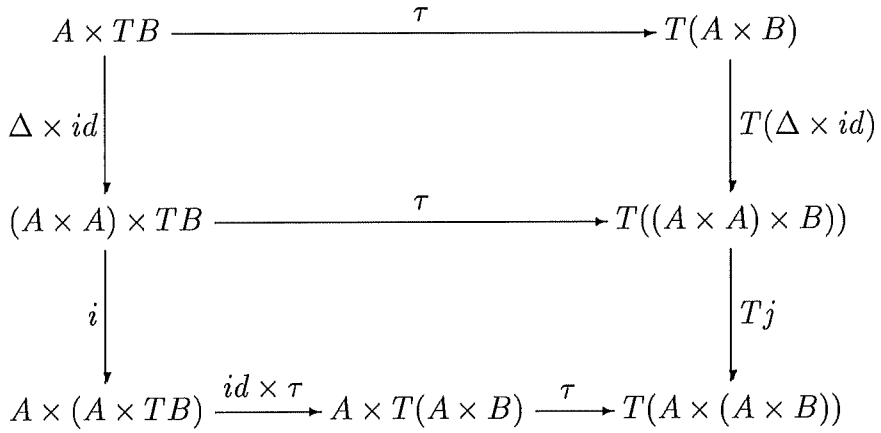
Now we have to check that our definition of lifting satisfies the necessary equations. Although this is essentially routine manipulation we supply the critical details; in each case we give the recipe for transforming the left-hand side of an equation to the right-hand side. `LiftS` follows from the naturality of τ and the functoriality of T. `LiftB` follows from `Tensor2`, the naturality of η and `Monad`η, thus



`LiftH` follows from functoriality of T, `Monad`η, naturality of τ and `Tensor1`. To show `LiftA` use functoriality of T followed by `Monad`μ and the naturality of μ. Then we need commutativity of



where the commutativity of $(*)$ follows from



with the lower square an instance of `Tensor3`. Finally apply `Tensor4` to get the result.

Conversely, suppose we are given a category $\mathcal{C}$ with the structure of Definition 1.3.2. We give the recipe for showing $\mathcal{C}$ also has the structure of Definition 1.3.1. The action of the monad T on objects and the components of the natural transformation η are clear. The definition of the monad T on morphisms is given by

$$\frac{A \xrightarrow{f} B}{TA \xrightarrow{let(\eta_B f)} TB}$$

The component of the natural transformation μ at A is $let(id_{TA})$ and the component of τ at A and B is $lift(\eta_{A \times B})$. We omit the routine details which verify that (T, η, μ, τ) is a strong monad. □

We have the following corollary:

Corollary 1.3.6 Given a category $\mathcal{C}$ over which there is a monad T, then $\mathcal{C}$ can be considered endowed with the following structure: for objects A and B of $\mathcal{C}$ and morphism $f\colon A \to TB$ there are morphisms $\eta_A\colon A \to TA$ and operations $A \mapsto TA$, $f \mapsto let(f)$, where the latter operation is defined by

$$\frac{A \xrightarrow{f} TB}{TA \xrightarrow{\mu \circ Tf} TB}$$

These data satisfy equations `LetB`, `LetH`, `LetA`.

Conversely, given a category $\mathcal{C}$ endowed with such a structure, we can define a monad T by

$$\frac{f\colon A \to B}{let(\eta_B f)\colon TA \to TB}$$

The components of η are clear and the component of μ at A is $let(id_{TA})$. □

Remark 1.3.7 The let operation arising from a monad T over a category $\mathcal{C}$ is closely allied to the construction of the Kleisli category $Kl(\mathcal{C})$ where composition in $Kl(\mathcal{C})$ is given by $(f, g) \mapsto let(g)f$.

1.4 Why Let Categories?

Motivating Ideas

A fundamental slogan for the categorical semantics of programming languages is: "types are interpreted as objects in a category and terms are interpreted as morphisms." Of course, for complicated languages this basic idea is adapted in many ways, but it is a good starting point. It has been known for some time now that formal systems correspond in a precise way to certain kinds of categorical structure, for example λ calculi (which are the theoretical backbone of functional programming languages) are the internal languages of cartesian closed categories and the cartesian closed categories provide a notion of syntax-independent presentations of λ calculi.

While functional languages are based on the principles of the λ calculus, the basic reduction strategies of λ calculi such as β and η do not elegantly distinguish between certain kinds of operational semantics such as call-by-name and call-by-value. For example, there is a notion of call-by-value strategy in λ calculi where we consider values to be those expressions of the formal system which are not applications. β reduction is then restricted to instances where the operand is a value. The naive formal system which captures this notion of call-by-value is a little ad-hoc: for an account see [Plo75].

Thus it might be reasonable to develop a formal system which separates the notions of computation and value: this is exactly what strong monads do for us, but in the world of semantics. Given a let category $\mathcal{C}$, if an object A models a type α then the object TA models computations of values of type α. This will be more clear if illustrated by example.

Examples Of Let Categories

Our examples will be over the category *Set*; they are lifted from [Mog89b].

Partial Computations

Given a type α, a partial computation of type α can be thought of as a program which either terminates yielding a value of type α, or which does not terminate. If values of type α are denoted by the set A, then we can denote the partial computations with values of type α by the coproduct $A + \{*\}$. Thus we have:

1. Operation on objects $A \mapsto A + \{*\}$.
2. Function $\eta_A \stackrel{\text{def}}{=} i\colon A \to A+\{*\}$ the left coproduct insertion (with right insertion j).

3. Given a function $f: A \times B \to C + \{*\}$ then we define $\mathit{lift}(f)$ by

$$\mathit{lift}(f)(a, s) \stackrel{\text{def}}{=} \begin{cases} f(a, b) & \text{if } s = i(b) \\ j(*) & \text{otherwise} \end{cases}$$

Corresponding to this let category, there is a strong monad with

$$\mu_A: (A + \{*\}) + \{*\} \to A + \{*\}$$

defined by

$$\mu_A(s) \stackrel{\text{def}}{=} \begin{cases} r & \text{if } s = i(r),\ r \in A + \{*\} \\ j(*) & \text{otherwise} \end{cases}$$

and tensorial strength

$$\tau_{(A,B)}: A \times (B + \{*\}) \to (A \times B) + \{*\}$$

defined by

$$\tau_{(A,B)}(a, s) \stackrel{\text{def}}{=} \begin{cases} i(a, b) & \text{if } s = i(b) \\ j(*) & \text{otherwise} \end{cases}$$

The intuitive meaning of the monad multiplication μ is that a partial computation P' of a partial computation P is defined when both P' and P are (with result given by composition of P' and P), and is otherwise undefined. The strength τ takes a pair consisting of a value and a partial computation and returns a partial computation, say P. If the original partial computation is defined, then so is P, with expected value. If the original computation is undefined, then so is P.

Computations which Raise Exceptions

Let E be a set which models certain exceptions. Then if the set A models values (of some type say α) we might consider the following (categorical) model:

1. Operation on sets $A \mapsto A + E$.
2. Function $\eta_A \stackrel{\text{def}}{=} i: A \to A + E$ the left coproduct insertion.
3. Given a function $f: A \times B \to (C + E)$ then we define $\mathit{lift}(f)$ by

$$\mathit{lift}(f)(a, s) \stackrel{\text{def}}{=} \begin{cases} f(a, b) & \text{if } s = i(b) \\ j(e) & \text{if } s = j(e) \end{cases}$$

The intuitive explanation of this example is similar to that for partial computations.

Computations with Side Effects in a Store S

Suppose that we are considering a model for an imperative language where the set S is to model some set of states. One such model is:

1. Operation on sets $A \mapsto S \to (A \times S)$.
2. Function $\eta_A : A \to S \to (A \times S)$ such that given $a \in A$ and $s \in S$ then $\eta_A(a)(s) \stackrel{\text{def}}{=} (a, s)$.
3. Given a function $f : A \times B \to (S \to (C \times S))$ then we define $\mathit{lift}(f)$ by

$$\mathit{lift}(f)(a, t) \stackrel{\text{def}}{=} s \mapsto f(a, \pi_1(ts))(\pi_2(ts)).$$

The intuitive meaning of the corresponding monad multiplication μ is to say that a computation which takes a state s and yields a computation c with side effect (i.e. new state) s' may be regarded as a computation which takes a state s and returns the value and state which are the result of the computation c on the value s'. The strength τ says that a value a and a computation c may be regarded as a computation which takes a state s and yields a pair of values, namely a and the value arising from cs, together with the state arising from cs.

Non-Deterministic Computations

Given a set A modeling values of type α we might model non-deterministic computations of type α via sets of possible results, namely $\mathcal{P}(A)$. A suitable model is:

1. Operation on sets $A \mapsto \mathcal{P}(A)$.
2. Function $\eta_A : A \to \mathcal{P}(A)$ such that given $a \in A$ then $\eta_A(a) \stackrel{\text{def}}{=} \{a\}$.
3. Given a function $f : A \times B \to \mathcal{P}(C)$ then $\mathit{lift}(f)$ is defined by

$$\mathit{lift}(f)(a, B') \stackrel{\text{def}}{=} \bigcup \{f(a, b') \mid b' \in B'\}.$$

We regard the denotation of a non-deterministic computation as the collection of all possible outputs. Any value can be trivially regarded as a non-deterministic computation. Finally, given a value and some non-deterministic computation we can think of the pair as a non-deterministic computation.

Chapter 2

The $\mathrm{FIX}_{=}$ Logical System

2.1 Review of the Computational Let Calculus

The computational let calculus[1] was introduced by Moggi [Mog89a]. Roughly, it is a formal system which embodies the idea of separating computations from values and is the syntactic analogue of the notion of let category. We make a formal distinction between the elements of a type α and computations of elements of that type; the latter are grouped into a new type $T\alpha$. We shall refer to a type of the form $T\alpha$ as a *computation type*. Moggi's computational let calculus contains the following formation rules:

$$\frac{\alpha \text{ type}}{T\alpha \text{ type}} \qquad \frac{M\colon \alpha}{\mathsf{Val}(M)\colon T\alpha} \qquad \frac{E\colon T\alpha \quad \overset{[x\colon \alpha]}{F(x)\colon T\beta}}{\mathsf{Let}\,(E,F)\colon T\beta}$$

These rules, together with the usual rules for unit type and binary product type constitute the term forming rules for the *computational let calculus*, which we shall denote by ML_T. Intuitively, $\mathsf{Val}(M)$ is the value M regarded as a trivial computation which immediately evaluates to itself. The term $\mathsf{Let}\,(E,F)$ denotes the computation which firstly tries to evaluate E to some value $M\colon \alpha$ and then proceeds to evaluate $F(M)$. These intended meanings are captured by three equational axioms:

$$\begin{aligned} \mathsf{Let}\,(\mathsf{Val}(M),F) &= F(M), \\ \mathsf{Let}\,(E,x.\mathsf{Val}(x)) &= E, \\ \mathsf{Let}\,(\mathsf{Let}\,(E,F),G) &= \mathsf{Let}\,(E,x.\mathsf{Let}\,(F(x),G)). \end{aligned}$$

2.2 Extensions of the System ML_T

The Basic Formal System ML_T

The reader may be wondering why the most fundamental equational logic which we discuss, namely ML_T, is assumed to contain unit and binary product types along with computation types. When considering practical computational issues, it will be useful to deal with algebraic terms (i.e. terms with a finite number of object level variables). According to the basic principle of categorical semantics (for an

[1] What we are calling the computational let calculus, Moggi refers to as the *computational metalanguage*.

account of this see [Cro90]), we shall need a category with at least finite products to interpret algebraic terms. However, it can be shown [Pit90a] that this is exactly the structure which we need to interpret a unit type and binary product type. Thus it makes little sense to exclude unit and binary product types and terms from the logic; and we gain some uniformity when considering details of the correspondence between the syntax of the logic and corresponding categorical structure.

The Use of ML_T and its Extensions

At least one of our tasks is to push forward the development of a versatile and general purpose metalogic for semantics of programming and computation languages. We can view the computational let calculus (and its extensions) as a formal metalanguage in which to translate the syntax and rules of other languages; for related work in this area the reader is referred to [Pit90b].

In this thesis we shall consider various extensions of ML_T. What extensions would be worthwhile investigating? At present, we have unit and binary product types, together with computation types. Additionally, some notion of function type will be essential for interpreting functional programming and computation languages. The equational logic ML_T extended by function types is usually called the computational λ calculus and denoted $\lambda\mathrm{ML}_T$. We refer the reader to [Mog89a] and [Pit90b].

Some other fundamental datatypes are natural numbers, booleans and coproducts. As discussed in Chapter 0, we shall introduce a new type called the fixpoint type. It will be interesting to study how the fixpoint type interacts with certain other types. For our purposes it will be convenient to study a calculus which, in addition to the fixpoint type, contains a null type, unit type, binary (co)product types and function type, along with a type of natural numbers. We are aiming to develop a constructive logic for reasoning about programming languages in general and recursive computations in particular: the logic arising from the terms and equations associated with the types just listed provides a good foundation on which to build.

2.3 Fixpoint Objects

We define the notion of a fixpoint object in a suitably structured category. This concept is due to Pitts; see also [CP90a] and [CP90b].

Definition 2.3.1 Given a category $\mathcal{C}$ with finite products, the $\mathcal{C}$ indexed category $\mathcal{C}^{(-)}$ is specified by:

- The objects of $\mathcal{C}^C$ are those of $\mathcal{C}$,
- $\mathcal{C}^C(A, B) \stackrel{\text{def}}{=} \mathcal{C}(C \times A, B)$ where composition of g and h in $\mathcal{C}^C$ is $h\langle \pi, g\rangle$ in $\mathcal{C}$,
- $\mathcal{C}^k(A) \stackrel{\text{def}}{=} A$ and $\mathcal{C}^k(g) \stackrel{\text{def}}{=} g(k \times id)$,

where $k\colon C' \to C$ in $\mathcal{C}$ and $A \stackrel{g}{\to} B \stackrel{h}{\to} D$ in $\mathcal{C}^C$.

Note that given a let category $\mathcal{C}$ there is an indexed endofunctor $T^{(-)}$ on $\mathcal{C}^{(-)}$ where $T^C(A) \stackrel{\text{def}}{=} TA$ and $T^C(g) \stackrel{\text{def}}{=} \mathit{lift}(\eta g)$. With this we make the following definition:

Definition 2.3.2 In a let category $\mathcal{C}$ a *fixpoint object* is specified by the following data

- An initial $T^{(-)}$ algebra whose structure map in the fibre $\mathcal{C}^C$ is the morphism $C \times T\Omega \xrightarrow{\pi} T\Omega \xrightarrow{\sigma} \Omega$ in $\mathcal{C}$. Thus given a morphism $f: TA \to A$ in $\mathcal{C}^C$, there is a unique morphism $\mathit{it}(f): \Omega \to A$ in $\mathcal{C}^C$ for which the following diagram commutes:

$$\begin{array}{ccc} C \times T\Omega & \xrightarrow{id \times \sigma} & C \times \Omega \\ {\scriptstyle \langle \pi_C, \mathit{lift}(\eta \circ \mathit{it}(f)) \rangle}\downarrow & & \downarrow{\scriptstyle \mathit{it}(f)} \\ C \times TA & \xrightarrow[f]{} & A \end{array}$$

- A global element $\omega: 1 \to T\Omega$ which gives rise to an equaliser diagram of the form

$$1 \xrightarrow{\omega} T\Omega \underset{id}{\overset{\eta\sigma}{\rightrightarrows}} T\Omega.$$

The definition of a fixpoint object (which we shall abbreviate to FPO) is reminiscent of a natural numbers object (NNO). Recall that the definition of a NNO in a category with finite products takes a particularly simple form when the ambient category is cartesian closed. Indeed we have the

Lemma 2.3.3 In a let ccc $\mathcal{C}$ a FPO is specified by

- An initial T algebra with structure map $\sigma: T\Omega \to \Omega$.
- A global element $\omega: 1 \to T\Omega$ which is the equaliser of $\eta\sigma$ and $id_{T\Omega}$ (exactly as in Definition 2.3.2).

Proof See Page 48; we shall use the internal logic of let ccc's to prove this result. □

The usual category-theoretic considerations imply that the structure constituting a FPO is determined uniquely up to isomorphism, within the given let category, by the above properties. One should note also that σ, being the structure morphism for the initial algebra of an endofunctor, is itself an isomorphism.

Some Examples of Fixpoint Objects

A domain-theoretic example of a let ccc with FPO is the category $\omega\mathcal{C}po$, whose objects are posets possessing joins of countably infinite chains, and whose morphisms are Scott continuous functions. We will refer to the objects as $\omega cpos$. The operation of adjoining a least element to an ωcpo D to give the lifted ωcpo

$$D_\perp = \{[d] \mid d \in D\} \cup \{\perp\}$$

gives a strong monad on $\omega\mathcal{C}po$. There is a FPO in $\omega\mathcal{C}po$, namely

$$\Omega = \{0 < 1 < \ldots < \top\},$$

which is equipped with structure map $\sigma\colon \Omega_\perp \to \Omega$ where σ is the continuous function

$$\sigma(e) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } e = \perp \\ n+1 & \text{if } e = [n] \\ \top & \text{if } e = [\top] \end{cases}$$

and $\omega \stackrel{\text{def}}{=} [\top] \in \Omega_\perp$. Some other monads on $\omega\mathcal{C}po$ that Moggi [Mog89b] points out as arising in denotational semantics also possess fixpoint objects. For example the exceptions monad $T(D) = (D+E)_\perp$ (with E some fixed discrete ωcpo of exceptions) and the side-effects monad $T(D) = S{\to}(D \times S)_\perp$ (with S some fixed discrete ωcpo of states) both possess fixpoint objects. We illustrate for the case of the exceptions monad when E is the terminal object $1 \stackrel{\text{def}}{=} \{*\}$ of $\omega\mathcal{C}po$. The FPO has an underlying set

$$\Omega \stackrel{\text{def}}{=} \{\top, (n,a), (n+1,b) \mid n \in \mathbb{N}\}$$

with order given by $\top$ a top element, $(n,a) \leq (n+1,b)$, and $(n,a) \leq (n+1,a)$, along with structure map $\sigma\colon (\Omega+1)_\perp \to \Omega$ where σ is the continuous function

$$\sigma(e) \stackrel{\text{def}}{=} \begin{cases} (0,a) & \text{if } e = \perp \\ (1,b) & \text{if } e = [*] \\ (n+1,a) & \text{if } e = [(n,a)] \\ (n+2,b) & \text{if } e = [(n+1,b)] \\ \top & \text{if } e = [\top] \end{cases}$$

and $\omega \stackrel{\text{def}}{=} [\top] \in \Omega_\perp$. This follows from the general theory for solving recursive domain equations over $\omega\mathcal{C}po$ enriched categories as presented in [SP82]. For suppose that T is an $\omega\mathcal{C}po$ enriched (strong) monad on $\omega\mathcal{C}po$ (where $\omega\mathcal{C}po$ is regarded as a symmetric monoidal category via finite products and is enriched over itself) and that T maps ωcpos to *pointed* ωcpos (i.e. ωcpos with least elements). To obtain a fixpoint object for such a T, one constructs the initial fixed object for T in the category of pointed ωcpos and embedding-projection pairs by iterating T starting at the one element ωcpo, yielding an isomorphism $\sigma : T(\Omega) \cong \Omega$. Then (Ω, σ) is an initial algebra for $T : \omega\mathcal{C}po \to \omega\mathcal{C}po$, and dually (Ω, σ^{-1}) is a final coalgebra for that functor: This follows from the limit-colimit coincidence for $\omega\mathcal{C}po$ enriched categories. The initial algebra property gives us the the first part of the definition of fixpoint object; and Freyd has observed that the second part of Definition 2.3.2 is implied by the coalgebra property. We record this latter observation as a lemma.

Lemma 2.3.4 Given a let ccc, suppose that $\sigma : T\Omega \to \Omega$ is an initial algebra for the functor T (so that in particular, σ is an isomorphism). Suppose further that $\sigma^{-1} : \Omega \to T\Omega$ is a final coalgebra for T. Then there is a global element $\omega : 1 \to T\Omega$ making Ω, σ, ω a fixpoint object for T.

Proof The final coalgebra property means that for any $g : A \to TA$ there is a unique morphism $\hat{g} : A \to \Omega$ satisfying $\sigma^{-1}\hat{g} = T(\hat{g})g$.
Define $\omega : 1 \to T\Omega$ to be $\sigma^{-1}\hat{\eta_1}$. From the defining property of $\hat{\eta_1}$ and the naturality of η we get

$$\omega = \sigma^{-1}\hat{\eta_1} = T(\hat{\eta_1})\eta_1 = \eta_\Omega\hat{\eta_1} = (\eta_\Omega\sigma)\omega.$$

If $f : A \to T\Omega$ is any other morphism satisfying $f = (\eta_\Omega\sigma)f$, we have to see that $f = \omega\,!$. But from $f = (\eta_\Omega\sigma)f$ and the naturality of η one has

$$\sigma^{-1}(\sigma f) = f = \eta_\Omega(\sigma f) = T(\sigma f)\eta_A$$

Hence by the uniqueness part of the coalgebra property, $\sigma f = \hat{\eta_A}$ and thus $f = \sigma^{-1}\eta_A$. The same argument applies equally well with $\omega\,!$ for f. Therefore $f = \sigma^{-1}\eta_A = \omega\,!$. □

2.4 The Internal Logic Corresponding to a Fixpoint Object

It is well known that there is a natural correspondence between let categories $\mathcal{C}$ and ML_T theories [Mog89b]. Given such a $\mathcal{C}$, the corresponding computational let calculus is often referred to as the (equational) internal logic of the category $\mathcal{C}$. It is possible to extend the calculus ML_T to capture syntactically the notion of a FPO. This will entail adding a type *fix*, called the *fixpoint* type, to ML_T together with certain term forming and equality rules. We present these rules here in an informal natural deduction style:

$$\frac{}{\omega : T\mathit{fix}} \qquad \frac{E : T\mathit{fix}}{\sigma(E) : \mathit{fix}} \qquad \frac{\begin{array}{c}[x : T\alpha] \\ F(x) : \alpha \quad N : \mathit{fix}\end{array}}{\mathsf{It}_\alpha(F, N) : \alpha}$$

$$\frac{}{\omega = \mathsf{Val}(\sigma(\omega))} \qquad \frac{E = \mathsf{Val}(\sigma(E))}{E = \omega}$$

$$\frac{\begin{array}{c}[x : T\alpha] \\ F(x) : \alpha \quad E : T\mathit{fix}\end{array}}{\mathsf{It}_\alpha(F, \sigma(E)) = F(\mathsf{Let}\,(E, n.\mathsf{Val}(\mathsf{It}_\alpha(F, n))))}$$

$$\frac{\begin{array}{ccc}[x : T\alpha] & [n : \mathit{fix}] & [e : T\mathit{fix}] \\ F(x) : \alpha & G(n) : \alpha & G(\sigma(e)) = F(\mathsf{Let}\,(e, n.\mathsf{Val}(G(n)))) \quad N : \mathit{fix}\end{array}}{G(N) = \mathsf{It}_\alpha(F, N)}$$

The type *fix* is so called because in its presence we are always able to form fixpoints of certain terms; correspondingly, in a let category with FPO, we are always guaranteed the existence of fixpoints of certain morphisms. We will leave the precise details until after we have defined a formal system which contains the equational rules for the fixpoint type.

2.5 The Equational Logic FIX$_=$

In Section 2.2 we discussed some appropriate extensions to ML$_T$. The most basic of these was the addition of function types, resulting in the computational λ calculus λML$_T$. Adding a fixpoint type *fix*, coproduct types $\alpha + \beta$ and a natural number type *nat* to the computational λ calculus λML$_T$, we arrive at a system FIX$_=$ which extends Gödel's system T [Gir89]. FIX$_=$ admits sound translations of Plotkin's PCF [Plo77] and we shall return to the topic of PCF translations in Chapter 6. We now formally define FIX$_=$.

Signatures for FIX$_=$

Definition 2.5.1 A FIX$_=$ *signature*, denoted by Sg, is specified by:

- A collection of *types*. The types are built up in the following way. We are given a collection of *basic ground types*, together with the *distinguished ground types unit*, *null*, *nat*, and *fix*. The types are now specified by the following grammar:

$$\alpha ::= \gamma \mid \alpha \times \alpha \mid \alpha + \alpha \mid \alpha \rightarrow \alpha \mid T\alpha$$

  where γ denotes any ground type.

- A collection of *basic function symbols*, together with the following distinguished function symbols: $\langle\rangle$, $\langle -, - \rangle$, Fst, Snd, Inl_α, Inr_β, $\{\}_\alpha$, $\{-,-\}$, λ_α, App, Val, Let, O, Suc, ItNat, ω, σ, It_α.

- A *sorting* for each of the basic function symbols, which is a list of $n+1$ types and will be written:

$$f\colon \alpha_1, \ldots, \alpha_n \rightarrow \alpha_{n+1}.$$

  In the case that n is zero, we shall write $f\colon \alpha$. We say that f is an *n-ary* basic function symbol when its sorting consists of $n+1$ types.

Given such a FIX$_=$ signature, we define from this an abstract syntax signature $\Sigma = (GAr, Con)$. The collection of ground arities, GAr, is simply the one element set $\{\text{TERM}\}$. The collection of constants Con consists of the basic function symbols which have arity $\text{TERM}^n \rightarrow \text{TERM}$ whenever the sorting of f consists of $n+1$ types, a countably infinite set of object level variables which have arity TERM, together with the distinguished function symbols. The distinguished function symbols which will represent the simply typed λ calculus, finite products and natural numbers have

their usual arities. The remaining distinguished function symbols have the following arities:

1. ω: TERM
2. $\{\}_\alpha, \mathsf{Inl}_\alpha, \mathsf{Inr}_\beta, \mathsf{Val}, \sigma$: TERM $\to$ TERM
3. Let : TERM $\to$ (TERM $\to$ TERM) $\to$ TERM
4. It_α: (TERM $\to$ TERM) $\to$ TERM $\to$ TERM
5. $\{-,-\}$: (TERM $\to$ TERM) $\to$ (TERM $\to$ TERM) $\to$ TERM $\to$ TERM

Associated with a FIX$_=$ signature is a collection of *raw* FIX$_=$ *terms.* When no confusion can arise we shall refer to these just as *raw terms.* The raw terms are the closed expressions of the abstract syntax generated from Σ with arity TERM.

Remark 2.5.2 We make the following abbreviations: Write FM for $\mathsf{App}(F, M)$ and $F^N(M)$ for $\mathsf{ItNat}(F, N, M)$.

Terms in Context for FIX$_=$

A *context*, Γ, is a finite list of (variable, type) pairs written

$$[x_1: \alpha_1, \ldots, x_n: \alpha_n]$$

where the object level variables $x_1, \ldots, x_n$ are *distinct.* An *empty* context will be denoted by white space. We will use the (self explanatory) notation $\Gamma, x: \alpha$ and Γ, Γ' for the concatenation of contexts, (where of course x does not occur in Γ) and will write $\Gamma \subset \Gamma'$ to mean that Γ is a sub-list of Γ'. We shall write

$$\Gamma \vdash M: \alpha$$

for the judgement that given the context Γ, the raw FIX$_=$ term M is well formed and has type α. In such cases the raw term M will be referred to as a FIX$_=$ *term in context.* These judgements are generated by the following rules:

Variables

$$\frac{}{\Gamma, x: \alpha, \Gamma' \vdash x: \alpha}$$

Basic Function Symbols

$$\frac{\Gamma \vdash M_1: \alpha_1, \ldots, M_n: \alpha_n}{\Gamma \vdash f(M_1, \ldots, M_n): \alpha}$$

where f is any function symbol with sorting $f: \alpha_1, \ldots, \alpha_n \to \alpha$

Unit Terms

$$\frac{}{\Gamma \vdash \langle\rangle : \mathit{unit}}$$

Null Terms

$$\frac{\Gamma \vdash M : \mathit{null}}{\Gamma \vdash \{\}_\alpha(M) : \alpha}$$

Binary Product Terms

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash N : \beta}{\Gamma \vdash \langle M, N\rangle : \alpha \times \beta} \qquad \frac{\Gamma \vdash P : \alpha \times \beta}{\Gamma \vdash \mathsf{Fst}(P) : \alpha} \qquad \frac{\Gamma \vdash P : \alpha \times \beta}{\Gamma \vdash \mathsf{Snd}(P) : \beta}$$

Binary Coproduct Terms

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathsf{Inl}_\beta(M) : \alpha + \beta} \qquad \frac{\Gamma \vdash N : \beta}{\Gamma \vdash \mathsf{Inr}_\alpha(N) : \alpha + \beta}$$

$$\frac{\Gamma, x : \alpha \vdash F(x) : \gamma \quad \Gamma, y : \beta \vdash G(y) : \gamma \quad \Gamma \vdash C : \alpha + \beta}{\Gamma \vdash \{F, G\}(C) : \gamma}$$

Function Terms

$$\frac{\Gamma, x : \alpha \vdash F(x) : \beta}{\Gamma \vdash \lambda_\alpha(F) : \alpha \to \beta} \qquad \frac{\Gamma \vdash F : \alpha \to \beta \quad \Gamma \vdash M : \alpha}{\Gamma \vdash FM : \beta}$$

Computation Terms

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathsf{Val}(M) : T\alpha} \qquad \frac{\Gamma \vdash E : T\alpha \quad \Gamma, x : \alpha \vdash F(x) : T\beta}{\Gamma \vdash \mathsf{Let}\,(E, F) : T\beta}$$

Natural Number Terms

$$\frac{}{\Gamma \vdash \mathsf{0} : \mathit{nat}} \qquad \frac{\Gamma \vdash M : \mathit{nat}}{\Gamma \vdash \mathsf{Suc}(M) : \mathit{nat}}$$

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma, x : \alpha \vdash F(x) : \alpha \quad \Gamma \vdash N : \mathit{nat}}{\Gamma \vdash F^N(M) : \alpha}$$

Fixpoint Terms

$$\frac{}{\Gamma \vdash \omega : T\mathit{fix}} \qquad \frac{\Gamma \vdash E : T\mathit{fix}}{\Gamma \vdash \sigma(E) : \mathit{fix}} \qquad \frac{\Gamma, x : T\alpha \vdash F(x) : \alpha \quad \Gamma \vdash N : \mathit{fix}}{\Gamma \vdash \mathsf{It}_\alpha(F, N) : \alpha}$$

Remark 2.5.3 The usual rules for weakening of contexts, and substitution of raw terms for object level variables, are derivable from the above rules by simple structural induction.

Equational Theories for $\mathrm{FIX}_=$

A $\mathrm{FIX}_=$ *equation in context* takes the form

$$\Gamma \vdash M = M' : \alpha$$

where M, M' are raw $\mathrm{FIX}_=$ terms satisfying $\Gamma \vdash M : \alpha$ and $\Gamma \vdash M' : \alpha$. A $\mathrm{FIX}_=$ *theory*, Th, is specified by a $\mathrm{FIX}_=$ signature, together with a specific collection of equations in context, which are called the *axioms* of Th. The collection of *theorems* of Th is the least collection of equations in context which contains the axioms of Th, and is closed under the following rules:

Function Symbol Congruence

Every function symbol is required to be a congruence

Weakening

$$\frac{\Gamma \vdash M = M' : \alpha}{\Gamma' \vdash M = M' : \alpha} \quad \text{where } \Gamma \subset \Gamma'.$$

Equational Logic

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash M = M : \alpha} \qquad \frac{\Gamma \vdash M = M' : \alpha}{\Gamma \vdash M' = M : \alpha} \qquad \frac{\Gamma \vdash M = M' : \alpha \quad \Gamma \vdash M' = M'' : \alpha}{\Gamma \vdash M = M'' : \alpha}$$

Unit Equations

$$\frac{\Gamma \vdash M : \mathit{unit}}{\Gamma \vdash M = \langle\rangle : \mathit{unit}}$$

Null Equations

$$\frac{\Gamma, x : \mathit{null} \vdash F(x) : \alpha \quad \Gamma \vdash M : \mathit{null}}{\Gamma \vdash F(M) = \{\}_\alpha(M) : \alpha}$$

Binary Product Equations

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash N : \beta}{\Gamma \vdash \mathsf{Fst}(\langle M, N\rangle) = M : \alpha} \qquad \frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash N : \beta}{\Gamma \vdash \mathsf{Snd}(\langle M, N\rangle) = N : \beta}$$

$$\frac{\Gamma \vdash P : \alpha \times \beta}{\Gamma \vdash \langle \mathsf{Fst}(P), \mathsf{Snd}(P)\rangle = P : \alpha \times \beta}$$

Binary Coproduct Equations

$$\frac{\Gamma, x{:}\alpha \vdash F(x){:}\gamma \quad \Gamma, y{:}\beta \vdash G(y){:}\gamma \quad \Gamma \vdash M{:}\alpha}{\Gamma \vdash \{F,G\}(\mathsf{Inl}_\beta(M)) = F(M){:}\gamma}$$

$$\frac{\Gamma, x{:}\alpha \vdash F(x){:}\gamma \quad \Gamma, y{:}\beta \vdash G(y){:}\gamma \quad \Gamma \vdash N{:}\beta}{\Gamma \vdash \{F,G\}(\mathsf{Inr}_\alpha(N)) = G(N){:}\gamma}$$

$$\frac{\Gamma, z{:}\alpha+\beta \vdash H(z){:}\gamma \quad \Gamma \vdash C{:}\alpha+\beta}{\Gamma \vdash \{u.H(\mathsf{Inl}_\beta(u)), v.H(\mathsf{Inr}_\alpha(v))\}(C) = H(C){:}\gamma}$$

Function Equations

$$\frac{\Gamma, x{:}\alpha \vdash F(x){:}\beta \quad \Gamma \vdash M{:}\alpha}{\Gamma \vdash \lambda_\alpha(F)M = F(M){:}\beta} \qquad \frac{\Gamma \vdash M{:}\alpha{\to}\beta}{\Gamma \vdash \lambda_\alpha(u.M\,u) = M{:}\alpha{\to}\beta}$$

Computation Equations

$$\frac{\Gamma \vdash M{:}\alpha \quad \Gamma, x{:}\alpha \vdash F(x){:}T\beta}{\Gamma \vdash \mathsf{Let}\,(\mathsf{Val}(M), F) = F(M){:}T\beta} \qquad \frac{\Gamma \vdash E{:}T\alpha}{\Gamma \vdash \mathsf{Let}\,(E, x.\mathsf{Val}(x)) = E{:}T\alpha}$$

$$\frac{\Gamma \vdash E{:}T\alpha \quad \Gamma, x{:}\alpha \vdash F(x){:}T\beta \quad \Gamma, y{:}\beta \vdash G(y){:}T\gamma}{\Gamma \vdash \mathsf{Let}\,(\mathsf{Let}\,(E,F),G) = \mathsf{Let}\,(E, u.\mathsf{Let}\,(F(u),G)){:}T\gamma}$$

Mono Condition

$$\frac{\Gamma \vdash \mathsf{Val}(M) = \mathsf{Val}(M'){:}T\alpha}{\Gamma \vdash M = M'{:}\alpha}$$

Natural Number Equations

$$\frac{\Gamma \vdash M{:}\alpha \quad \Gamma, x{:}\alpha \vdash F(x){:}\alpha}{\Gamma \vdash F^{\mathsf{O}}(M) = M{:}\alpha}$$

$$\frac{\Gamma \vdash M{:}\alpha \quad \Gamma, x{:}\alpha \vdash F(x){:}\alpha \quad \Gamma \vdash N{:}nat}{\Gamma \vdash F^{\mathsf{Suc}(N)}(M) = F(F^N(M)){:}\alpha}$$

$$\frac{\left\{\begin{array}{l}\Gamma \vdash N{:}nat \quad \Gamma \vdash G(\mathsf{O}) = M{:}\alpha \\ \Gamma, x{:}\alpha \vdash F(x){:}\alpha \quad \Gamma, n{:}nat \vdash G(\mathsf{Suc}(n)) = F(G(n)){:}nat \quad \Gamma, n{:}nat \vdash G(n){:}\alpha\end{array}\right.}{\Gamma \vdash G(N) = F^N(M){:}\alpha}$$

Fixpoint Equations

$$\frac{}{\Gamma \vdash \omega = \mathsf{Val}(\sigma(\omega)){:}T\mathit{fix}} \qquad \frac{\Gamma \vdash E = \mathsf{Val}(\sigma(E)){:}T\mathit{fix}}{\Gamma \vdash E = \omega{:}T\mathit{fix}}$$

$$\frac{\Gamma, x{:}T\alpha \vdash F(x){:}\alpha \quad \Gamma \vdash E{:}T\mathit{fix}}{\Gamma \vdash \mathsf{It}_\alpha(F, \sigma(E)) = F(\mathsf{Let}\,(E, n.\mathsf{Val}(\mathsf{It}_\alpha(F,n)))){:}\alpha}$$

$$\frac{\left\{\begin{array}{l}\Gamma \vdash N{:}\mathit{fix} \quad \Gamma, x{:}T\alpha \vdash F(x){:}\alpha \\ \Gamma, e{:}T\mathit{fix} \vdash G(\sigma(e)) = F(\mathsf{Let}\,(e, n.\mathsf{Val}(G(n)))){:}\alpha \quad \Gamma, n{:}\mathit{fix} \vdash G(n){:}\alpha\end{array}\right.}{\Gamma \vdash G(N) = \mathsf{It}_\alpha(F, N){:}\alpha}$$

The usual rule for substitution is derivable from the above rules. More precisely we have

Lemma 2.5.4 The rule

$$\frac{\Gamma, x{:}\,\alpha \vdash N(x) = N'(x){:}\,\beta \quad \Gamma \vdash M = M'{:}\,\alpha}{\Gamma \vdash N(M) = N'(M'){:}\,\beta}$$

is derivable from the rules given on Page 27.

Proof Use Function Symbol Congruence to deduce $\Gamma \vdash \lambda_\alpha(N)\, M = \lambda_\alpha(N')\, M'{:}\,\beta$. The result follows. □

Remark 2.5.5 The Mono Condition is the syntactic requirement which captures the idea that values may be trivially regarded as computations which evaluate immediately to themselves. Note also that there are no side conditions on any of the extensionality rules due to the fact that object level variables are regarded as constants in the meta λ calculus.

Chapter 3
Categorical Semantics of the $\mathrm{FIX}_=$ Logic

3.1 FIX Categories

Definition 3.1.1 A *FIX category* is a let ccc which is endowed with finite coproducts, FPO, NNO and for which the components of the unit of the monad are monics.

Imposing the condition that the unit components are monic captures semantically the idea that a value may be trivially regarded as a computation. Note also that to specify a FIX category we take a *fixed* choice of strong monad.

The syntax of $\mathrm{FIX}_=$, in particular Null and Binary Coproduct Terms and Equations, will be interpreted in FIX categories. In order to interpret such syntax soundly, the FIX category must have *stable* finite coproducts i.e. the functor $C \times (-)$ must preserve finite coproducts. This is automatic: any FIX category is cartesian closed and so $C \times (-)$ has a right adjoint.

Definition 3.1.2 A FIX category *morphism* $F\colon \mathcal{C} \to \mathcal{D}$ is a monad morphism (F, i) between the underlying monads for which F preserves the categorical structure up to isomorphism, and each i is a natural *iso*morphism.

Remark 3.1.3 Note that the operation on objects $A \mapsto TA$ is not a categorical property of the FIX category; as noted above there is a fixed choice of strong monad. Thus to be given a FIX category morphism $F\colon (\mathcal{C}, T) \to (\mathcal{D}, S)$ means for each object A in $\mathcal{C}$ we are given an isomorphism $i_A\colon FTA \cong SFA$ which is compatible with the remaining structure. For example, suppose we are given a morphism $f\colon A \times B \to TC$ in $\mathcal{C}$. Then there is a morphism $g\colon FA \times FB \cong F(A \times B) \xrightarrow{Ff} FTC \xrightarrow{i_C} SFC$ and a morphism $h\colon FA \times FTB \cong F(A \times TB) \xrightarrow{F(lift(f))} FTC$ for which, by definition, the following diagram commutes:

$$\begin{array}{ccc} FA \times SFB & \xrightarrow{lift(g)} & SFC \\ {\scriptstyle id \times i_B}\uparrow & & \uparrow{\scriptstyle i_C} \\ FA \times FTB & \xrightarrow[h]{} & FTC \end{array}$$

The (categorical) isomorphisms are, of course, the canonical ones, arising from the categorical property of "having (specified) binary products" together with the definition of finite product preserving functor.

An example of such a FIX category is $\omega\mathcal{C}po$. Note that the forgetful functor from $\omega\mathcal{C}po$ to $\mathcal{S}et$ reflects monics.

3.2 Categorical Semantics of FIX$_=$

Structures for FIX$_=$ Signatures

Let Sg be a FIX$_=$ signature. A *structure*, **M**, in a FIX category $\mathcal{C}$ is specified by the following data:

- An object $[\![\gamma]\!]$ for each basic ground type γ of Sg, and
- for each basic function symbol $f\colon \alpha_1, \ldots, \alpha_n \to \alpha$, a morphism in $\mathcal{C}$ of the form

$$[\![f]\!]\colon [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!] \to [\![\alpha]\!].$$

Interpretation of the FIX$_=$ Types

Given a structure **M** we shall now show how to interpret the syntax of FIX$_=$ in a FIX category $\mathcal{C}$. The types are interpreted as objects in the category, where the interpretation of a type α is denoted by $[\![\alpha]\!]$. We make the following definition

Definition 3.2.1 The interpretation of a type α is given by:

- $[\![unit]\!] \stackrel{\text{def}}{=} 1$ where 1 is the terminal object.
- $[\![null]\!] \stackrel{\text{def}}{=} 0$ where 0 is the initial object.
- $[\![nat]\!] \stackrel{\text{def}}{=} N$ where N is the NNO.
- $[\![fix]\!] \stackrel{\text{def}}{=} \Omega$ where Ω is the FPO.
- $[\![\alpha \times \beta]\!] \stackrel{\text{def}}{=} [\![\alpha]\!] \times [\![\beta]\!]$.
- $[\![\alpha + \beta]\!] \stackrel{\text{def}}{=} [\![\alpha]\!] + [\![\beta]\!]$.
- $[\![\alpha \to \beta]\!] \stackrel{\text{def}}{=} [\![\alpha]\!] \to [\![\beta]\!]$.
- $[\![T\alpha]\!] \stackrel{\text{def}}{=} T[\![\alpha]\!]$.

Interpretation of the FIX$_=$ Terms in Context

Given a context $\Gamma = [x_1{:}\,\alpha_1, \ldots, x_n{:}\,\alpha_n]$ let $[\![\Gamma]\!] \stackrel{\text{def}}{=} [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!]$. Then for each context Γ, raw term M and type α for which $\Gamma \vdash M{:}\,\alpha$ is a valid judgement, we interpret this by giving a morphism

$$[\![\Gamma \vdash M{:}\,\alpha]\!]{:}\,[\![\Gamma]\!] \to [\![\alpha]\!].$$

Note that when $\Gamma \vdash M{:}\,\alpha$ is a valid judgement, because the type α is uniquely determined by M and Γ, we will abbreviate $[\![\Gamma \vdash M{:}\,\alpha]\!]$ to just $[\![\Gamma.M]\!]$, and to $[\![M]\!]$ if Γ is empty.

Definition 3.2.2 The semantics of terms in context is defined by a structural induction on terms:

- $[\![\Gamma, x{:}\,\alpha, \Gamma'.x]\!] \stackrel{\text{def}}{=} \pi{:}\,[\![\Gamma]\!] \times [\![\alpha]\!] \times [\![\Gamma']\!] \to [\![\alpha]\!]$.
- Let $f{:}\,\alpha_1, \ldots, \alpha_n \to \alpha$ be a basic function symbol; then $[\![\Gamma.f(M_1, \ldots, M_n)]\!] \stackrel{\text{def}}{=} [\![f]\!]\langle [\![\Gamma.\vec{M}]\!]\rangle{:}\,[\![\Gamma]\!] \to [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!] \to [\![\alpha]\!]$.
- $[\![\Gamma.\langle\rangle]\!] \stackrel{\text{def}}{=} !{:}\,[\![\Gamma]\!] \to 1$.
- $[\![\Gamma.\{\}_\alpha(M)]\!] \stackrel{\text{def}}{=} ![\![\Gamma.M]\!]{:}\,[\![\Gamma]\!] \to 0 \to [\![\alpha]\!]$.
- $[\![\Gamma.\langle M, N\rangle]\!] \stackrel{\text{def}}{=} \langle [\![\Gamma.M]\!], [\![\Gamma.N]\!]\rangle{:}\,[\![\Gamma]\!] \to [\![\alpha]\!] \times [\![\beta]\!]$.
- $[\![\Gamma.\mathsf{Fst}(P)]\!] \stackrel{\text{def}}{=} \pi[\![\Gamma.P]\!]{:}\,[\![\Gamma]\!] \to [\![\alpha]\!] \times [\![\beta]\!] \to [\![\alpha]\!]$.
- $[\![\Gamma.\mathsf{Snd}(P)]\!] \stackrel{\text{def}}{=} \pi[\![\Gamma.P]\!]{:}\,[\![\Gamma]\!] \to [\![\alpha]\!] \times [\![\beta]\!] \to [\![\beta]\!]$.
- $[\![\Gamma.\{F, G\}(C)]\!] \stackrel{\text{def}}{=} \{[\![\Gamma, x{:}\,\alpha.F(x)]\!], [\![\Gamma, y{:}\,\beta.G(y)]\!]\}\langle id, [\![\Gamma.C]\!]\rangle$
  $: [\![\Gamma]\!] \to [\![\Gamma]\!] \times ([\![\alpha]\!] + [\![\beta]\!]) \to [\![\gamma]\!]$.
- $[\![\Gamma.\mathsf{Inl}_\beta(M)]\!] \stackrel{\text{def}}{=} i[\![\Gamma.M]\!]{:}\,[\![\Gamma]\!] \to [\![\alpha]\!] \to [\![\alpha]\!] + [\![\beta]\!]$.
- $[\![\Gamma.\mathsf{Inr}_\alpha(N)]\!] \stackrel{\text{def}}{=} j[\![\Gamma.N]\!]{:}\,[\![\Gamma]\!] \to [\![\beta]\!] \to [\![\alpha]\!] + [\![\beta]\!]$.
- $[\![\Gamma.\lambda_\alpha(F)]\!] \stackrel{\text{def}}{=} cur([\![\Gamma, x{:}\,\alpha.F(x)]\!]){:}\,[\![\Gamma]\!] \to ([\![\alpha]\!] \to [\![\beta]\!])$.
- $[\![\Gamma.FM]\!] \stackrel{\text{def}}{=} ap\langle [\![\Gamma.F]\!], [\![\Gamma.M]\!]\rangle{:}\,[\![\Gamma]\!] \to ([\![\alpha]\!] \to [\![\beta]\!]) \times [\![\alpha]\!] \to [\![\beta]\!]$.
- $[\![\Gamma.\mathsf{Val}(M)]\!] \stackrel{\text{def}}{=} \eta[\![\Gamma.M]\!]{:}\,[\![\Gamma]\!] \to [\![\alpha]\!] \to T[\![\alpha]\!]$.
- $[\![\Gamma.\mathsf{Let}\,(E, F)]\!] \stackrel{\text{def}}{=} lift([\![\Gamma, x{:}\,\alpha.F(x)]\!])\langle id, [\![\Gamma.E]\!]\rangle{:}\,[\![\Gamma]\!] \to [\![\Gamma]\!] \times T[\![\alpha]\!] \to T[\![\beta]\!]$.
- $[\![\Gamma.\mathsf{O}]\!] \stackrel{\text{def}}{=} 0!{:}\,[\![\Gamma]\!] \to 1 \to N$.
- $[\![\Gamma.\mathsf{Suc}(N)]\!] \stackrel{\text{def}}{=} s[\![\Gamma.N]\!]{:}\,[\![\Gamma]\!] \to N \to N$.

- $[\![\Gamma.F^N(M)]\!] \stackrel{\text{def}}{=} h\langle id, [\![\Gamma.N]\!]\rangle : [\![\Gamma]\!] \to [\![\Gamma]\!] \times N \to [\![\alpha]\!]$

  where h is the unique morphism arising from the universal property of the NNO together with the morphism

$$[\![\Gamma, x{:}\alpha.F(x)]\!]\langle id, [\![\Gamma.M]\!]\rangle : [\![\Gamma]\!] \to [\![\Gamma]\!] \times [\![\alpha]\!] \to [\![\alpha]\!].$$

- $[\![\Gamma.\omega]\!] \stackrel{\text{def}}{=} \omega! : [\![\Gamma]\!] \to 1 \to T\Omega.$

- $[\![\Gamma.\sigma(E)]\!] \stackrel{\text{def}}{=} \sigma[\![\Gamma.E]\!] : [\![\Gamma]\!] \to T\Omega \to \Omega.$

- $[\![\Gamma.\mathsf{lt}_\alpha(F,N)]\!] \stackrel{\text{def}}{=} h\langle id, [\![\Gamma.N]\!]\rangle : [\![\Gamma]\!] \to [\![\Gamma]\!] \times \Omega \to [\![\alpha]\!]$

  where h is the unique morphism arising from the universal property of the FPO together with the morphism

$$[\![\Gamma, x{:}T\alpha.F(x)]\!] : [\![\Gamma]\!] \times T[\![\alpha]\!] \to [\![\alpha]\!].$$

This completes the definition of the categorical interpretations of the terms in context.

Models of FIX$_=$ Theories

A structure $\mathbf{M}$ in $\mathcal{C}$ for a signature Sg *satisfies* an equation in context $\Gamma \vdash M = M' : \alpha$ if $[\![\Gamma.M]\!]$ and $[\![\Gamma.M']\!]$ are equal morphisms in $\mathcal{C}$. Given a FIX$_=$ theory, Th, then $\mathbf{M}$ is called a *model* of the FIX$_=$ theory if it satisfies all of the axioms of Th.

The Substitution Lemma

Lemma 3.2.3 The categorical semantics interprets the substitution of a term for a variable in a term via composition in the category. More precisely, if $\Gamma \vdash M_i : \alpha_i$ for $i = 1, \ldots, n$ and also $\Gamma' \vdash N(x_1, \ldots, x_n) : \beta$ where $\Gamma' = [x_1 : \alpha_1, \ldots, x_n : \alpha_n]$, we have

$$[\![\Gamma.N(\vec{M})]\!] = [\![\Gamma'.N(\vec{x})]\!] \circ \langle [\![\Gamma.M_1]\!], \ldots, [\![\Gamma.M_n]\!]\rangle.$$

Proof The proof is a routine induction on the structure of N. Note that in Section 3.3 we shall investigate the most general interpretation of the syntax of a FIX$_=$ theory subject to the requirement that substitution of syntax is modelled by composition of morphisms; this, in essence, supplies the details of the proof for terms N of computation or fixpoint type. □

The Soundness Theorem

The most important property of models $\mathbf{M}$ of FIX$_=$ theories is that any theorem of a theory must be satisfied by $\mathbf{M}$. Indeed we have the

Theorem 3.2.4 ["FIX$_=$ Soundness"] Let $\mathcal{C}$ be a FIX category, Th a FIX$_=$ theory, and $\mathbf{M}$ a model of Th in $\mathcal{C}$. Then $\mathbf{M}$ satisfies any equation in context which is a theorem of Th.

Proof We have to check that the rules for deriving equations in context are closed with respect to satisfaction by $\mathbf{M}$. The details are omitted. □

3.3 Interpretations of Computation Types and the Fixpoint Type

The interpretation of most of the syntax of a FIX$_=$ theory is perfectly standard and well understood. However, the interpretation of computation types and the fixpoint type is relatively new. Thus we shall show that our interpretations are the most general we could hope for, given the proviso that substitution of terms in the syntax is to be modelled by composition of morphisms in the category theory. Let us suppose that we are given a FIX$_=$ signature Sg and a FIX category $\mathcal{C}$.

Modelling the Types

The interpretation of the computation types $T\alpha$ is forced. As noted on Page 31, the operation $A \mapsto TA$ is not a categorical property of the FIX category, but is specified as part of the definition. Thus we interpret the type $T\alpha$ by the object $[\![T\alpha]\!] \stackrel{\text{def}}{=} T[\![\alpha]\!]$. As for the type *fix*, we shall, for the time being, interpret it as an undefined object Ω.

Modelling Terms in Context of Computation Type

Consider the rules

$$\frac{\Gamma \vdash M\colon \alpha}{\Gamma \vdash \mathsf{Val}(M)\colon T\alpha} \qquad \frac{\Gamma \vdash E\colon T\alpha \quad \Gamma, x\colon \alpha \vdash F(x)\colon T\beta}{\Gamma \vdash \mathsf{Let}\,(E, F)\colon T\beta}$$

In the syntax, substitution must commute with term formation. We model substitution of terms, in the syntax, via composition of the interpreting morphisms in the category $\mathcal{C}$. Thus, to interpret the first rule, we need a natural transformation

$$\theta\colon \mathcal{C}(-, A) \to \mathcal{C}(-, TA).$$

Using the Yoneda Lemma, we have a bijection

$$[\mathcal{C}^{op}, \mathcal{S}et](\mathcal{C}(-, A), \mathcal{C}(-, TA)) \cong \mathcal{C}(A, TA),$$

and in particular the action of the components θ_C arise by post-composition with a morphism $f\colon A \to TA$. By definition, this morphism f will be η_A. To summarise,

$$[\![\Gamma.\mathsf{Val}(M)\colon T\alpha]\!] \stackrel{\text{def}}{=} \eta_{[\![\alpha]\!]}[\![\Gamma.M]\!].$$

Similarly, to interpret the second rule, we shall need a natural transformation which has components

$$\theta_C : \mathcal{C}(C, TA) \times \mathcal{C}(C \times A, TB) \longrightarrow \mathcal{C}(C, TB).$$

If we apply naturality to a morphism $\langle id, g\rangle : C \longrightarrow C \times TA$ we find that

$$\theta(g, f) = \theta(\pi_2, f(\pi_1 \times id_A))\langle id, g\rangle$$

where $\pi_1 : C \times TA \to C$ and $\pi_2 : C \times TA \to TA$. Thus we can soundly interpret the second rule with a natural transformation which has components

$$\theta_C : \mathcal{C}(C \times A, TB) \longrightarrow \mathcal{C}(C \times TA, TB).$$

Given a morphism $f : C \times A \to TB$ we shall denote the effect of the components of θ by $f \mapsto f^*$. Combining our results, we are led to the following definition

$$[\![\Gamma \vdash \mathsf{Let}\,(E, F)]\!] \stackrel{\text{def}}{=} f^*\langle id, e\rangle,$$

where by structural induction we already have

$$\begin{aligned} [\![\Gamma, x : \alpha.F(x)]\!] &= f \\ [\![\Gamma.E]\!] &= e. \end{aligned}$$

Modelling Equations in Context of Computation Type

Proposition 3.3.1 In order to soundly model the computation type equations we need exactly the structure of a let category.

Proof Recall Definition 1.3.1. We established above the necessity of the existence of the function

$$(-)^* : \mathcal{C}(C \times A, TB) \longrightarrow \mathcal{C}(C \times TA, TB)$$

which is natural in C. This amounts to asking that

$$(f(g \times id))^* = f^*(g \times id)$$

where $f : C \times A \to TB$ and $g : C' \to C$. This is precisely `LiftS`.
The computation type equations are

$$\begin{aligned} \Gamma \vdash \mathsf{Let}\,(\mathsf{Val}(M), F) &= F(M) && (3.1) \\ \Gamma \vdash \mathsf{Let}\,(E, u.\mathsf{Val}(u)) &= E && (3.2) \\ \Gamma \vdash \mathsf{Let}\,(\mathsf{Let}\,(E, F), G) &= \mathsf{Let}\,(E, u.\mathsf{Let}\,(F(u), G)) && (3.3) \end{aligned}$$

Working through the details, in order to soundly interpret equation 3.1 we need

$$f^*(id \times \eta)\langle id, m\rangle = f\langle id, m\rangle \qquad (3.4)$$

where $f: C \times A \to TB$ and $m: C \to A$. For this, it is clearly sufficient to ask that `LiftB` holds. For necessity, take

$$\pi_1: C \times A \to A \quad \pi_2: C \times A \to C.$$

Then we have

$$\begin{array}{rcl} f & = & f(\pi_2 \times id)\langle id, \pi_1 \rangle \\ \text{using an instance of 3.4} & = & (f(\pi_2 \times id))^*(id \times \eta)\langle id, \pi_1 \rangle \\ \text{using naturality of } (-)^* & = & f^*(\pi_2 \times id)(id \times \eta)\langle id, \pi_1 \rangle \\ & = & f^*(id \times \eta). \end{array}$$

In order to soundly interpret equation 3.2 we need

$$(\eta\pi_1)^*\langle id, e \rangle = e \tag{3.5}$$

where $e: C \to TA$ and $\pi_1: C \times A \to A$. It is sufficient for `LiftH` to hold. For necessity, take

$$\begin{array}{ll} \pi_2: C \times TA \to TA & \pi_3: C \times TA \to C \\ \pi_4: (C \times TA) \times A \to A. & \end{array}$$

Then we have

$$\begin{array}{rcl} (\eta\pi_1)^* & = & (\eta\pi_1)^*(\pi_3 \times id)\langle id, \pi_2 \rangle \\ \text{from naturality of } (-)^* & = & (\eta\pi_1(\pi_3 \times id))^*\langle id, \pi_2 \rangle \\ & = & (\eta\pi_4)^*\langle id, \pi_2 \rangle \\ \text{using an instance of 3.5} & = & \pi_2 \end{array}$$

as required.
Finally, in order to model equation 3.3 soundly, we need to ask that

$$g^*(id \times f^*)\langle id, \langle id, e \rangle\rangle = ((g(\pi_1 \times id))^*\langle id, f \rangle)^*\langle id, e \rangle \tag{3.6}$$

where

$$\begin{array}{ll} \pi_1: C \times A \to C & f: C \times A \to TB \\ g: C \times B \to TD & e: C \to TA. \end{array}$$

Now consider the projection morphisms

$$\begin{array}{ll} \pi_2: C \times TA \to C & \pi_3: C \times TA \to TA \\ \pi_4: (C \times TA) \times A \to (C \times TA). & \end{array}$$

Using naturality of $(-)^*$ the equation 3.6 becomes

$$g^*(id \times f^*)\langle id, \langle id, e \rangle\rangle = (g^*\langle \pi_1, f \rangle)^*\langle id, e \rangle. \tag{3.7}$$

It is certainly sufficient for equation `LiftA` to hold. Indeed, it is also necessary, as we now show:

$$\begin{array}{rcl}
(g^*\langle \pi_1, f\rangle)^* & = & (g^*\langle \pi_1, f\rangle)^*(\pi_2 \times id)\langle id, \pi_3\rangle \\
\text{using naturality of } (-)^* & = & (g^*\langle \pi_1, f\rangle(\pi_2 \times id))^*\langle id, \pi_3\rangle \\
& = & (g^*\langle \pi_2\pi_4, f(\pi_2 \times id)\rangle)^*\langle id, \pi_3\rangle \\
& = & ((g(\pi_2 \times id))^*\langle \pi_4, f(\pi_2 \times id)\rangle)^*\langle id, \pi_3\rangle \\
\text{using equation 3.7} & = & (g(\pi_2 \times id))^*(id \times (f(\pi_2 \times id))^*)\langle id, \langle id, \pi_3\rangle\rangle \\
& = & g^*(id \times f^*)(\pi_2 \times (\pi_2 \times id))\langle id, \langle id, \pi_3\rangle\rangle \\
& = & g^*\langle \pi_2, f^*\rangle.
\end{array}$$

□

Modelling Terms in Context of Fixpoint Type

Consider the rules

$$\frac{}{\Gamma \vdash \omega : T\mathit{fix}} \qquad \frac{\Gamma \vdash E : T\mathit{fix}}{\Gamma \vdash \sigma(E) : \mathit{fix}} \qquad \frac{\Gamma, x : T\alpha \vdash F(x) : \alpha \quad \Gamma \vdash N : \mathit{fix}}{\Gamma \vdash \mathsf{It}_\alpha(F, N) : \alpha}$$

To interpret the first rule, we shall need a global element of $T\Omega$. (Remember that for the time being, Ω is just some arbitrary object of a let category). So we have

$$[\![\Gamma.\omega]\!] \stackrel{\text{def}}{=} \omega! : [\![\Gamma]\!] \to 1 \to T\Omega.$$

To interpret the second rule, with the usual assumptions about the way we shall model substitution, we shall need a natural transformation

$$\theta : \mathcal{C}(-, T\Omega) \to \mathcal{C}(-, \Omega).$$

Using the Yoneda Lemma, the effect of the components θ_C arise by post-composition with a morphism $\sigma : T\Omega \to \Omega$. Thus we have

$$[\![\Gamma.\sigma(E)]\!] \stackrel{\text{def}}{=} \sigma e,$$

where, by structural induction we already have $[\![\Gamma.E]\!] = e$.
To interpret the third rule, we shall need a natural transformation with components

$$\theta_C : \mathcal{C}(C \times TA, A) \times \mathcal{C}(C, \Omega) \longrightarrow \mathcal{C}(C, A)$$

Applying naturality to $\langle id, n\rangle : C \to C \times \Omega$, we get

$$\theta(f, n) = \theta(f(\pi_1 \times id), \pi_2)\langle id, n\rangle$$

where $\pi_1 : C \times \Omega \to C$ and $\pi_2 : C \times \Omega \to \Omega$. Thus we can soundly model the third rule with a natural transformation which has components

$$\theta_C : \mathcal{C}(C \times TA, A) \longrightarrow \mathcal{C}(C \times \Omega, A),$$

and we shall write $f \mapsto f^*$ for the effect of this function. Thus we are led to setting

$$[\![\Gamma.\mathsf{It}(F,N)]\!] \stackrel{\text{def}}{=} f^*\langle id, n\rangle,$$

where we already have

$$\begin{aligned} [\![\Gamma, x{:}T\alpha.F(x)]\!] &= f \\ [\![\Gamma.N]\!] &= n. \end{aligned}$$

Modelling Equations in Context of Fixpoint Type

Proposition 3.3.2 In order to soundly model the fixpoint type equations, we shall need exactly a let category $\mathcal{C}$ which is endowed with a FPO.

Proof The fixpoint type equations are

$$\Gamma \vdash \omega = \mathsf{Val}(\sigma(\omega)) \tag{3.8}$$

$$\Gamma \vdash E = \mathsf{Val}(\sigma(E)) \supset \Gamma \vdash E = \omega \tag{3.9}$$

$$\Gamma, e \vdash G(\sigma(e)) = F(\mathsf{Let}\,(e, n.\mathsf{Val}(G(n)))) \supset \Gamma \vdash G(N) = \mathsf{It}_\alpha(F,N) \tag{3.10}$$

$$\Gamma \vdash \mathsf{It}_\alpha(F, \sigma(E)) = F(\mathsf{Let}\,(E, n.\mathsf{Val}(\mathsf{It}_\alpha(F,n)))) \tag{3.11}$$

It is easy to see that for the sound modelling of 3.8 and 3.9 it is necessary and sufficient that the triple (ω, σ, Ω) forms part of the equaliser diagram of Definition 2.3.2 of a FPO.

Note that in the FIX logic, (modulo rules for which we have soundness), rule 3.10 is equivalent to

$$\Gamma, e \vdash G(\sigma(e)) = F(\mathsf{Let}\,(e, n.\mathsf{Val}(G(n)))) \supset \Gamma, n \vdash G(n) = \mathsf{It}_\alpha(F,n) \tag{3.12}$$

In order to soundly model 3.12 we see it is necessary that

$$\frac{g(id \times \sigma) = f\langle \pi_1, \mathit{lift}(\eta g)\rangle}{g = f^*}\,(*)$$

where $f{:}C \times TA \to A$, $g{:}C \times \Omega \to A$ and $\pi_1{:}C \times T\Omega \to C$; sufficiency of the uniqueness requirement of a FPO is immediate.

Finally, we look at the structure needed to soundly model 3.11. Put

$$\pi_2{:}C \times T\Omega \to T\Omega \qquad \pi_3{:}C \times \Omega \to C \qquad \pi_4{:}C \times \Omega \to \Omega.$$

Working through the details we shall need

$$f^*(id \times \sigma)\langle id, e\rangle = f\langle id, \mathit{lift}(\eta(f(\pi_3 \times id))^*\langle id, \pi_4\rangle)\langle id, e\rangle\rangle \tag{3.13}$$

where $e{:}C \to T\Omega$. However, we can appeal to $(*)$ to deduce that $(f(\pi_3 \times id))^* = f^*(\pi_3 \times id)$; thus 3.13 reduces to

$$f^*(id \times \sigma)\langle id, e\rangle = f\langle id, \mathit{lift}(\eta f^*)\langle id, e\rangle\rangle. \tag{3.14}$$

The universal property of a FPO is certainly sufficient to ensure that 3.14 holds. To see that it is also necessary, note that

$$\begin{aligned} f\langle \pi_1, \mathit{lift}(\eta f^*)\rangle &= f(\pi_1 \times id)\langle id, \mathit{lift}(\eta f^*(\pi_1 \times id))\langle id, \pi_2\rangle\rangle \\ &= f(\pi_1 \times id)\langle id, \mathit{lift}(\eta(f(\pi_1 \times id))^*)\langle id, \pi_2\rangle\rangle \\ \text{using an instance of 3.14} \quad &= (f(\pi_1 \times id))^*(id \times \sigma)\langle id, \pi_2\rangle \\ &= f^*(\pi_1 \times id)(id \times \sigma)\langle id, \pi_2\rangle \\ &= f^*(id \times \sigma). \end{aligned}$$

Finally note that the universal property of a FPO ensures Ω and σ are determined up to isomorphism. □

3.4 The Categorical Logic Correspondence

Now we have all the ingredients to describe the usual categorical logic correspondence for FIX$_=$ theories and FIX categories.

Proposition 3.4.1 Given a FIX category $\mathcal{C}$, we can define a certain FIX$_=$ theory, which we denote by $Th(\mathcal{C})$.

Proof The basic ground types are the objects of $\mathcal{C}$. For each morphism

$$f\colon A_1 \times \ldots \times A_n \longrightarrow B$$

in $\mathcal{C}$ there is a basic function symbol $f\colon A_1, \ldots, A_n \longrightarrow B$. There is clearly a canonical structure $\mathbf{G}$ for this signature in $\mathcal{C}$. The terms of the theory are then generated up according to the rules on Page 27. The axioms of $Th(\mathcal{C})$ are specified by

$$\Gamma \vdash M = N\colon \alpha \text{ is an axiom of } Th(\mathcal{C}) \qquad \text{iff} \qquad [\![\Gamma.M]\!]_{\mathbf{G}} = [\![\Gamma.N]\!]_{\mathbf{G}}.$$

□

Proposition 3.4.2 For each FIX$_=$ theory, Th, over some signature, Sg, we can construct a syntactic FIX category which we denote by $\mathcal{C}(Th)$.

Proof

- The objects of $\mathcal{C}(Th)$ are the types of the signature Sg.
- The morphisms with domain α and codomain β are specified by

  $$\mathcal{C}(Th)(\alpha, \beta) \stackrel{\text{def}}{=} \{M(x) \mid x\colon \alpha \vdash M(x)\colon \beta \}/\sim,$$

  where the equivalence relation $\sim$ is defined by

  $$M(x) \sim M'(y) \text{ iff } x\colon \alpha \vdash M(x) = M'(x)\colon \beta.$$

Composition is given by the usual substitution of terms; it is a tedious but straightforward task to check that this does define a FIX category. □

Theorem 3.4.3 Given a FIX category $\mathcal{C}$, then there is an equivalence of FIX categories

$$Eq : \mathcal{C}(Th(\mathcal{C})) \simeq \mathcal{C} : Eq^{-1}$$

where Eq and Eq^{-1} are FIX category morphisms.

Proof Define Eq and Eq^{-1} by setting

- $Eq(\alpha) \stackrel{\text{def}}{=} [\![\alpha]\!]_{\mathbf{G}}$ and $Eq^{-1}(A) \stackrel{\text{def}}{=} A$ on objects, and
- $Eq(M(x)) \stackrel{\text{def}}{=} [\![x.M(x)]\!]_{\mathbf{G}}$ and $Eq^{-1}(f) \stackrel{\text{def}}{=} f(x)$ on morphisms.

Note that Eq is well defined by appealing to Theorem 3.2.4. That we have an equivalence of categories via inverse FIX category morphisms is a lengthy calculation which is omitted. □

3.5 Definability of Fixpoints

Fixpoints in FIX$_=$

The fixpoint type is so called because in its presence one can always define fixpoint terms at all types of the form $\alpha \to T\beta$. We make this precise in the next proposition:

Proposition 3.5.1 ["Fixpoint Definability"] In the presence of a fixpoint object, one may define expressions $\mathsf{Y}_{\alpha,\beta}$ of the meta λ calculus with arity TERM $\to$ TERM for which given a FIX term in context $\Gamma \vdash F : (\alpha \to T\beta) \to \alpha \to T\beta$ we may derive

$$\Gamma \vdash \mathsf{Y}_{\alpha,\beta}(F) : \alpha \to T\beta \quad \text{and} \quad \Gamma \vdash F\mathsf{Y}_{\alpha,\beta}(F) = \mathsf{Y}_{\alpha,\beta}(F) : \alpha \to T\beta.$$

Proof We define $\mathsf{Y}_{\alpha,\beta}$ by giving the representative

$$\mathsf{Y}_{\alpha,\beta}(f) \stackrel{\text{def}}{=} \mathsf{It}_{\alpha \to T\beta}(e.\lambda_\alpha(x.\mathsf{Let}\,(e, y.fyx)), \sigma(\omega)).$$

It is easy to see that the first judgement is derivable. For the second let us put $G \stackrel{\text{def}}{=} e.\lambda_\alpha(x.\mathsf{Let}\,(e, y.fyx))$. Then

$$\begin{aligned}
\mathsf{Y}_{\alpha,\beta}(F) &= \mathsf{It}_{\alpha \to T\beta}(e.\lambda_\alpha(x.\mathsf{Let}\,(e, y.Fyx)), \sigma(\omega)) \\
&= \lambda_\alpha(x.\mathsf{Let}\,(\mathsf{Let}\,(\omega, n.\mathsf{Val}(\mathsf{It}_{\alpha \to T\beta}(G, n))), y.Fyx)) \\
&= \lambda_\alpha(x.\mathsf{Let}\,(\mathsf{Let}\,(\mathsf{Val}(\sigma(\omega)), n.\mathsf{Val}(\mathsf{It}_{\alpha \to T\beta}(G, n))), y.Fyx)) \\
&= \lambda_\alpha(x.F\mathsf{It}_{\alpha \to T\beta}(G, \sigma(\omega))x) \\
&= F\mathsf{Y}_{\alpha,\beta}(F).
\end{aligned}$$

□

Fixpoints in $\omega\mathcal{C}po$

Using the categorical logic correspondence we can easily see that in a FIX category fixpoints of certain morphisms always exist. In order to illustrate this we apply the categorical equivalent of Proposition 3.5.1 on Fixpoint Definability to the FIX category $\omega\mathcal{C}po$.

Suppose that D is an ωcpo and that $D_\perp$ is its lifting. Write $[D, D_\perp]$ for the continuous maps from D to $D_\perp$. Then it is well known that any continuous map

$$\phi\colon [D, D_\perp] \to [D, D_\perp]$$

has a least fixpoint. Now the categorical version of Proposition 3.5.1 on Fixpoint Definability says that a fixpoint of ϕ should be given by $it(\phi)(\top)$ where we note that in $\omega\mathcal{C}po$ it is the case that $\omega \stackrel{\text{def}}{=} [\top]$, $\top = \sigma([\top])$ and $it(\phi)$ arises as the unique mediating morphism of the following diagram

$$\begin{array}{ccc} \Omega_\perp & \xrightarrow{\sigma} & \Omega \\ {\scriptstyle it(\phi)_\perp}\downarrow & & \downarrow{\scriptstyle it(\phi)} \\ [D, D_\perp]_\perp & \xrightarrow{\phi_\perp} & [D, D_\perp] \end{array}$$

It is easy to see, using the commutativity of the diagram, that given $n \in \Omega/\{\top\}$ we have $it(\phi)(n) = \phi^n(\perp)$ and that from the continuity of $it(\phi)$

$$it(\phi)(\top) = \bigvee_{n\in\mathbb{N}} \phi^n(\perp).$$

Thus $it(\phi)(\top)$ is exactly the least fixpoint of ϕ with respect to the order on $[D, D_\perp]$; of course the argument above is really the proof that $\omega\mathcal{C}po$ *has* a FPO.

A Category without a Fixpoint Object

It is not always the case that a concrete category of domains has a FPO even though the morphisms between lifted domains may be guaranteed to have fixpoints.

Definition 3.5.2 Let $On\mathcal{C}po$ be the category which has

- objects posets possessing suprema of *all* chains and
- morphisms monotone set functions; (we will call them *maps*).

It is easy to see that this category is indeed a let ccc when we regard the lifting functor as giving rise to a strong monad. It is the case that all maps $f\colon D_\perp \to D_\perp$

have a least fixpoint. For given any such map f, define for any ordinal α the element $f^\alpha(\bot)$ by

$$\begin{aligned} f^0(\bot) &= \bot \\ f^{\alpha+1}(\bot) &= f(f^\alpha(\bot)) \\ f^\lambda(\bot) &= \bigvee_{\alpha<\lambda} f^\alpha(\bot) \quad \text{where } \lambda \text{ is a limit ordinal.} \end{aligned}$$

This is a good definition. The only thing that is not trivial is the existence of $f^\lambda(\bot)$: the set $\{f^\alpha(\bot) \mid \alpha < \lambda\}$ must be a chain. But a simple transfinite induction shows that $\alpha \leq \alpha' < \lambda$ implies $f^\alpha(\bot) \leq f^{\alpha'}(\bot)$. Next note that there is some ordinal β for which $f^\beta(\bot)$ is a least fixpoint of f. Suppose this were not the case. By appealing to Hartog's Lemma, we can find a least ordinal (say γ) whose cardinality is strictly greater than the cardinality of the domain of f, say $|D_\bot|$. Using the hypothesis, it must be the case that the cardinality of $\{f^\alpha(\bot) \mid \alpha \leq \gamma\}$ is strictly greater than $|D_\bot|$, a contradiction.

Proposition 3.5.3 The let ccc $\mathcal{OnCpo}$ does not possess a fixpoint object.

Proof As $\mathcal{OnCpo}$ is cartesian closed we may appeal to Lemma 2.3.3.

Let us suppose that the fixpoint object Ω exists in $\mathcal{OnCpo}$. We write $i\colon \Omega \hookrightarrow \Omega_\bot$, $n \mapsto [n]$ for the (monic) component of the unit of the lifting monad at Ω. From the equaliser condition of the fixpoint object, $i\sigma\colon \Omega_\bot \to \Omega_\bot$ has a unique fixpoint (which is not bottom), say $[\top] \in \Omega_\bot$. From remarks above there is an ordinal β for which $(i\sigma)^\beta(\bot) = [\top]$; note that $\beta > 1$.

We write σ^1 for $\sigma(\bot)$, $\sigma^{\alpha+1}$ for $\sigma[\sigma^\alpha]$ and σ^λ for $\bigvee_{\alpha<\lambda} \sigma^\alpha$; note that each supremum exists.

The ordinals $\omega + 2$ and $\omega + 1$ are objects of $\mathcal{OnCpo}$; we take $\omega \stackrel{\text{def}}{=} \{1, 2, \ldots\}$. Define a map $f\colon (\omega+2)_\bot \to \omega + 2$ by setting

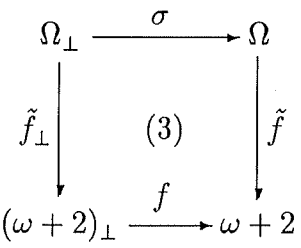
$$f(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x = \bot \\ n+1 & \text{if } x = [n] \\ \omega & \text{if } x = [\omega] \\ \omega+1 & \text{if } x = [\omega+1] \end{cases}$$

and a map $f_r\colon (\omega+1)_\bot \to \omega + 1$ to be f with restricted (co)domain.

Consider the diagram

$$\begin{array}{ccc} \Omega_\bot & \xrightarrow{\sigma} & \Omega \\ \downarrow{\scriptstyle \tilde{f}_{r\bot}} & (1) & \downarrow{\scriptstyle \tilde{f}_r} \\ (\omega+1)_\bot & \xrightarrow{f_r} & \omega+1 \\ \downarrow{\scriptstyle j_\bot} & (2) & \downarrow{\scriptstyle j} \\ (\omega+2)_\bot & \xrightarrow{f} & \omega+2 \end{array}$$

(1) commutes using the universal property of a FPO, (2) trivially. Similarly, there is $\tilde{f}: \Omega \to \omega + 2$ making



commute. From uniqueness, $\tilde{f} = j\tilde{f}_r$. We shall now consider the cases when β is strictly less than, or greater than, ω. In the latter case we obtain a contradiction by defining a map different from $\tilde{f}$ which makes (3) commute.

(*Case* $1 < \beta < \omega$): By chasing (1) we see that for any finite ordinal k, $\tilde{f}_r(\sigma^k) = k$. From the definition of β it is easy to see that $\sigma^\beta = \top$. Chasing (1) at $[\top] \in \Omega_\perp$ we get

$$\beta = \tilde{f}_r(\top) = \tilde{f}_r\sigma([\top]) = f_r\tilde{f}_{r\perp}([\top]) = f_r([\tilde{f}_r(\top)]) = \beta + 1$$

a contradiction. Hence we must have

(*Case* $\omega \leq \beta$): Certainly $\sigma^k \leq \sigma^\omega$ for any finite ordinal k, and so $k = \tilde{f}_r(\sigma^k) \leq \tilde{f}_r(\sigma^\omega)$. Hence $\tilde{f}_r(\sigma^\omega) = \omega$. As $\sigma^\beta = \top$ we have $\omega = \tilde{f}_r(\sigma^\omega) \leq \tilde{f}_r(\top)$ and so $\tilde{f}_r(\top) = \omega$.

Define a function $F: \Omega \to \omega + 2$ by

$$F(n) \stackrel{\text{def}}{=} \begin{cases} \omega + 1 & \text{if } \top \leq n \\ \tilde{f}(n) & \text{otherwise} \end{cases}$$

Note that $F(\top) = \omega + 1$ and $\tilde{f}(\top) = j\tilde{f}_r(\top) = \omega$, that is $F \neq \tilde{f}$. It is the case that F is monotone:

1. On elements of Ω not greater than $\top$, F is monotone for $\tilde{f}$ is.
2. If $\top \leq n \leq n'$ in Ω then $F(n) = F(n')$.
3. If $n \leq n'$ in Ω where n' is greater than $\top$ but n is not, then $F(n) \leq \omega + 1 = F(n')$.

Finally, F makes the square (3) commute.

1. Clearly $\sigma^1 \leq \top$ and σ^1 is not $\top$. So $F\sigma(\perp) = \tilde{f}(\sigma^1) = 1 = fF_\perp(\perp)$.
2. Suppose $[n] \in \Omega_\perp$ and n is greater than $\top$. Then $\top = \sigma([\top]) \leq \sigma([n])$. So
$$F\sigma([n]) = \omega + 1 = f([F(n)]) = fF_\perp([n]).$$
3. Suppose $[n] \in \Omega_\perp$ and n is not greater than $\top$. Then $\sigma([n])$ is not greater than $\top$, for if so, as σ is an isomorphism, $[\top] = \sigma^{-1}(\top) \leq [n]$ and this is not so. Hence,
$$F\sigma([n]) = \tilde{f}(\sigma([n])) = f(\tilde{f}_\perp([n])) = f([\tilde{f}(n)]) = f([F(n)]) = fF_\perp([n]).$$

By the uniqueness criterion, we have $F = \tilde{f}$. This is not so. Hence there can be no fixpoint object in *OnCpo*, as claimed. □

3.6 Functional Completeness

A concept which is closely related to the correspondence between $\text{FIX}_=$ theories and FIX categories is *functional completeness*. We begin by defining the notion of functional completeness for let ccc's and show these categories are functionally complete. We refer to [LS86] for a similar discussion concerning cartesian closed categories.

Definition 3.6.1 Let $\mathcal{C}$ be a let ccc and let A be an object of $\mathcal{C}$. The *polynomial category* $\mathcal{C}[X]$ *in variable* X is specified by the following data:

- The objects of $\mathcal{C}[X]$ are just the objects of $\mathcal{C}$ and
- a morphism of $\mathcal{C}[X]$ is an equivalence class, obtained by quotienting the collection consisting of the morphisms of $\mathcal{C}$ together with an indeterminate global element $X\colon 1 \to A$ by the equivalence relation $\sim$, where $\sim$ is the least such relation satisfying:

1. If $fg = h$ in $\mathcal{C}$, then $gf \sim h$.
2. If $u \sim u'$ and $v \sim v'$, then $vu \sim v'u'$.
3. $u \circ id \sim u \sim id \circ u$.
4. $(wv)u \sim w(vu)$.
5. Conditions forcing (formal) cartesian closure of $\mathcal{C}[X]$; see [LS86].
6. If $\mathit{lift}(f) = h$ in $\mathcal{C}$, then $\mathit{lift}(f) \sim h$ in $\mathcal{C}[X]$.
7. If $u \sim u'$ then $\mathit{lift}(u) \sim \mathit{lift}(u')$.
8. Conditions forcing $\mathcal{C}[X]$ to be a let category; for example, this amounts to requiring
$$\mathit{lift}(v(u \times id)) \sim \mathit{lift}(v)(u \times id),$$
corresponding to `LiftS` of Definition 1.3.2; similar conditions hold for the other let structure equations.

Remark 3.6.2 The polynomial category in n variables, $\mathcal{C}[X_1 \ldots X_n]$, is defined by iterating the construction just given.

Now we prove a version of functional completeness for let ccc's. If we assume that we have already the categorical logic correspondence for let ccc's, then we may prove the next theorem using internal languages. Alternatively, the theorem can be proved with bare hands, and the categorical logic developed in the same way that Lambek and Scott prove an equivalence between ccc's and typed lambda calculus.

Theorem 3.6.3 ["Functional Completeness"] Let $\mathcal{C}$ be a let category and $\mathcal{C}[X]$ its polynomial category. For every morphism $u\colon B \to C$ in $\mathcal{C}[X]$ there is a unique morphism

$$\Lambda(u)\colon A \times B \to C$$

in $\mathcal{C}$ such that

$$\Lambda(u)\langle X!, id\rangle = u$$

where $!\colon B \to 1$. In particular, each global element of C in $\mathcal{C}[X]$ is of the form fX for some unique morphism $f\colon A \to C$ in $\mathcal{C}$. Note that we now write $=$ for $\sim$, that is, for equality in the polynomial category.

Proof As there is a proof of this result for ccc's in [LS86], we just give details of the proof for the let structure. Take morphisms

$$\begin{aligned} u &: B \times B' \to TC, \\ \text{and hence} \quad \mathit{lift}(u) &: B \times TB' \to TC, \\ l &: (A \times B) \times B' \to A \times (B \times B'), \\ m &: A \times (B \times TB') \to (A \times B) \times TB', \end{aligned}$$

where l and m are the obvious isomorphisms. By induction, we shall assume that there is a morphism

$$\Lambda(u)\colon A \times (B \times B') \to TC$$

that satisfies functional completeness. Then we define

$$\Lambda(\mathit{lift}(u)) \stackrel{\text{def}}{=} \mathit{lift}(\Lambda(u)l)m,$$

and refer the reader to [LS86] for the definition of Λ on the ccc structure. We have to check that this is well defined, i.e. if $u = v$ in $\mathcal{C}[X]$ then $\Lambda(u) = \Lambda(v)$ in $\mathcal{C}[X]$. This amounts to showing that Λ respects each of conditions (1) to (8) of Definition 3.6.1; most of these are easy to verify, but we check in detail one of the let structure equations of condition (8), namely $\mathit{lift}(v\langle u\pi, \pi'\rangle) = \mathit{lift}(v)\langle u\pi, \pi'\rangle$. We have

$$\begin{aligned} \Lambda(\mathit{lift}(v\langle u\pi, \pi'\rangle)) &= \mathit{lift}(\Lambda(v\langle u\pi, \pi'\rangle)l)m \\ \text{by definition is} \quad &= \mathit{lift}(\Lambda(v)\langle \pi, \langle \Lambda(u)\langle \pi, \pi\pi'\rangle, \pi'\pi'\rangle\rangle\langle \pi\pi, \langle \pi'\pi, \pi'\rangle\rangle)m \\ \text{which is} \quad &= \mathit{lift}(\Lambda(v)\langle \pi\pi, \langle \Lambda(u)\pi, \pi'\rangle\rangle)m \\ \text{for appropriate } l' \quad &= \mathit{lift}(\Lambda(v)l'(\langle \pi, \Lambda(u)\rangle \times id))m \\ \text{from the let structure} \quad &= \mathit{lift}(\Lambda(v)l')(\langle \pi, \Lambda(u)\rangle \times id)m \\ &= \mathit{lift}(\Lambda(v)l')m'\langle \pi, \langle \Lambda(u)\langle \pi, \pi\pi'\rangle, \pi'\pi'\rangle\rangle \\ &= \Lambda(\mathit{lift}(v))\langle \pi, \Lambda(\langle u\pi, \pi'\rangle)\rangle \\ &= \Lambda(\mathit{lift}(v)\langle u\pi, \pi'\rangle) \end{aligned}$$

which is what we had to show. Now that we know Λ is well defined we prove the result itself. Indeed,

$$\Lambda(\mathit{lift}(u))\langle X!, id\rangle = \mathit{lift}(\Lambda(u)l)m\langle X!, id\rangle$$

$$
\begin{aligned}
& = lift(\Lambda(u)l)(\langle X, id\rangle \times id) \\
\text{which as } \mathcal{C}[X] \text{ is a let ccc} & = lift((\Lambda(u)l)(\langle X!, id\rangle \times id)) \\
& = lift(\Lambda(u)\langle X!, id\rangle) \\
\text{and by induction} & = lift(u).
\end{aligned}
$$

Hence the proof is complete. □

Thus we have presented a proof of the functional completeness result by direct calculation. As remarked above, we can then use the result to derive the categorical logic correspondence. For an explanation of the general ideas, see [LS86].

Theorem 3.6.4 Given a FIX category $\mathcal{C}$, then there is an equivalence of FIX categories

$$Eq : \mathcal{C}(Th(\mathcal{C})) \simeq \mathcal{C},$$

where Eq is a FIX category morphism.

Proof(*Sketch*) The types of the theory $Th(\mathcal{C})$ are the objects of $\mathcal{C}$ and the terms of the theory $Th(\mathcal{C})$ of type A are defined to be global elements in $\mathcal{C}[X_1 \ldots X_n]$ of the object A. The rules of inference for finite products and lambda terms can be found in [LS86]. For the let structure suppose that we are given the terms $E\colon 1 \to TA$ and $M\colon 1 \to TB$ in $Th(\mathcal{C})$. Then the term $\mathsf{Let}\,(E, X.M)$ is defined to be

$$lift(\Lambda(M)i)\langle id, E\rangle\colon 1 \to TB$$

where i is the isomorphism $1 \times A \cong A \times 1$. Equality of terms is given by equality in $\mathcal{C}[X_1 \ldots X_n]$.
The category $\mathcal{C}(Th(\mathcal{C}))$ has objects the types of $Th(\mathcal{C})$ and morphisms terms of $Th(\mathcal{C})$ with one free variable. Now define Eq by the following recipe:

- Set $Eq(A) \stackrel{\text{def}}{=} A$ and
- given that $(X\colon A, M(X)\colon B) : A \to B$ in $\mathcal{C}(Th(\mathcal{C}))$ where of course $X\colon 1 \to A$ and $M(X)\colon A \to B$ in $\mathcal{C}[X]$, we put

  $$Eq(X\colon A, M(X)\colon B) \stackrel{\text{def}}{=} \Lambda(M(X) \circ X)\theta$$

  where $\theta\colon A \cong A \times 1$ in $\mathcal{C}$ and we have made use of Theorem 3.6.3. The remainder of the proof is routine calculation.

□

3.7 Further Results about FIX Categories and $\text{FIX}_=$ Theories

The Proof of Lemma 2.3.3

Proof Suppose that $\mathcal{C}$ is a let ccc and that the triple (ω, σ, Ω) is a FPO satisfying the hypotheses of Lemma 2.3.3. Let $f: C \times TA \to A$. We shall use the categorical logic correspondence and work with the internal language of $\mathcal{C}$. We need to find a unique morphism $\tilde{f}: C \times \Omega \to A$ such that

$$\tilde{f}(id \times \sigma) = f\langle \pi_1, \mathit{lift}(\eta \tilde{f})\rangle. \tag{3.15}$$

With a view to using the uniqueness property of the T algebra, we look for a morphism (say g) of the form $g: T(C \to A) \to (C \to A)$. Define such a g via the term in context

$$e: T(C \to A) \vdash \lambda_C(c.f(c, \mathsf{Let}\,(e, x.\mathsf{Val}(xc)))): (C \to A).$$

Let us write $F(e)$ for the raw term (in context e). From this, we can define $\tilde{f}$ by the term in context

$$c: C, n: \Omega \vdash \mathsf{It}(F, n)c: A.$$

Let $u: T\Omega$. Using the rules of the logic $\text{FIX}_=$ we get

$$\begin{aligned} c: C, u: T\Omega \vdash \mathsf{It}(F, \sigma(u))c &= F(\mathsf{Let}\,(u, y.\mathsf{Val}(\mathsf{It}(F, y))))c \\ \text{by definition of } F(e) &= f(c, \mathsf{Let}\,(\mathsf{Let}\,(u, y.\mathsf{Val}(\mathsf{It}(F, y))), x.\mathsf{Val}(xc))) \\ &= f(c, \mathsf{Let}\,(u, y.\mathsf{Let}\,(\mathsf{Val}(\mathsf{It}(F, y)), x.\mathsf{Val}(xc))) \\ &= f(c, \mathsf{Let}\,(u, y.\mathsf{Val}(\mathsf{It}(F, y)c))). \end{aligned}$$

This says exactly that the equation 3.15 holds in $\mathcal{C}$, as desired, with uniqueness immediate from the $\text{FIX}_=$ rules.

Conversely, given the existence of a T^C algebra for each C, just take C to be 1 to get the required T algebra. □

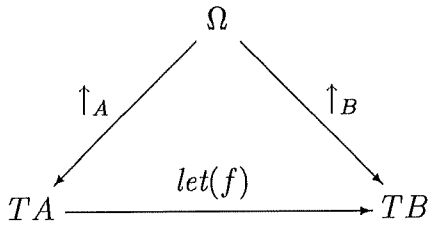
Elementary Domain Theoretic Features of $\text{FIX}_=$

$\text{FIX}_=$ has certain features which seem to be close in spirit to those of axiomatic domain theory and some properties of the abstract categorical semantics are very similar to those of the concrete model $\omega\mathcal{C}po$. We note certain results concerning $\text{FIX}_=$ which have well known analogues in $\omega\mathcal{C}po$ and begin with the following lemma:

Lemma 3.7.1 Work in an arbitrary FIX category. We can define a morphism $\uparrow_A: \Omega \to TA$ which is unique such that $\mathit{let}(\uparrow_A) = \uparrow_A \sigma$.

Proof Set $\uparrow_A \stackrel{\text{def}}{=} \mathit{it}(\mathit{let}(id_{TA}))$. Note that $\mathit{let}(id)\mathit{let}(\eta\uparrow_A) = \mathit{let}(\mathit{let}(id)\eta\uparrow_A) = \mathit{let}(\uparrow_A)$. Hence the result follows from the universal property of the FPO. □

Corollary 3.7.2 Given a FIX category $\mathcal{C}$ and a morphism $f: A \to TB$ then the following diagram commutes:



Proof Note that $let(let(f)\uparrow_A) = let(f)let(\uparrow_A) = let(f)\uparrow_A\sigma$. Then the result is immediate from Lemma 3.7.1. □

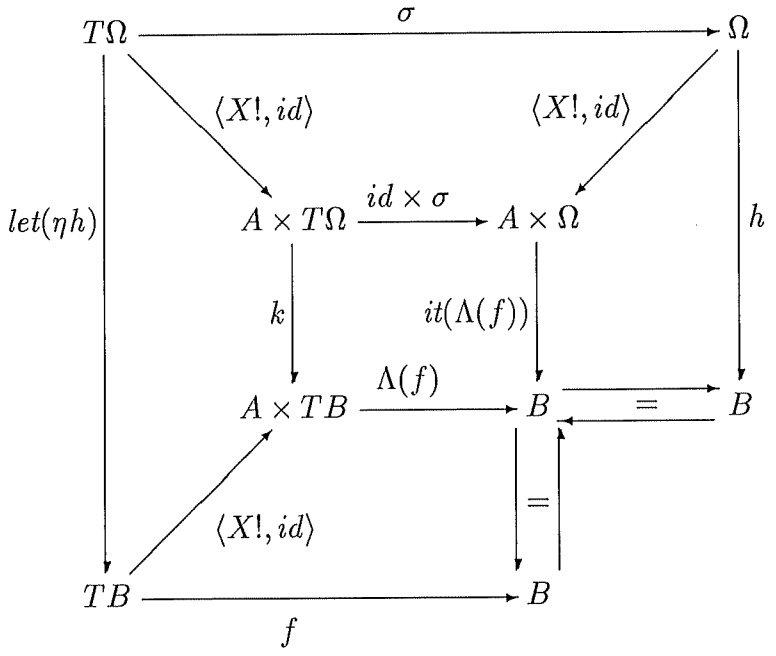
For any object A of the FIX category $\mathcal{C}$ there is a morphism $\bot_A: 1 \to TA$ which is defined to be $\uparrow_A\sigma\omega$. In the concrete FIX category $\omega\mathcal{C}po$, the morphism $\uparrow_A: \Omega \to A_\bot$ is of course the continuous function which is constantly bottom. Thus the morphism $\bot_A: 1 \to A_\bot$ just selects the bottom element from the domain $A_\bot$.

A question that remains unanswered is just how like axiomatic domain theory is the $\text{FIX}_=$ logic? For example, how do abstract fixpoints in $\text{FIX}_=$ relate to least fixpoints in the category of domains $\omega\mathcal{C}po$? We shall return to these issues in Chapter 10.

Fixpoint Objects in Polynomial Categories

Proposition 3.7.3 Let $\mathcal{C}$ be a FIX category with FPO Ω. Then $\mathcal{C}[X]$ also has a FPO, given by $i(\Omega)$, where $i : \mathcal{C} \hookrightarrow \mathcal{C}[X]$ is the canonical inclusion functor.

Proof We show that the category $\mathcal{C}[X]$ has a FPO by appealing to Lemma 2.3.3. Consider the diagram:



where $k \stackrel{\text{def}}{=} \langle \pi_A, \mathit{lift}(\eta \mathit{it}(\Lambda(f)))\rangle$. We set $h \stackrel{\text{def}}{=} \mathit{it}(\Lambda(f))\langle X!, id\rangle$, where the morphism $\Lambda(f)$ is defined by appealing to Theorem 3.6.3. In order to prove that $h\sigma = f\mathit{let}(\eta h)$ we inspect the diagram above. It is clear all regions commute except for $k\langle X!, id\rangle = \langle X!, id\rangle \mathit{let}(\eta h)$. To prove this it is enough to show that $\mathit{lift}(\eta \mathit{it}(\Lambda(f)))\langle X!, id\rangle = \mathit{let}(\eta \mathit{it}(\Lambda(f))\langle X!, id\rangle)$. We appeal to Theorem 3.6.3 and note

$$\begin{aligned}
\Lambda(\mathit{let}(\eta \mathit{it}(\Lambda(f))\langle X!, id\rangle)) &= \mathit{lift}(\Lambda(\eta \mathit{it}(\Lambda(f)))\langle \pi, \Lambda(\langle X!, id\rangle)\rangle) \\
&= \mathit{lift}(\eta\pi'\langle \pi, \mathit{it}(\Lambda(f))\pi'\rangle\langle \pi, \langle \pi, \pi'\rangle\rangle) \\
&= \mathit{lift}(\eta \mathit{it}(\Lambda(f))\langle \pi, \pi'\rangle).
\end{aligned}$$

It remains to prove uniqueness. Suppose also that $g\sigma = f\mathit{let}(\eta g)$. Note that if $\Lambda(g) = \mathit{it}(\Lambda(f))$ then $h = \mathit{it}(\Lambda(f))\langle X!, id\rangle = \Lambda(g)\langle X!, id\rangle = g$; so it is sufficient to prove the former. For this, if we can show $\Lambda(g)(id \times \sigma) = \Lambda(f)\langle \pi_A, \mathit{lift}(\eta\Lambda(g))\rangle$ we are done using the universal property of the FPO Ω in $\mathcal{C}$. As $\Lambda(g\sigma) = \Lambda(g)(id \times \sigma)$ it is sufficient to prove $\Lambda(f)\langle \pi_A, \mathit{lift}(\eta\Lambda(g))\rangle\langle X!, id\rangle = g\sigma$. We have

$$\begin{aligned}
\Lambda(f)\langle \pi_A, \mathit{lift}(\eta\Lambda(g))\rangle\langle X!, id\rangle &= \Lambda(f)\langle X!, \mathit{lift}(\eta\Lambda(g))\langle X!, id\rangle\rangle \\
\text{using uniqueness of } \Lambda(-) &= \Lambda(f)\langle X!, \mathit{let}(\eta g)\rangle \\
&= f\mathit{let}(\eta g)
\end{aligned}$$

and so we are done. Finally, the equaliser condition is simple to verify. □

Completeness of the Models of FIX$_=$ Theories

Theorem 3.7.4 The semantics which we have given to FIX$_=$ theories Th is both sound and complete. By complete, we mean that an equation in context is a theorem of Th just in case it is satisfied by all models of Th.

Proof We proved soundness on Page 35. An equation E is a theorem of the theory Th just in case it is satisfied by the generic model $\mathbf{G}$. But if E is satisfied by all models, in particular it is satisfied by $\mathbf{G}$. □

Remark 3.7.5 It is possible to check that for any FIX category $\mathcal{D}$ and model $\mathbf{M}$ of Th in $\mathcal{D}$ there is an essentially unique FIX category morphism $I\colon \mathcal{C}(Th) \to \mathcal{D}$ for which we have $I[\![\Gamma.M]\!]_{\mathbf{G}} = [\![\Gamma.M]\!]_{\mathbf{M}}$. In particular, if Th is the pure FIX$_=$ theory (no extralogical axioms), then $\mathcal{C}(Th)$ is (essentially) an initial object in the category of FIX categories and FIX category morphisms.

3.8 Gluing for Let Cartesian Closed Categories

The technique of gluing originated with Freyd, who presented a neat method for proving the existence and disjunction properties of certain intuitionistic type theories. His proof made use of a certain topos $\mathcal{E}$ manufactured from two other toposes; $\mathcal{E}$ is the so-called glued topos and for an account of this see [LS86]. The essence of

the method involves the careful application of a certain theorem from topos theory. A simple version of the theorem just for ccc's can be found in [Laf88]. We now prove a version of the theorem for let ccc's.

Lemma 3.8.1 ["Let ccc Gluing"] Let $\Gamma\colon \mathcal{D} \to \mathcal{C}$ be a functor where $\mathcal{D}$ is a let ccc and $\mathcal{C}$ is a ccc. Suppose also that $\mathcal{C}$ has pullbacks and that Γ preserves finite products. Write $\mathcal{G}l(\Gamma)$ for the category $(\mathcal{C}{\downarrow}\Gamma)$ and note that there is an obvious functor $\pi\colon (\mathcal{C}{\downarrow}\Gamma) \to \mathcal{D}$. Then it is the case that

1. $\mathcal{G}l(\Gamma)$ is a let ccc.
2. π is a morphism of let ccc's.

Proof It is a well known result that $\mathcal{G}l(\Gamma)$ is a ccc, for example see [Laf88]. We shall denote a typical morphism in $\mathcal{G}l(\Gamma)$ by

$$(a, s)\colon (A, f, X) \to (B, g, Y).$$

Note that the property of being a let ccc is not a categorical property. By the statement that $\mathcal{G}l(\Gamma)$ is a let ccc, we mean that there is an obvious canonical choice for a let structure. We make the following definitions:

1. $T(A, f, X) \stackrel{\text{def}}{=} (A, \Gamma(\eta)f, TX)$.
2. $\eta_{(A,f,X)} \stackrel{\text{def}}{=} (id, \eta)\colon (A, f, X) \longrightarrow T(A, f, X)$.
3. $$\frac{(a, s)\colon (A, f, X) \times (B, g, Y) \longrightarrow T(C, h, Z)}{\mathit{lift}(a, s) \stackrel{\text{def}}{=} (a, \mathit{lift}(s))\colon (A, f, X) \times T(B, g, Y) \longrightarrow T(C, h, Z)}$$

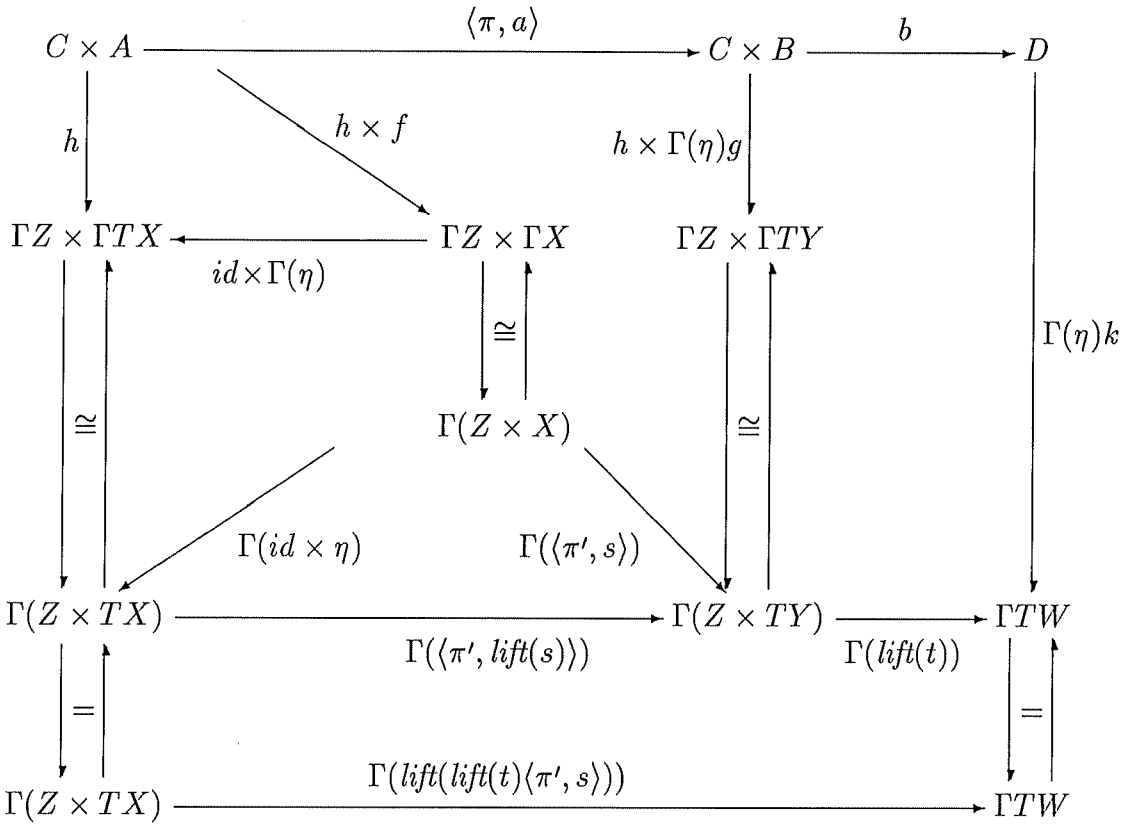
The first two definitions clearly make sense. We note that the third definition is good:

$$\begin{array}{ccccccc}
A \times B & \xrightarrow{f \times g} & \Gamma X \times \Gamma Y & \underset{\cong}{\rightleftarrows} & \Gamma(X \times Y) & \xrightarrow{\Gamma(id \times \eta)} & \Gamma(X \times TY) \\
\Big\downarrow{\scriptstyle a} & & & & \Big\downarrow{\scriptstyle \Gamma(s)} & & \Big\downarrow{\scriptstyle \Gamma(\mathit{lift}(s))} \\
C & \xrightarrow{h} & \Gamma Z & \xrightarrow{\Gamma(\eta)} & \Gamma TZ & \underset{=}{\rightleftarrows} & \Gamma TZ
\end{array}$$

We have to verify that the equations of Definition 1.3.2 on page 12 hold. We just check `LiftA`. Take morphisms

$$\begin{aligned}
&(a, s)\colon (C, h, Z) \times (A, f, X) \longrightarrow T(B, g, Y) \\
&(b, t)\colon (C, h, Z) \times (B, g, Y) \longrightarrow T(D, k, W) \\
&(\pi, \pi')\colon (C, h, Z) \times (A, f, X) \longrightarrow (C, h, Z).
\end{aligned}$$

Consider the commutative diagram:



From this, it is easy to see that

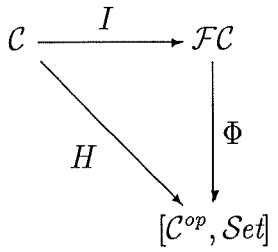
$$\begin{aligned} lift(lift(b,t) \circ \langle (\pi, \pi'), (a, s) \rangle) &= lift(lift(b,t) \circ (\langle \pi, a \rangle, \langle \pi', s \rangle)) \\ \text{and using the diagram} \quad &= lift(b,t) \circ (\langle \pi, a \rangle, \langle \pi', lift(s) \rangle) \\ &= lift(b,t) \circ \langle (\pi, \pi'), (a, lift(s)) \rangle. \end{aligned}$$

This is exactly `LiftA` for the glued category $\mathcal{G}l(\Gamma)$. It is immediate that π is a morphism of let ccc's. □

Let U be the forgetful functor from the category of locally small let ccc's to the category of locally small categories. There is, of course, a free functor F from the category of locally small categories to the category of locally small let ccc's, where on objects we write $\mathcal{F}\mathcal{C}$ for $F(\mathcal{C})$. Then F is left adjoint to U and we write I for the unit of the adjunction; thus there is for each locally small $\mathcal{C}$ a canonical functor $I_{\mathcal{C}}: \mathcal{C} \hookrightarrow \mathcal{F}\mathcal{C}$. Now we can prove

Corollary 3.8.2 Let $\mathcal{C}$ be a locally small category, and $\mathcal{F}\mathcal{C}$ the freely generated let ccc. Then the canonical functor $I: \mathcal{C} \hookrightarrow \mathcal{F}\mathcal{C}$ is full and faithful.

Proof Consider the commutative diagram



where H is the Yoneda embedding of $\mathcal{C}$ into its topos of presheaves, and the functor Φ arises (essentially) as the mate of H across the adjunction $(F \vdash U)$ where $[\mathcal{C}^{op}, \mathcal{S}et]$ is regarded as a let category with the identity strong monad. H is faithful, implying I is too.

Let Γ be the composite

$$I^* \circ H : \mathcal{FC} \longrightarrow [\mathcal{FC}^{op}, \mathcal{S}et] \longrightarrow [\mathcal{C}^{op}, \mathcal{S}et].$$

There is certainly a natural transformation $\alpha : \mathcal{C}(-,+) \longrightarrow \mathcal{FC}(I(-), I(+))$. So, for each A in $\mathcal{C}$ there is a natural transformation

$$\alpha_A : H_A \longrightarrow \Gamma \circ I(A) : \mathcal{C} \longrightarrow [\mathcal{C}^{op}, \mathcal{S}et].$$

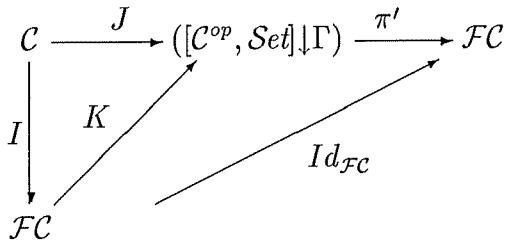
Thus we may define a functor

$$J : \mathcal{C} \longrightarrow ([\mathcal{C}^{op}, \mathcal{S}et]{\downarrow}\Gamma)$$

by setting $J(A) \stackrel{\text{def}}{=} \alpha_A$ and $J(f) \stackrel{\text{def}}{=} (f_*, \Gamma I(f))$. That J is well defined follows the naturality of α and that $([\mathcal{C}^{op}, \mathcal{S}et]{\downarrow}\Gamma)$ is a let ccc follows from Lemma 3.8.1.

If $\pi : ([\mathcal{C}^{op}, \mathcal{S}et]{\downarrow}\Gamma) \to [\mathcal{C}^{op}, \mathcal{S}et]$ and $\pi' : ([\mathcal{C}^{op}, \mathcal{S}et]{\downarrow}\Gamma) \to \mathcal{FC}$ are the usual projection functors, then clearly we have $\pi J = H$ and $\pi' J = I$.

The universal property of the category $\mathcal{FC}$ says that the following diagram commutes up to natural isomorphism,



where, say, $\iota : KI \cong J$. Now, $\pi'\iota : \pi' KI \cong \pi' J = I$. By the uniqueness condition of the universal property, there is a natural isomorphism $\kappa : \pi' K \cong Id_{\mathcal{FC}}$. It is also clearly the case that $\kappa_I = \pi'\iota$.

Now take objects A and B in $\mathcal{C}$ and a morphism $g: IA \to IB$ in $\mathcal{FC}$, and consider the diagram

$$\begin{array}{ccccc} H_A & = & \pi JA & \xrightarrow{\pi\iota_A^{-1}} & \pi KIA \\ & & \Big\downarrow{\scriptstyle \pi(\iota_B \circ Kg \circ \iota_A^{-1})} & & \Big\downarrow{\scriptstyle \pi Kg} \\ H_B & = & \pi JB & \xleftarrow{\pi\iota_B} & \pi KIB \end{array}$$

The Yoneda embedding is full, thus there is $f: A \to B$ in $\mathcal{C}$ for which $f_* = \pi(h)$ where we set $h \stackrel{\text{def}}{=} \iota_B \circ Kg \circ \iota_A^{-1}$. Now, for each object A in $\mathcal{C}$, we have a natural transformation

$$\alpha_A: H_A = \pi JA \longrightarrow \Gamma\pi' JA = \Gamma IA.$$

This leads to

$$\begin{aligned} \alpha_B f_* &= \alpha_B \circ \pi(h) \\ &= (\Gamma \circ \pi')(h) \circ \alpha_A \\ &= \Gamma(\kappa_{IB} \circ \pi' Kg \circ \pi' \iota_A^{-1}) \circ \alpha_A \\ &= \Gamma(g) \circ \alpha_A \\ &= H_g^{I(-)} \circ \alpha_A \end{aligned}$$

Hence it is the case that $\alpha_{B,A} \circ H_f^A = H_g^{IA} \circ \alpha_{A,A}$ and applying this to id_A we get

$$\begin{aligned} I(f \circ id_A \circ id_A) &= g \circ id_{IA} \circ id_{IA} \\ \text{that is } I(f) &= g, \end{aligned}$$

which says exactly that the functor I is full. □

The last result follows from an adaptation of Pitts' proof of the corresponding result for ccc's. We may derive from Corollary 3.8.2 the following

Proposition 3.8.3 Given a $\lambda\mathrm{ML}_T$ theory Th, any term M of ground type γ containing no object level variables is provably equal in the equational logic of $\lambda\mathrm{ML}_T$ to a ground term M' of type γ. (A ground term is simply a raw term of $\lambda\mathrm{ML}_T$ for which all function symbols are basic.)

Proof The term M corresponds to a global element of the denotation of γ in the classifying category of the theory Th. But by Corollary 3.8.2, this global element arises as a global element of the denotation of γ in the syntactic category arising from the ground types and terms. This says exactly that M is provably equal to a ground term M' in the logic of $\lambda\mathrm{ML}_T$. □

Presheaves on Categories with Monads

We finish this chapter with a miscellaneous result, namely:

Proposition 3.8.4 Let $\mathcal{C}$ be a locally small category and (T, η, μ) a monad over $\mathcal{C}$. Then the category of presheaves on $\mathcal{C}$ is a let category for a certain choice of monad.

Proof We define a strong monad (S, ϵ, ν, τ) on $[\mathcal{C}^{op}, \mathcal{S}et]$ by setting

- $SF(A) \stackrel{\text{def}}{=} F(TA)$ and $SF(f) \stackrel{\text{def}}{=} F(Tf)$,
- $(S\alpha)_A \stackrel{\text{def}}{=} \alpha_{TA}$,
- $(\epsilon_F)_A \stackrel{\text{def}}{=} F(\eta_A)$,
- $(\nu F)_A \stackrel{\text{def}}{=} F(\mu_A)$ and
- $(\tau_{(F,G)})_A \stackrel{\text{def}}{=} F(\eta_A) \times id_{G(TA)}$

where $f: A \to A'$ is a morphism of $\mathcal{C}$ and $\alpha: F \to G$ is a morphism of $[\mathcal{C}^{op}, \mathcal{S}et]$. It is routine to check that that we have defined a strong monad and appealing to Lemma 1.3.5 we are done. □

Chapter 4
The FIX Logical System

4.1 Why Introduce the FIX Logic?

The definition of an initial T algebra $\sigma: T\Omega \to \Omega$ for an endofunctor T contains both an existence and a uniqueness clause. The uniqueness requirement amounts to a form of induction principle, which is stated precisely in the following theorem for which a reference is [LS81].

Theorem 4.1.1 ["Initial T Algebra Induction Principle"] To show that a subobject $i: S \hookrightarrow \Omega$ is the whole of Ω, it suffices to show that the composition $\sigma Ti: TS \to \Omega$ factors through $i: S \hookrightarrow \Omega$. □

For the functor $(-)+1$ on the category of sets, the object part of the initial algebra is the natural numbers and the above theorem is equivalent to the usual principle of mathematical induction; this is a well known fact.

As another instance of initial T algebra induction, we consider the case when the category is $\omega\mathcal{C}po$, the endofunctor is lifting and subobjects are given by inclusive subsets. Suppose that $i: S \hookrightarrow \Omega$ is an inclusive subset of the fixpoint object $\Omega = \{0 < 1 < \ldots < \top\}$. Then we have that $S_\perp$ is just the inclusive subset of $\Omega_\perp$ given by $\{e \in \Omega_\perp \mid \forall n \in \Omega.[n] = e \supset n \in S\}$. Thus we can state Theorem 4.1.1 by way of following induction principle:

$$\frac{\forall e \in \Omega_\perp.(\forall n \in \Omega.[n] = e \supset n \in S) \supset \sigma(e) \in S}{\Omega = S}$$

Just as least fixed points are definable using the universal property of the initial $(-)_\perp$ algebra Ω, so is Scott's induction principle for least fixed points [Sco69b] derivable from the above induction rule.

Proposition 4.1.2 ["Scott Induction"] Let $P \subset D_\perp$ be an inclusive subset, and let $f: D_\perp \to D_\perp$ be a continuous function. Then we have

$$\frac{\perp \in P \qquad \forall d \in D_\perp.d \in P \supset f(d) \in P}{\mathit{fix}(f) \in P}$$

Proof In Theorem 4.1.1 take $S \stackrel{\text{def}}{=} \{n \in \Omega \mid h(n) \in P\}$ where h is the unique mediating morphism arising from $f_\perp: (D_\perp)_\perp \to D_\perp$ and the FPO Ω. Then $S = \Omega$ and so $\mathit{fix}(f) = h(\top) \in P$. □

In order to formulate this induction principle for the fixpoint type in $\text{FIX}_=$ we introduce a constructive logic [Bee85], called FIX, of properties of terms over $\text{FIX}_=$. There are similarities between FIX and the traditional "axiomatic domain theory" of LCF [Pau87] and to Plotkin's approach to denotational semantics using partial continuous functions [Plo85].

4.2 The Predicate Logic FIX

The FIX propositions constitute part of a predicate logic with equality. The rules for equality, conjunction and universal quantification (over elements of a given type) form a fragment of first-order intuitionistic predicate calculus [Dum77]. Additionally there are certain predicate constructors which implicitly contain forms of implication, disjunction and existential quantification. In order to set up a formal system for our logic, we begin by defining an extension of the notion of $\text{FIX}_=$ signature, which was defined in Section 2.5.1.

Signatures for FIX

Definition 4.2.1 A *FIX signature* Sg is specified as in Section 2.5.1, together with the following data:

- A collection of *basic relation symbols*, together with the following *distinguished relation symbols*: $=_\alpha$, true, false, $\&$, $\forall_\alpha$, $\Box$, $\Diamond$, $+$.
- A *sorting* for each of the basic relation symbols, which is a list of n types, and will be written:

  $$R\colon \alpha_1, \ldots, \alpha_n.$$

  In the case that n is one, we shall write $R\colon \alpha$. We say that R is an *n-ary* basic relation symbol when its sorting consists of n types.

We use the signature Sg to define the propositions of our logic. Given such a Sg, we shall define from this an abstract syntax signature Σ. The collection of ground arities, GAr, is the set $\{\text{TERM}, \text{PROP}\}$ and the collection of constants, Con, consists of all function symbols, all relation symbols, and a countably infinite set of object level variables. The function symbols have the same arities as those designated on Page 24 and the object level variables arity TERM. The n-ary relation symbols are considered to have arity $\text{TERM}^n \to \text{PROP}$, the distinguished relation symbols which will represent truth, falsity, equality, conjunction and universal quantification have their usual arities and the remaining distinguished relation symbols have the following arities:

1. $+\colon (\text{TERM} \to \text{PROP}) \to (\text{TERM} \to \text{PROP}) \to \text{TERM} \to \text{PROP}$
2. $\Box\colon \text{TERM} \to (\text{TERM} \to \text{PROP}) \to \text{PROP}$

3. $\Diamond$: TERM $\rightarrow$ (TERM $\rightarrow$ PROP) $\rightarrow$ PROP

Associated with a FIX signature *Sg* is a collection of *raw FIX terms* and *raw FIX propositions*. These are the closed expressions of the abstract syntax generated from Σ of arities TERM and PROP respectively.

Propositions in Context for FIX

We shall write

$$\Gamma \vdash \Phi \text{ prop}$$

for the judgement that given the context Γ the raw proposition Φ is well formed. These judgements are generated by the following rules, where in the case of the basic relation symbols, the sorting of a certain symbol is used to determine the form of the introduction rule.

Basic Relation Symbols

$$\frac{\Gamma \vdash M_1 : \alpha_1, \ldots, \Gamma \vdash M_n : \alpha_n}{\Gamma \vdash R(M_1, \ldots, M_n) \text{ prop}}$$

where R is a basic relation symbol with sorting $R : \alpha_1, \ldots, \alpha_n$

Equality Propositions

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash M' : \alpha}{\Gamma \vdash M =_\alpha M' \text{ prop}}$$

Truth

$$\frac{}{\Gamma \vdash \mathsf{true} \text{ prop}}$$

Falsity

$$\frac{}{\Gamma \vdash \mathsf{false} \text{ prop}}$$

Conjunction Propositions

$$\frac{\Gamma \vdash \Phi \text{ prop} \quad \Gamma \vdash \Psi \text{ prop}}{\Gamma \vdash \Phi \,\&\, \Psi \text{ prop}}$$

Coproduct Propositions

$$\frac{\Gamma, x : \alpha \vdash \Phi(x) \text{ prop} \quad \Gamma, y : \beta \vdash \Psi(y) \text{ prop} \quad \Gamma \vdash C : \alpha + \beta}{\Gamma \vdash (\Phi + \Psi)(C) \text{ prop}}$$

Universal Quantification Propositions

$$\frac{\Gamma, x{:}\alpha \vdash \Phi(x)\ \text{prop}}{\Gamma \vdash \forall_\alpha(\Phi)\ \text{prop}}$$

Universal Modality Propositions

$$\frac{\Gamma, x{:}\alpha \vdash \Phi(x)\ \text{prop} \quad \Gamma \vdash E{:}T\alpha}{\Gamma \vdash \Box(E, \Phi)\ \text{prop}}$$

Existential Modality Propositions

$$\frac{\Gamma, x{:}\alpha \vdash \Phi(x)\ \text{prop} \quad \Gamma \vdash E{:}T\alpha}{\Gamma \vdash \Diamond(E, \Phi)\ \text{prop}}$$

Remark 4.2.2 The usual rules for weakening of contexts, and substitution of raw terms for object level variables, are derivable from the above rules by simple structural induction.

Propositional Theories for FIX

Now that we have the propositional syntax for the FIX logical system, we present rules for deducing the validity of the propositions. These rules will be presented in a sequent natural deduction style. We will use a sequent in context as our basic judgement, which will take the form:

$$\Gamma, \Lambda \vdash \Phi.$$

Here, Λ is a finite set of propositions. The intended meaning of a judgement is that one has a deduction of Φ which involves a certain number of undischarged hypotheses, each of which must occur in the set Λ. We shall write $\Gamma, \Lambda, \Psi \vdash \Phi$ for $\Gamma, \Lambda \cup \{\Psi\} \vdash \Phi$ and in the case that Λ is empty, we simply omit the symbol Λ from the judgement. A *FIX theory*, *Th*, is specified by a FIX signature, together with a specific collection of sequents in context, which are called the *axioms* of *Th*. The collection of *theorems* of *Th* consists of the least collection of sequents in context which contains the axioms of *Th*, and is closed under the following rules:

Weakening

$$\frac{\Gamma, \Lambda \vdash \Phi}{\Gamma', \Lambda \vdash \Phi}\,(\texttt{wk}) \qquad \text{where } \Gamma \subset \Gamma'$$

Identity

$$\frac{\Gamma \vdash \Phi\ \text{prop} \quad \Gamma \vdash \Lambda\ \text{prop}}{\Gamma, \Lambda, \Phi \vdash \Phi}\,(\texttt{id})$$

Substitution

$$\frac{\Gamma,\ x{:}\alpha, \Lambda(x) \vdash \Phi(x) \quad \Gamma \vdash M{:}\alpha}{\Gamma,\ \Lambda(M) \vdash \Phi(M)}\ (\mathtt{sub})$$

Cut

$$\frac{\Gamma, \Lambda \vdash \Phi \quad \Gamma, \Phi, \Lambda' \vdash \Psi}{\Gamma, \Lambda, \Lambda' \vdash \Psi}\ (\mathtt{cut})$$

Equations

Every logical rule for deducing equations in $\mathrm{FIX}_{=}$ becomes a rule in the FIX logic, where judgements of the form $\Gamma \vdash M = M' : \alpha$ become judgements of the form $\Gamma \vdash M =_\alpha M'$.

Null Type Falsity Entailment

$$\frac{}{x{:}\mathit{null} \vdash \mathsf{false}}\ (\mathtt{null})$$

Truth Entailment

$$\frac{\Gamma \vdash \Lambda\ \mathrm{prop}}{\Gamma, \Lambda \vdash \mathsf{true}}\ (\top)$$

Falsity Entailment

$$\frac{\Gamma, \Lambda \vdash \mathsf{false} \quad \Gamma \vdash \Phi\ \mathrm{prop}}{\Gamma, \Lambda \vdash \Phi}\ (\bot)$$

Equality Entailment

$$\frac{}{x{:}\alpha \vdash x =_\alpha x}\ (=\mathtt{ref})$$

$$\frac{}{x{:}\alpha,\ x'{:}\alpha,\ x =_\alpha x' \vdash x' =_\alpha x}\ (=\mathtt{sym})$$

$$\frac{}{x{:}\alpha,\ x'{:}\alpha,\ x''{:}\alpha,\ x =_\alpha x',\ x' =_\alpha x'' \vdash x =_\alpha x''}\ (=\mathtt{tran})$$

$$\frac{\Gamma, x{:}\alpha \vdash M(x){:}\beta}{\Gamma, x{:}\alpha, x'{:}\alpha,\ x =_\alpha x' \vdash M(x) =_\beta M(x')}\ (=\mathtt{sub1})$$

$$\frac{\Gamma, x{:}\alpha \vdash \Phi(x)\ \mathrm{prop}}{\Gamma, x{:}\alpha, x'{:}\alpha,\ x =_\alpha x', \Phi(x) \vdash \Phi(x')}\ (=\mathtt{sub2})$$

Conjunction Entailment

$$\frac{\Gamma,\ \Lambda \vdash \Phi \quad \Gamma,\ \Lambda \vdash \Psi}{\Gamma,\ \Lambda \vdash \Phi\ \&\ \Psi}\ (\&\mathtt{i})$$

$$\frac{\Gamma,\ \Lambda \vdash \Phi\ \&\ \Psi}{\Gamma,\ \Lambda \vdash \Phi}\ (\&\mathtt{e}) \qquad \frac{\Gamma,\ \Lambda \vdash \Phi\ \&\ \Psi}{\Gamma,\ \Lambda \vdash \Psi}\ (\&\mathtt{e})$$

Universal Quantification Entailment

$$\frac{\Gamma, x{:}\alpha,\ \Lambda \vdash \Phi(x)}{\Gamma, \Lambda \vdash \forall_\alpha(\Phi)}\ (\forall\mathtt{i})$$

$$\frac{\Gamma,\ \Lambda \vdash \forall_\alpha(\Phi) \quad \Gamma \vdash M{:}\alpha}{\Gamma,\ \Lambda \vdash \Phi(M)}\ (\forall\mathtt{e})$$

Universal Modality Entailment

$$\frac{\Gamma, x{:}\alpha,\ \Lambda, \mathsf{Val}(x) =_{T\alpha} E \vdash \Phi(x)}{\Gamma, \Lambda \vdash \Box(E, \Phi)}\ (\Box\mathtt{i})$$

$$\frac{\Gamma,\ \Lambda \vdash \Box(E, \Phi) \quad \Gamma,\ \Lambda \vdash \mathsf{Val}(M) =_{T\alpha} E}{\Gamma, \Lambda \vdash \Phi(M)}\ (\Box\mathtt{e})$$

Existential Modality Entailment

$$\frac{\Gamma,\ \Lambda \vdash \mathsf{Val}(M) =_{T\alpha} E \quad \Gamma,\ \Lambda \vdash \Phi(M)}{\Gamma,\ \Lambda \vdash \Diamond(E, \Phi)}\ (\Diamond\mathtt{i})$$

$$\frac{\Gamma, x{:}\alpha,\ \Lambda, \mathsf{Val}(x) =_{T\alpha} E, \Phi(x) \vdash \Psi \quad \Gamma,\ \Lambda \vdash \Diamond(E, \Phi)}{\Gamma,\ \Lambda \vdash \Psi}\ (\Diamond\mathtt{e})$$

Coproduct Entailment

$$\frac{\Gamma,\ \Lambda \vdash \Phi(M) \quad \Gamma, y{:}\beta \vdash \Psi(y)\ \mathrm{prop}}{\Gamma,\ \Lambda \vdash (\Phi + \Psi)(\mathsf{Inl}_\beta(M))}\ (+\mathtt{i}) \qquad \frac{\Gamma,\ \Lambda \vdash \Psi(N) \quad \Gamma, x{:}\alpha \vdash \Phi(x)\ \mathrm{prop}}{\Gamma,\ \Lambda \vdash (\Phi + \Psi)(\mathsf{Inr}_\alpha(N))}\ (+\mathtt{i})$$

$$\frac{\left\{\begin{array}{l}\Gamma, \Lambda \vdash (\Phi + \Psi)(C) \\ \Gamma, x{:}\alpha,\ \Lambda,\ \mathsf{Inl}_\beta(x) =_{\alpha+\beta} C,\ \Phi(x) \vdash \Theta(F(x)) \\ \Gamma, y{:}\beta,\ \Lambda,\ \mathsf{Inr}_\alpha(y) =_{\alpha+\beta} C,\ \Psi(y) \vdash \Theta(G(y))\end{array}\right.}{\Gamma, \Lambda \vdash \Theta(\{F, G\}(C))}\ (+\mathtt{e})$$

Disjoint Sum Condition

$$\frac{}{\Gamma, \Lambda, \mathsf{Inl}_\beta(M) =_{\alpha+\beta} \mathsf{Inr}_\alpha(N) \vdash \mathsf{false}}\ (\mathtt{djsum})$$

Modality Condition

$$\frac{\Gamma,\ \Lambda \vdash \mathsf{Let}\,(E,F) =_{T\alpha} \mathsf{Val}(M)}{\Gamma,\Lambda \vdash \Diamond(E,x.F(x) =_{T\alpha} \mathsf{Val}(M))}\ (\mathtt{mod})$$

Nat Induction

$$\frac{\Gamma,\ \Lambda \vdash \Phi(\mathsf{O}) \quad \Gamma,n{:}\,nat,\ \Lambda,\Phi(n) \vdash \Phi(\mathsf{Suc}(n)) \quad \Gamma \vdash N{:}\,nat}{\Gamma,\ \Lambda \vdash \Phi(N)}\ (\mathtt{natin})$$

Fix Induction

$$\frac{\Gamma,e{:}\,Tfix,\ \Lambda,\Box(e,\Phi) \vdash \Phi(\sigma(e)) \quad \Gamma \vdash N{:}\,fix}{\Gamma,\ \Lambda \vdash \Phi(N)}\ (\mathtt{fixin})$$

This completes the rules for deriving sequents.

Informal Explanation of the FIX Propositions

The FIX logic has many features in common with intuitionistic predicate calculus; for the latter see [Dum77]. However, it introduces propositions of the form $\Box(e,\Phi)$, $\Diamond(e,\Phi)$, $(\Phi+\Psi)(z)$, and so we shall describe informally the intended meaning of this syntax.

For the universal modality, $\Box(e,\Phi)$, the intended meaning is

$$\forall x{:}\,\alpha.(\mathsf{Val}(x) = e \supset \Phi(x)),$$

which we read as "for all x of type α, if it is the case that e is provably equal to the value of x then necessarily $\Phi(x)$ holds."

For the existential modality, $\Diamond(e,\Phi)$, the intended meaning is

$$\exists x{:}\,\alpha.\mathsf{Val}(x) = e\ \&\ \Phi(x),$$

which we read as "it is possible that e is provably equal to $\mathsf{Val}(x)$ and that $\Phi(x)$ holds."

For coproduct propositions, $(\Phi+\Psi)(z)$, the intended meaning is

$$(\exists x{:}\,\alpha.z = \mathsf{Inl}_\beta(x)\ \&\ \Phi(x)) \vee (\exists y{:}\,\beta.z = \mathsf{Inr}_\alpha(y)\ \&\ \Psi(y)).$$

which we read as "it is either the case that z is provably equal to $\mathsf{Inl}_\beta(x)$ and that $\Phi(x)$ holds, or it is the case that z is provably equal to $\mathsf{Inr}_\alpha(y)$ and that $\Psi(y)$ holds."

Remark 4.2.3 Each of the terms $F^N(M)$ and $\mathsf{It}_\alpha(F,N)$ is unique up to provable equality in the FIX logic. In $\mathrm{FIX}_=$ it is necessary to impose rules which make uniqueness explicit. However, in the FIX logic, this uniqueness is derivable from

the rules for Nat and Fix Induction. Consider, for example, the uniqueness rule for the fixpoint type. Set $\Phi(n) \stackrel{\text{def}}{=} G(n) =_\alpha \mathsf{It}(F, n)$ where $n{:}\mathit{fix}$. Using the FIX rules we may deduce that

$$\begin{cases} \Gamma, e{:}T\mathit{fix}, n{:}\mathit{fix}, \Lambda, e = \mathsf{Val}(n), \\ \Box(e, \Phi) \vdash F(\mathsf{Let}\,(\mathsf{Val}(n), u.\mathsf{Val}(G(u)))) = F(\mathsf{Let}\,(\mathsf{Val}(n), u.\mathsf{Val}(\mathsf{It}(F, u)))) \end{cases}$$

and this implies $\Gamma, e, \Lambda, \Box(e, \Phi) \vdash \Phi(\sigma(e))$; we are done by (`fixin`).

4.3 Adjoint Style Formulation of the FIX Logic

The Adjoint Rules

The FIX logic can be presented using rules which are closely related to the categorical semantics given in Chapter 5. The new system is given by substituting the following rules for their counterparts in the FIX logic.

Equality Entailment

$$\frac{\Gamma,\ x{:}\alpha,\ x'{:}\alpha, \Lambda,\ x =_\alpha x' \vdash \Phi(x')}{\Gamma,\ x{:}\alpha,\ \Lambda \vdash \Phi(x)}\ (=\mathsf{ad})$$

Conjunction Entailment

$$\frac{\Gamma, \Lambda \vdash \Phi \quad \Gamma, \Lambda \vdash \Psi}{\Gamma, \Lambda \vdash \Phi\ \&\ \Psi}\ (\&\mathsf{ad})$$

Universal Quantification Entailment

$$\frac{\Gamma,\ x{:}\alpha, \Lambda \vdash \Phi(x)}{\Gamma, \Lambda \vdash \forall_\alpha(\Phi)}\ (\forall\mathsf{ad})$$

Universal Modality Entailment

$$\frac{\Gamma,\ x{:}\alpha, \Lambda(\mathsf{Val}(x)) \vdash \Phi(x)}{\Gamma,\ e{:}T\alpha, \Lambda(e) \vdash \Box(e, \Phi)}\ (\Box\mathsf{ad})$$

Existential Modality Entailment

$$\frac{\Gamma,\ x{:}\alpha, \Lambda(\mathsf{Val}(x)),\ \Phi(x) \vdash \Psi(\mathsf{Val}(x))}{\Gamma,\ e{:}T\alpha, \Lambda(e),\ \Diamond(e, \Phi) \vdash \Psi(e)}\ (\Diamond\mathsf{ad})$$

Coproduct Entailment

$$\frac{\Gamma,\ x{:}\alpha, \Lambda(\mathsf{Inl}_\beta(x)),\ \Phi(x) \vdash \Theta(F(x)) \quad \Gamma,\ y{:}\beta, \Lambda(\mathsf{Inr}_\alpha(y)),\ \Psi(y) \vdash \Theta(G(y))}{\Gamma,\ z{:}\alpha+\beta, \Lambda(z),\ (\Phi+\Psi)(z) \vdash \Theta(\{F, G\}(z))}\ (+\mathsf{ad})$$

Equivalence of the Systems

Lemma 4.3.1 The following are derived rules in the FIX logic, where $\Gamma \vdash M : \alpha$ is a FIX term in context:

$$\frac{\Gamma, \Lambda \vdash \Phi(M)}{\Gamma, x, \Lambda, x = M \vdash \Phi(x)}\,(\texttt{ts}) \qquad \frac{\Gamma, \Lambda(M) \vdash \Phi}{\Gamma, x, \Lambda(x), x = M \vdash \Phi}\,(\texttt{es})$$

Proof We omit the proof of rule (ts); for rule (es), note that the backwards direction follows from (sub). For the forwards direction, it suffices to show the case when Λ is a single proposition. Indeed, we have:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x \vdash \Theta(x)\ \text{prop}}\,(\texttt{hyp})}{\Gamma, x, x', \Theta(x), x = x' \vdash \Theta(x')}\,(\texttt{sub2})}{\Gamma, x, \Theta(x), x = M \vdash \Theta(M)}\,(\texttt{sub}) \qquad \dfrac{}{\Gamma, x, \Theta(M) \vdash \Phi}\,(\texttt{hyp})}{\Gamma, x, \Theta(x), x = M \vdash \Phi}\,(\texttt{cut})$$

□

Theorem 4.3.2 The original FIX logic and the system defined on Page 66 are equivalent.

Proof The proof of the equivalence for equality, conjunction and universal quantification is well known. We give details for the remaining forms of proposition: (*Case* (□i)(□e) *imply* (□ad)):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\texttt{hyp})}{\Gamma, e, \Lambda(e) \vdash \Box(e, \Phi)}}{\Gamma, x, e, \Lambda(e), e = \mathsf{Val}(x) \vdash \Box(e, \Phi)} \qquad \dfrac{}{\Gamma, x, e, \Lambda(e), e = \mathsf{Val}(x) \vdash e = \mathsf{Val}(x)}}{\Gamma, x, e, \Lambda(e), e = \mathsf{Val}(x) \vdash \Phi(x)}\,(\Box\texttt{e})}{\Gamma, x, \Lambda(\mathsf{Val}(x)), \mathsf{Val}(x) = \mathsf{Val}(x) \vdash \Phi(x)}\,(\texttt{sub})}{\Gamma, x, \Lambda(\mathsf{Val}(x)) \vdash \Phi(x)}$$

$$\dfrac{\dfrac{\dfrac{(\texttt{hyp})}{\Gamma, x, \Lambda(\mathsf{Val}(x)) \vdash \Phi(x)}}{\Gamma, x, e, \Lambda(e), e = \mathsf{Val}(x) \vdash \Phi(x)}\,(\texttt{es})}{\Gamma, e, \Lambda(e) \vdash \Box(e, \Phi)}\,(\Box\texttt{i})$$

where (es) was established in Lemma 4.3.1.

(*Case* ($\Box$ad) *implies* ($\Box$e)($\Box$i)):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash \Box(E, \Phi)}}{\Gamma, e', \Lambda, e' = E \vdash \Box(e', \Phi)}\,(\mathtt{ts})}{\Gamma, x, \Lambda, \mathsf{Val}(x) = E \vdash \Phi(x)}\,(\Box\mathtt{ad})}{\Gamma, \Lambda, \mathsf{Val}(M) = E \vdash \Phi(M)}\,(\mathtt{sub}) \qquad \dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash \mathsf{Val}(M) = E}}{\Gamma, \Lambda \vdash \Phi(M)}\,(\mathtt{cut})$$

$$\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, x, \Lambda, \mathsf{Val}(x) = E \vdash \Phi(x)}}{\Gamma, e, \Lambda, e = E \vdash \Box(e, \Phi)}\,(\Box\mathtt{ad})}{\Gamma, \Lambda \vdash \Box(E, \Phi)}$$

(*Case* ($\Diamond$i)($\Diamond$e) *imply* ($\Diamond$ad)):

$$\dfrac{\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, x, \Lambda(\mathsf{Val}(x)), \Phi(x) \vdash \Psi(\mathsf{Val}(x))}}{\Gamma, e, x, \Lambda(e), \Phi(x), e = \mathsf{Val}(x) \vdash \Psi(e)}}{\Gamma, e, x, \Lambda(e), \Diamond(e, \Phi), \Phi(x), e = \mathsf{Val}(x) \vdash \Psi(e)} \qquad \dfrac{}{\Gamma, e, \Lambda(e), \Diamond(e, \Phi) \vdash \Diamond(e, \Phi)}}{\Gamma, e, \Lambda(e), \Diamond(e, \Phi) \vdash \Psi(e)}$$

where the last rule is ($\Diamond$e).

$$\dfrac{\dfrac{\dfrac{}{\Gamma, e, x, \Lambda(e), \Phi(x), \mathsf{Val}(x) = e \vdash \Diamond(e, \Phi)}\,(\mathtt{id})(\Diamond\mathtt{i}) \qquad \dfrac{(\mathtt{hyp})}{\Gamma, x, e, \Lambda(e), \Diamond(e, \Phi) \vdash \Psi(e)}}{\Gamma, e, x, \Lambda(e), \Phi(x), \mathsf{Val}(x) = e \vdash \Psi(e)}}{\Gamma, x, \Lambda(\mathsf{Val}(x)), \Phi(x) \vdash \Psi(\mathsf{Val}(x))}$$

(*Case* ($\Diamond$ad) *implies* ($\Diamond$i)($\Diamond$e)):

$$\dfrac{\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, x, \Lambda, E = \mathsf{Val}(x), \Phi(x) \vdash \Psi}}{\Gamma, e, \Lambda, E = e, \Diamond(e, \Phi) \vdash \Psi}\,(\Diamond\mathtt{ad})}{\Gamma, \Lambda, \Diamond(E, \Phi) \vdash \Psi} \qquad \dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash \Diamond(E, \Phi)}}{\Gamma, \Lambda \vdash \Psi}$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma, e, \Lambda, \Diamond(e,\Phi) \vdash \Diamond(e,\Phi)}}{\Gamma, x, \Lambda, \Phi(x) \vdash \Diamond(\mathsf{Val}(x),\Phi)}(\Diamond\mathtt{ad}) \qquad \dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash \Phi(M)}}{\Gamma, x, \Lambda, x = M \vdash \Phi(x)}(\mathtt{ts})}{\dfrac{\Gamma, x, \Lambda, x = M \vdash \Diamond(\mathsf{Val}(x),\Phi)}{\Gamma, \Lambda \vdash \Diamond(\mathsf{Val}(M),\Phi)}(\mathtt{ts})}$$

and the result follows using $\Gamma, \Lambda \vdash E = \mathsf{Val}(M)$.

(*Case* (+i)(+e) *imply* (+ad)):

$$\dfrac{\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, z, x, \Lambda(\mathsf{Inl}(x)), \Phi(x) \vdash \Theta(F(x))}}{\Gamma, z, x, \Lambda(z), z = \mathsf{Inl}(x), \Phi(x) \vdash \Theta(F(x))}}{\left\{\begin{array}{l}\Gamma, z, x, \Lambda(z),\\ (\Phi+\Psi)(z), z = \mathsf{Inl}(x), \Phi(x) \vdash \Theta(F(x))\\ \qquad\text{and similarly for } G\end{array}\right. \qquad \dfrac{}{\left\{\begin{array}{l}\Gamma, z, \Lambda(z),\\ (\Phi+\Psi)(z) \vdash (\Phi+\Psi)(z)\end{array}\right.}}{\Gamma, z, \Lambda(z), (\Phi+\Psi)(z) \vdash \Theta(\{F,G\}(z))}$$

where the final rule is (+e).

$$\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, z, \Lambda(z), (\Phi+\Psi)(z) \vdash \Theta(\{F,G\}(z))} \qquad \dfrac{\dfrac{}{\Gamma, x, \Lambda(\mathsf{Inl}(x)), \Phi(x) \vdash \Phi(x)}}{\left\{\begin{array}{l}\Gamma, x, \Lambda(\mathsf{Inl}(x)),\\ \Phi(x) \vdash (\Phi+\Psi)(\mathsf{Inl}(x))\end{array}\right.}(+\mathtt{i})}{\left\{\begin{array}{l}\Gamma, x,\\ \Lambda(\mathsf{Inl}(x)), (\Phi+\Psi)(\mathsf{Inl}(x)) \vdash \Theta(F(x))\end{array}\right.}$$
$$\Gamma, x, \Lambda(\mathsf{Inl}(x)), \Phi(x) \vdash \Theta(F(x))$$

(*Case* (+ad) *implies* (+i)(+e)):

$$\dfrac{\dfrac{\dfrac{}{\Gamma, z, \Lambda, (\Phi+\Psi)(z) \vdash (\Phi+\Psi)(z)}}{\Gamma, x, \Lambda, \Phi(x) \vdash (\Phi+\Psi)(\mathsf{Inl}(x))}(+\mathtt{ad}) \qquad \dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash \Phi(M)}}{\Gamma, \Lambda \vdash (\Phi+\Psi)(\mathsf{Inl}(M))}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{(\mathtt{hyp})}{\Gamma, x, \Lambda, \mathsf{Inl}(x) = C, \Phi(x) \vdash \Theta(F(x))}}{\Gamma, z, \Lambda, z = C, (\Phi+\Psi)(z) \vdash \Theta(\{F,G\}(z))}(+\mathtt{ad})}{\Gamma, \Lambda, (\Phi+\Psi)(C) \vdash \Theta(\{F,G\}(C))} \qquad \dfrac{(\mathtt{hyp})}{\Gamma, \Lambda \vdash (\Phi+\Psi)(C)}}{\Gamma, \Lambda \vdash \Theta(\{F,G\}(C))}$$

□

4.4 Extensions of FIX

Some Inconsistent Extensions of the FIX Logic

The induction rule for *nat* is just the usual principle of mathematical induction. The induction rule for *fix* can be rendered informally as: to prove that a property $\Phi(n)$ holds of all elements n in *fix*, it is sufficient to prove for all computations e of an element of *fix* that $\Phi(\sigma(e))$ holds if whenever e evaluates to a value, that value satisfies $\Phi(x)$. This principle is consistent (see Section 5.1) but only because the FIX propositions have limited forms. In fact, extending the FIX logic with unrestricted intuitionistic negation, implication or existential quantification renders it inconsistent.

Proposition 4.4.1 Extending the FIX logic with intuitionistic implication renders the system inconsistent.

Proof Since FIX contains falsity (false), adding implication ($\Phi \supset \Psi$) means that one also has negation ($\neg\Phi \equiv (\Phi \supset \mathsf{false})$). So consider the proposition

$$\Phi(n) \stackrel{\text{def}}{=} \neg(\sigma(\omega) = n)$$

about $n{:}\mathit{fix}$. From (ts) we can deduce that $\Gamma, e{:}T\mathit{fix}, \Lambda, \Box(e,\Phi), \omega = e \vdash \Box(\omega,\Phi)$ and using Proposition 4.5.1 we see that $\Gamma, \Lambda, \Box(\mathsf{Val}(\sigma(\omega)),\Phi) \vdash \Phi(\sigma(\omega))$. Recalling that ω is provably equal to $\mathsf{Val}(\sigma(\omega))$ together with the rules for intuitionistic implication it is easy to see that

$$\Gamma, e{:}T\mathit{fix}, \Lambda, \Box(e,\Phi), \omega = e, \sigma(\omega) = \sigma(\omega) \vdash \mathsf{false}.$$

But the structure map of any initial T algebra is an isomorphism, which means that in FIX we have $\Gamma, e, \Lambda, \sigma(\omega) = \sigma(e) \vdash \omega = e$. Hence

$$\Gamma, e{:}T\mathit{fix}, \Lambda, \Box(e,\Phi), \sigma(\omega) = \sigma(e) \vdash \mathsf{false}$$

and applying (fixin) we have $\Gamma, n{:}\mathit{fix}, \Lambda \vdash \Phi(n)$.
So the induction principle for *fix* entails that $\Phi(n)$ holds of all $n \in \mathit{fix}$, and in particular of $\sigma(\omega)$, which is a contradiction. □

Proposition 4.4.2 Extending the FIX logic with intuitionistic existential quantification renders the system inconsistent.

Proof This proof mimics the ideas which show that the category $\omega\mathcal{C}po$ together with inclusive subsets does not model standard intuitionistic predicate calculus [Dum77]. Recall that in $\omega\mathcal{C}po$, Beck Chevalley conditions fail for left adjoints to projections; for if this is not the case we can deduce that such left adjoints take inclusive subsets to inclusive subsets by unravelling Beck Chevalley at a global element in $\omega\mathcal{C}po$. Then considering the ωcpo $\mathbb{N} \times \Omega$ and inclusive subset

$$\{(m,n) \mid m \in \mathbb{N} \;\&\; n \in \Omega \setminus \{\top\} \;\&\; n \leq m\}$$

we can deduce that $\{n \mid n \in \Omega \setminus \{\top\}\}$ is inclusive in Ω. This is not so.

Consider the term $\bot \stackrel{\text{def}}{=} \mathsf{It}_{Tfix}(e.\mathsf{Let}\,(e, x.x), \sigma(\omega)) : Tfix$ (recall the discussion of domain theoretic properties of FIX on Page 48). Set

$$\Phi(n) \stackrel{\text{def}}{=} \exists_{nat}(m.(u.\sigma(\mathsf{Val}(u)))^m(\sigma(\bot))) =_{fix} n.$$

Using the usual rules for intuitionistic existential quantification together with the FIX rules we may deduce $e, n, e = \mathsf{Val}(n), \Box(e, \Phi) \vdash \Phi(n) \vdash \Phi(\sigma(\mathsf{Val}(n)))$ and from (`fixin`) we have $n : fix \vdash \Phi(n)$. In particular this means that

$$\vdash \exists_{nat}(m.(u.\sigma(\mathsf{Val}(u)))^m(\sigma(\bot))) =_{fix} \sigma(\omega).$$

Using (`mono`) and that σ is an isomorphism we conclude $\vdash \bot = \omega$. □

4.5 Further Results about the FIX Logic

The Modality Conditions

We state and prove a proposition which we shall make use of in Chapter 5 where it will be used in establishing a categorical logic correspondence for the FIX logic.

Proposition 4.5.1 Within the FIX logical system, the following birules are derivable:

$$\frac{\Gamma, \Lambda \vdash \Phi(M)}{\Gamma, \Lambda \vdash \Box(\mathsf{Val}(M), \Phi)} \qquad \frac{\Gamma, \Lambda \vdash \Phi(M)}{\Gamma, \Lambda \vdash \Diamond(\mathsf{Val}(M), \Phi)}$$

$$\frac{\Gamma, \Lambda \vdash \Phi(M) \quad \Gamma,\, y{:}\beta,\, \Lambda \vdash \Psi(y)\ \mathrm{prop}}{\Gamma, \Lambda \vdash (\Phi + \Psi)(\mathsf{Inl}_\alpha(M))} \qquad \frac{\Gamma, \Lambda \vdash \Psi(N) \quad \Gamma,\, x{:}\alpha,\, \Lambda \vdash \Phi(x)\ \mathrm{prop}}{\Gamma, \Lambda \vdash (\Phi + \Psi)(\mathsf{Inr}_\beta(N))}$$

Proof The forward directions of each of these birules are easy to see and thus we omit the details. We shall give details of the backwards directions for the existential modality and coproduct.

First, the existential modality. We have

$$\cfrac{\cfrac{\cfrac{(\mathtt{mono})(\mathtt{cut})}{\Gamma, x, y, \Lambda, \mathsf{Val}(x) = \mathsf{Val}(y), \Phi(x) \vdash \Phi(y)}}{\Gamma, e, y, \Lambda, e = \mathsf{Val}(y), \Diamond(e, \Phi) \vdash \Phi(y)}\,(\Diamond\mathtt{ad})}{\Gamma, y, \Lambda, \Diamond(\mathsf{Val}(y), \Phi) \vdash \Phi(y)}\,(\mathtt{sub})$$

from which the result is immediate. Now we move to the coproduct.

We have

$$\dfrac{\left\{\begin{array}{l}\dfrac{}{\Gamma,\Lambda\vdash(\Phi+\Psi)(\mathsf{Inl}(M))}\,(\mathtt{hyp})\\[2ex]\dfrac{}{\Gamma,x,\Lambda,\mathsf{Inl}(x)=\mathsf{Inl}(M),\Phi(x)\vdash\Phi((u.u)(x))}\,(\mathtt{id})\\[2ex]\dfrac{}{\Gamma,y,\Lambda,\mathsf{Inr}(y)=\mathsf{Inl}(M),\Psi(y)\vdash\Phi((v.M)(y))}\,(\mathtt{djsum})(\bot)\end{array}\right.}{\Gamma,\Lambda\vdash\Phi(\{u.u,v.M\}(\mathsf{Inl}(M)))}\,(+\mathtt{e})$$

and the result follows from the rules for Coproduct Equations. □

Frobenius Reciprocity for the FIX Logic

To prove a Frobenius Reciprocity style rule for the FIX logic, it is convenient to use the next lemma:

Lemma 4.5.2 In the FIX logic, if $\Gamma,\Lambda,\Phi\vdash\Psi$, then

$$\Gamma,e,\Lambda,\Diamond(e,y.\Phi)\vdash\Diamond(e,y.\Psi).$$

Proof Use ($\Diamond$ad) and (sub). □

Proposition 4.5.3 Within the fix logical system, the following birule is derivable:

$$\dfrac{\Gamma,e,x,\Lambda\vdash\Diamond(e,y.\Phi(x,\mathsf{Val}(y))\ \&\ \Psi(x,y))}{\Gamma,e,x,\Lambda\vdash\Phi(x,e)\ \&\ \Diamond(e,y.\Psi(x,y))}\,(\mathtt{fr})$$

Proof First we prove the forwards direction. From the hypothesis, Lemma 4.5.2, and the rules (&ad) for conjunction, we deduce[1]

$$\Diamond(e,y.\Phi(x,\mathsf{Val}(y))\ \&\ \Psi(x,y))\vdash\Diamond(e,y.\Phi(x,\mathsf{Val}(y)))\ \&\ \Diamond(e,y.\Psi(x,y)).$$

Also, we have

$$\dfrac{\left\{\begin{array}{l}e=\mathsf{Val}(y),\\ \Phi(x,\mathsf{Val}(y))\vdash\Phi(x,e)\end{array}\right.\quad\dfrac{}{\Diamond(e,y.\Phi(x,\mathsf{Val}(y)))\vdash\Diamond(e,y.\Phi(x,\mathsf{Val}(y)))}\,(\mathtt{id})}{\Diamond(e,y.\Phi(x,\mathsf{Val}(y)))\vdash\Phi(x,e)}\,(\mathtt{id})(=\mathtt{ad})(\mathtt{sub})\ (\Diamond\mathtt{e})$$

and the result is immediate from the rules (&ad).
For the backwards direction, note that using ($\Diamond$i) we have

$$\Phi(x,e),\Psi(x,y),\mathsf{Val}(y)=e\vdash\Diamond(e,y.\Phi(x,\mathsf{Val}(y))\ \&\ \Psi(x,y));$$

[1]For sake of space we omit contexts in this proof.

label this sequent $(*)$. Hence

$$\frac{\overline{\Diamond(e, y.\Psi(x,y)), (*)} \qquad \overline{\Phi(x,e), \Diamond(e, y.\Psi(x,y)) \vdash \Diamond(e, y.\Psi(x,y))}}{\Phi(x,e), \Diamond(e, y.\Psi(x,y)) \vdash \Diamond(e, y.\Phi(x, \mathsf{Val}(y)) \;\&\; \Psi(x,y))}\,(\Diamond\mathsf{e})$$

and from (&ad) and (cut) we are done. □

The Existence and Disjunction Properties

We now give two results which witness the constructive nature of the FIX logic. They bear some resemblance to the existence and disjunction properties of standard intuitionistic logic [LS80], [Pit89].

Theorem 4.5.4 ["Existence Property"] If E is a closed term of type $T\alpha$, then $\vdash \Diamond(E, \Phi)$ is derivable in FIX if and only if there is a closed term M of type α for which $\vdash E =_{T\alpha} \mathsf{Val}(M)$ and $\vdash \Phi(M)$ are derivable. (In other words, a formal proof that E evaluates to a value satisfying $\Phi(x)$ necessitates the existence of a term denoting that value.)

Theorem 4.5.5 ["Disjunction Property"] If E is a closed term of coproduct type $\alpha + \beta$, $\Phi(x)$ and $\Psi(y)$ are properties of α and β and $\vdash (\Phi + \Psi)(E)$ is derivable in FIX, then either $\vdash E =_{\alpha+\beta} \mathsf{Inl}(M)$ and $\vdash \Phi(M)$ are derivable for some closed term M of type α, or $\vdash E =_{\alpha+\beta} \mathsf{Inr}(N)$ and $\vdash \Psi(N)$ are derivable for some closed term N of type β.

Standardness of the Natural Number Type

The Existence Property enables one to produce closed terms of type *nat* from a computation of a number (i.e. a closed term of type $T\mathit{nat}$) together with a proof that the computation converges. There remains the possibility that a closed term of type *nat* is not a value, i.e. a standard numeral. However, this is not so:

Theorem 4.5.6 ["Standardness of *nat*"] Every closed term of type *nat* in FIX is provably equal to a standard numeral $\mathsf{Suc}^n(\mathsf{O})$.

Theorems 4.5.4, 4.5.5 and 4.5.6 will be proved in Chapter 5.

Miscellaneous Results

Lemma 4.5.7 Modulo the remaining rules of the FIX logic, the modality condition

$$\frac{\Gamma,\ \Lambda \vdash \mathsf{Let}\,(E, F) =_{T\alpha} \mathsf{Val}(M)}{\Gamma, \Lambda \vdash \Diamond(E, x.F(x) =_{T\alpha} \mathsf{Val}(M))}\,(\mathtt{mod})$$

is equivalent to the rule

$$\frac{\Gamma, \Lambda \vdash \Box(E, x.\Box(F(x), \Phi))}{\Gamma, \Lambda \vdash \Box(\mathsf{Let}\,(E, F), \Phi)}\,(\mathtt{mod}')$$

Proof (*Case* (mod) *implies* (mod′)): Applying ($\Box$e) twice to the hypothesis of (mod′) we have

$$\Gamma, y, x, \Lambda, E = \mathsf{Val}(x), F(x) = \mathsf{Val}(y) \vdash \Phi(y)$$

and using (mod) we obtain

$$\Gamma, y, \Lambda, \mathsf{Let}\,(E, F) = \mathsf{Val}(y) \vdash \Diamond(E, x.F(x) = \mathsf{Val}(y)).$$

By a suitable weakening of the hypotheses of the above derived judgements, we can apply ($\Diamond$e) and ($\Box$i) to obtain the result.

(*Case* (mod′)*implies* (mod)): As this direction is not entirely straightforward we give a full proof tree:

$$\dfrac{\dfrac{\left\{\dfrac{\dfrac{}{\Gamma, x, y, \Lambda, E = \mathsf{Val}(x), F(x) = \mathsf{Val}(y) \vdash E = \mathsf{Val}(x)}(\mathtt{id})}{\Gamma, x, y, \Lambda, E = \mathsf{Val}(x), F(x) = \mathsf{Val}(y) \vdash F(x) = \mathsf{Val}(y)}(\mathtt{id})\right.}{\Gamma, x, y, \Lambda, E = \mathsf{Val}(x), F(x) = \mathsf{Val}(y) \vdash \Diamond(E, u.F(u) = \mathsf{Val}(y))}(\Diamond\mathtt{i})}{\Gamma, \Lambda \vdash \Box(E, x.\Box(F(x), y.\Diamond(E, u.F(u) = \mathsf{Val}(y))))}(\Box\mathtt{i})}{\Gamma, \Lambda \vdash \Box(\mathsf{Let}\,(E, F), y.\Diamond(E, u.F(u) = \mathsf{Val}(y)))}(\mathtt{mod}')$$

Using this conclusion together with the premiss of (mod) as the premisses for ($\Box$e) completes the proof. □

Lemma 4.5.8 Modulo the remaining rules of the FIX logic, the rule

$$\frac{\Gamma, \Lambda \vdash \mathsf{Val}(M) = \mathsf{Val}(M')}{\Gamma, \Lambda \vdash M = M'}\,(\mathtt{mono})$$

is equivalent to the rule

$$\frac{\Gamma, \Lambda \vdash \Phi(M)}{\Gamma, \Lambda \vdash \Box(\mathsf{Val}(M), \Phi)}\,(\mathtt{mono}')$$

Proof We sketch the details.

(*Case* (mono) *implies* (mono′)): The rule (mono) yields

$$\Gamma, x, \Lambda, \mathsf{Val}(M) = \mathsf{Val}(x) \vdash M = x.$$

Applying (ts) to the hypothesis of (mono′), cutting (cut) and using ($\Box$i) we are done.

(*Case* (mono′) *implies* (mono)): The rule (mono′) yields

$$\Gamma, \Lambda \vdash \Box(\mathsf{Val}(M), x.x = M).$$

Now apply ($\Box e$) to this along with the hypothesis of (mono). □

Chapter 5
Categorical Semantics of the FIX Logic

5.1 FIX Hyperdoctrines

For background on hyperdoctrines and indexed categories see [JP78], [See83] and [Pit89].

Definition 5.1.1 A *FIX hyperdoctrine* is specified by a FIX category $\mathcal{C}$ (referred to as the base category) together with a $\mathcal{C}$ indexed poset, $\mathcal{C}:\mathcal{C}^{op} \rightarrow \mathcal{P}oset$, where if $f:A \rightarrow B$ is a morphism in the base category $\mathcal{C}$ we denote the corresponding pullback function by $f^*:\mathcal{C}(B) \rightarrow \mathcal{C}(A)$ with the fibre at A denoted by $\mathcal{C}(A)$. We adopt the following notational convention. If

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a commuting square in $\mathcal{C}$ then *right Beck-Chevalley conditions* are said to hold (which will be abbreviated to RBC) if $f^*:\mathcal{C}(B) \rightarrow \mathcal{C}(A)$ and $k^*:\mathcal{C}(D) \rightarrow \mathcal{C}(C)$ have right adjoints which satisfy the identity $\forall f \circ g^* = h^* \circ \forall k$. We use a dual convention for *left Beck-Chevalley conditions*, LBC. The indexed poset satisfies the following conditions:

1. The fibres are meet semi-lattices with least element, and the fibre over the initial object is a singleton. The top element is denoted by $\top$, the bottom element by $\bot$, and the meet of elements $x \in \mathcal{C}(A)$ and $y \in \mathcal{C}(A)$ by $x \wedge y \in \mathcal{C}(A)$. The pullback functions preserve meets, top and bottom elements.

2. RBC holds for all squares of the form

$$\begin{array}{ccc} C \times A & \xrightarrow{\pi} & C \\ {\scriptstyle f \times id}\downarrow & & \downarrow{\scriptstyle f} \\ C' \times A & \xrightarrow[\pi']{} & C' \end{array}$$

where the morphisms π and π' are product projections.

3. RBC and LBC hold for all squares of the form

$$\begin{array}{ccc} C \times A & \xrightarrow{id \times \eta} & C \times TA \\ {\scriptstyle f \times id}\downarrow & & \downarrow{\scriptstyle f \times id} \\ C' \times A & \xrightarrow[id \times \eta]{} & C' \times TA \end{array}$$

Also, the hyperdoctrine enjoys a form of Frobenius Reciprocity, namely given $x \in \mathcal{C}(C \times TA)$ and $y \in \mathcal{C}(C \times A)$ we have

$$\exists(id \times \eta)((id \times \eta)^*(x) \wedge y) = x \wedge \exists(id \times \eta)(y).$$

These conditions ensure the soundness of the rules for Universal and Existential Modality Entailment.

4. There is an operation $+$ on fibres

$$+ : \mathcal{C}(C \times A) \times \mathcal{C}(C \times B) \longrightarrow \mathcal{C}(C \times (A + B))$$

which is natural in C. Suppose we are given elements

$$\begin{array}{ll} x \in \mathcal{C}(C \times A) & u \in \mathcal{C}(C \times (A + B)) \\ y \in \mathcal{C}(C \times B) & z \in \mathcal{C}(C \times D) \end{array}$$

and morphisms $f : C \times A \to D$, $g : C \times B \to D$. Then we demand that

$$\frac{(id_C \times i)^*(u) \wedge x \leq \langle \pi_A, f \rangle^*(z) \quad (id_C \times j)^*(u) \wedge y \leq \langle \pi_B, g \rangle^*(z)}{u \wedge (x + y) \leq \langle \pi, \{f, g\} \rangle^*(z)}$$

where $i : A \to (A + B)$ and $j : B \to (A + B)$ are coproduct insertions,

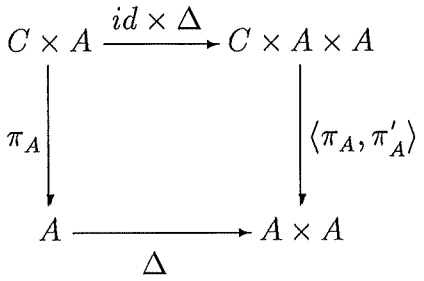
$$(C \times A) + (C \times B) \xrightarrow{\phi} C \times (A + B)$$

is the obvious isomorphism and

$$\pi_A : C \times A \to C \quad \pi_B : C \times B \to C \quad \pi : C \times (A + B) \to C$$

are product projections. Finally, $\{f, g\} \stackrel{\text{def}}{=} [f, g] \circ \phi^{-1}$ where $[f, g]$ arises from the coproduct structure of $\mathcal{C}$. Note that if $x+y$ exists, it is determined uniquely. These requirements ensure the soundness of the rules for Coproduct Entailment.

5. LBC holds for



The left adjoint to $id \times \Delta$ satisfies the following Frobenius Reciprocity condition,

$$\exists(id \times \Delta) \circ (id \times \Delta)^*(x) = x \wedge \exists(id \times \Delta) \circ \pi_C^*(\top)$$

where $x \in \mathcal{C}(C \times A \times A)$ and $\pi_C : C \times A \to C$. (Recall that the pullback function π_C^* preserves the top element by definition.) These conditions ensure the soundness of the rules for Equality Entailment.

6. We demand the inequalities

$$(\eta \times \eta)^* \circ \exists\Delta(\top_{TA}) \leq \exists\Delta(\top_A)$$

and

$$\langle i, j \rangle^* \circ \exists\Delta(\top_{A+B}) = \bot$$

where $\top_A \in \mathcal{C}(A)$, $\top_{TA} \in \mathcal{C}(TA)$ and $\top_{A+B} \in \mathcal{C}(A+B)$ are the top elements of the fibres and $i : A \to (A+B)$, $j : B \to (A+B)$ are coproduct insertions. This guarantees the soundness of the Mono Condition and the Disjoint Sum Condition.

7. Given a morphism $f : C \times TA \times B \times A \to TB$ then we demand the inequality

$$\langle \mathit{lift}(f), \eta\pi_1 \rangle^* \circ \exists\Delta(\top) \leq \exists(id \times \eta) \circ \langle f, \eta\pi_2 \rangle^* \circ \exists\Delta(\top)$$

where $\top \in \mathcal{C}(TA)$ is the top element of the fibre and

$$\pi_1 : C \times TA \times B \times TA \to B \quad \pi_2 : C \times TA \times B \times A \to B$$

are product projections. This ensures the soundness of the Modality Condition.

Finally, to complete the definition of a FIX hyperdoctrine, there are two fibrewise induction conditions and a coherence condition. The induction conditions ensure soundness of the induction rules in the logic and the coherence condition guarantees that semantic equality of terms coincides with derivable equality in the FIX logic.

8. Given elements $x \in \mathcal{C}(C)$, $y \in \mathcal{C}(C \times N)$ then we demand that

$$\frac{x \leq \langle id, 0 \circ ! \rangle^*(y) \quad \pi^*(x) \wedge y \leq (id \times s)^*(y)}{\pi^*(x) \leq y}$$

where $\pi : C \times N \to C$ is a product projection and $0 : 1 \to N$, $s : N \to N$ are given as part of the structure of the NNO in the base category.

9. Given elements $x \in \mathcal{C}(C)$, $y \in \mathcal{C}(C \times \Omega)$ then we demand that

$$\frac{\pi^*(x) \wedge \forall(id \times \eta)(y) \leq (id \times \sigma)^*(y)}{\pi'^*(x) \leq y}$$

where $\pi: C \times T\Omega \to C$, $\pi': C \times \Omega \to C$ are product projections and $\sigma: T\Omega \to \Omega$ is given as part of the structure of the FPO in the base category.

10. Given morphisms $f, g: B \to A$ and the diagonal $\Delta: A \to A \times A$, then we ask that

$$\frac{\langle f, g\rangle^* \circ \exists\Delta(\top) = \top}{f = g \text{ in } \mathcal{C}}$$

This completes Definition 5.1.1. A *morphism* of FIX hyperdoctrines $\mathcal{C}$ and $\mathcal{C}'$ is specified by a FIX category morphism between the base categories (referred to as the base functor) say $F: \mathcal{C} \to \mathcal{C}'$, together with an indexed collection of monotone functions, called fibre morphisms, $F_A: \mathcal{C}(A) \to \mathcal{C}'(FA)$ for each object A in $\mathcal{C}$. These monotone functions are required to preserve the structure of the fibres in a canonical fashion. For example, the pullback functions are preserved by the fibre morphisms in the sense that given a morphism $f: A \to B$ in $\mathcal{C}$, the following square commutes

$$\begin{array}{ccc} \mathcal{C}(A) & \xrightarrow{F_A} & \mathcal{C}'(FA) \\ {\scriptstyle f^*}\uparrow & & \uparrow{\scriptstyle (F(f))^*} \\ \mathcal{C}(B) & \xrightarrow[F_B]{} & \mathcal{C}'(FB) \end{array}$$

Also, the structure of the fibres is preserved by the fibre morphisms; for example

- Given $\top \in \mathcal{C}(A)$, then $F_A(\top) = \top \in \mathcal{C}'(FA)$,
- given $x, y \in \mathcal{C}(A)$, then $F_A(x \wedge y) = F_A(x) \wedge F_A(y)$, and
- given $x \in \mathcal{C}(C \times A)$ and $y \in \mathcal{C}(C \times B)$ then

$$F_{C \times (A+B)}(x + y) = F_{C \times A}(x) + F_{C \times B}(y).$$

The remaining structure of the fibres is preserved in a similar way; the details are omitted.

The FIX Hyperdoctrine $\omega\mathcal{C}po$

The definition of a FIX hyperdoctrine is quite involved and so we must give a concrete example. We have the

Proposition 5.1.2 Recall that the category $\omega\mathcal{C}po$ is a FIX category. There is an $\omega\mathcal{C}po$ indexed poset, $\mathcal{I}: \omega\mathcal{C}po^{op} \to \mathcal{P}oset$, where $\mathcal{I}$ takes an ωcpo D to the set of inclusive subsets of D which are ordered by inclusion and $\mathcal{I}$ takes each continuous function $f: D \to D'$ to its inverse image function f^{-1} restricted to inclusive subsets. This gives rise to a FIX hyperdoctrine.

Proof It is trivial to check that $f^{-1}: \mathcal{I}(D') \to \mathcal{I}(D)$ is well defined and indeed monotone and that $\mathcal{I}$ is a functor. We define the operations that make $\omega\mathcal{C}po$ a FIX hyperdoctrine, but omit detailed verifications.

1. With meet given by intersection of inclusive subsets, it is clear that each fibre is a meet semi-lattice with least element. It is easy to see that each pullback function is a morphism of pointed meet semi-lattices. Finally $\mathcal{I}(\emptyset) = \{\emptyset\}$ is a singleton.

2. The right adjoint to projection is given by restriction of the dual image functions to inclusive subsets; that RBC holds is trivial.

3. The existence of left adjoints is well known, given by restriction of the set theoretic direct image functions to inclusive subsets. The right adjoint to
$$(id \times \iota)^{-1}: \mathcal{I}(C \times D_\perp) \to \mathcal{I}(C \times D)$$
is given by
$$\forall(id \times \iota)(I) \stackrel{\text{def}}{=} (id \times \iota)(I) \cup \{(c, \perp) \mid c \in C\}$$
where $I \in \mathcal{I}(C \times D)$. It is easy to see that this is a good definition and yields the required adjoint. Checking RBC and LBC is easy; Frobenius Reciprocity is virtually immediate.

4. Let $i: D \to D + D'$ and $j: D' \to D + D'$ be coproduct insertions. Given elements $I \in \mathcal{I}(C \times D)$ and $J \in \mathcal{I}(C \times D')$ we define
$$+ : \mathcal{I}(C \times D) \times \mathcal{I}(C \times D') \to \mathcal{I}(C \times (D + D'))$$
by
$$I + J \stackrel{\text{def}}{=} \exists(id \times i)(I) \cup \exists(id \times j)(J).$$

Note that the fibrewise induction conditions are satisfied because any inclusive subset of an ωcpo is an ωcpo. Finally note that the existence of this concrete model implies the consistency of the pure FIX logic relative to Zermelo Fraenkel set theory. □

5.2 Categorical Semantics of FIX

Structures for FIX Signatures

Let $\mathcal{C}$ be a FIX hyperdoctrine and Sg a FIX signature. Then a *structure*, $\mathbf{M}$, in $\mathcal{C}$ is specified by the following data:

- A structure in the base FIX category $\mathcal{C}$ (see Section 3.2), and
- for each basic relation symbol $R: \alpha_1, \ldots, \alpha_n$ an element $[\![R]\!]$ in the fibre over $[\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!]$.

Interpretation of the FIX Propositions in Context

Given a structure $\mathbf{M}$ we now show how to interpret the FIX logic in a FIX hyperdoctrine. The propositions of FIX are modelled by elements in the fibres of the hyperdoctrine; more precisely, for each context Γ and proposition Φ for which we can derive $\Gamma \vdash \Phi$ prop, we specify an element $[\![\Gamma.\Phi]\!]$ of the fibre $\mathcal{C}([\![\Gamma]\!])$. If Γ is empty we write $[\![\Phi]\!]$ for this. The semantics of propositions in context is defined using the structure of the propositions:

- $[\![\Gamma.\mathsf{true}]\!] \stackrel{\text{def}}{=} \top \in \mathcal{C}([\![\Gamma]\!])$.
- $[\![\Gamma.\mathsf{false}]\!] \stackrel{\text{def}}{=} \bot \in \mathcal{C}([\![\Gamma]\!])$.
- $[\![\Gamma.M =_\alpha N]\!] \stackrel{\text{def}}{=} \langle [\![\Gamma.M]\!], [\![\Gamma.N]\!] \rangle^* \circ \exists\Delta(\top)$.
- $[\![\Gamma.\Phi \,\&\, \Psi]\!] \stackrel{\text{def}}{=} [\![\Gamma.\Phi]\!] \wedge [\![\Gamma.\Psi]\!]$.
- $[\![\Gamma.\forall_\alpha(\Phi)]\!] \stackrel{\text{def}}{=} \forall\pi_1([\![\Gamma, x{:}\,\alpha.\Phi]\!])$.
- $[\![\Gamma.\Box(E,\Phi)]\!] \stackrel{\text{def}}{=} \langle id, [\![\Gamma.E]\!] \rangle^* \circ \forall(id \times \eta)([\![\Gamma, x{:}\,\alpha.\Phi(x)]\!])$.
- $[\![\Gamma.\Diamond(E,\Phi)]\!] \stackrel{\text{def}}{=} \langle id, [\![\Gamma.E]\!] \rangle^* \circ \exists(id \times \eta)([\![\Gamma, x{:}\,\alpha.\Phi(x)]\!])$.
- $[\![\Gamma.(\Phi + \Psi)(C)]\!] \stackrel{\text{def}}{=} \langle id, [\![\Gamma.C]\!] \rangle^*([\![\Gamma, x{:}\,\alpha.\Phi(x)]\!] + [\![\Gamma, y{:}\,\beta.\Psi(y)]\!])$.

Models of FIX Theories

If Λ is a finite set of propositions, each of which is well formed in the context Γ, then set

$$[\![\Gamma.\Lambda]\!] \stackrel{\text{def}}{=} \bigwedge_{\Theta \in \Lambda} [\![\Gamma.\Theta]\!].$$

A structure $\mathbf{M}$ in a FIX hyperdoctrine $\mathcal{C}$ *satisfies* a sequent in context $\Gamma, \Lambda \vdash \Phi$ if $[\![\Gamma.\Lambda]\!] \leq [\![\Gamma.\Phi]\!]$ holds in the fibre $\mathcal{C}([\![\Gamma]\!])$. Given a FIX theory, Th, then $\mathbf{M}$ is called a *model* of the theory if it satisfies all the axioms of Th.

The Substitution Lemma

The next lemma tells us how substitution of terms for variables in propositions is modelled.

Lemma 5.2.1 Put $\Gamma' = [x_1{:}\,\alpha_1, \ldots, x_n{:}\,\alpha_n]$, let $\Gamma' \vdash \Phi(x_1, \ldots, x_n)$ prop be a FIX proposition in context and let $\Gamma \vdash M_i{:}\,\alpha_i$ for $i = 1, \ldots, n$ be FIX terms in context. Then

$$[\![\Gamma.\Phi(\vec{M})]\!] = \langle [\![\Gamma.M_1]\!], \ldots, [\![\Gamma.M_n]\!] \rangle^*([\![\Gamma'.\Phi(\vec{x})]\!]).$$

Proof The proof proceeds by induction on the structure of propositions; we illustrate with $\Box(E, \Phi)$ and $(\Phi + \Psi)(C)$.

$$\begin{aligned}
& [\![\Gamma.\vec{u}.\Box(E(\vec{u}), \Phi(\vec{u}))(\vec{M})]\!] \\
= \; & [\![\Gamma.\Box(E(\vec{M}), \Phi(\vec{M}))]\!] \\
= \; & \langle id, [\![\Gamma.E(\vec{M})]\!] \rangle^* [\![\Gamma, e.\Box(e, \Phi(\vec{M}))]\!] \\
= \; & \langle id, [\![\Gamma.E(\vec{M})]\!] \rangle^* \circ \forall(id \times \eta)([\![\Gamma, x.\Phi(\vec{M}, x)]\!]) \\
\text{induction} \; = \; & \langle id, [\![\Gamma.E(\vec{M})]\!] \rangle^* \circ \forall(id \times \eta) \circ (\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^*([\![\Gamma', x.\Phi(\vec{x})(x)]\!]) \\
= \; & \langle id, [\![\Gamma.E(\vec{M})]\!] \rangle^* \circ (\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^* \circ \forall(id \times \eta)([\![\Gamma', x.\Phi(\vec{x})(x)]\!]) \\
= \; & \langle id, [\![\Gamma'.E(\vec{x})]\!] \langle [\![\Gamma.\vec{M}]\!] \rangle \rangle^* \circ (\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^*([\![\Gamma', e'.\Box(e', \Phi(\vec{x}))]\!]) \\
= \; & ((\langle [\![\Gamma.\vec{M}]\!] \rangle \times id) \circ \langle id, [\![\Gamma'.E(\vec{x})]\!] \langle [\![\Gamma.\vec{M}]\!] \rangle \rangle)^*([\![\Gamma', e'.\Box(e', \Phi(\vec{x}))]\!]) \\
= \; & \langle \langle [\![\Gamma.\vec{M}]\!] \rangle, [\![\Gamma'.E(\vec{x})]\!] \langle [\![\Gamma.\vec{M}]\!] \rangle \rangle^*([\![\Gamma', e'.\Box(e', \Phi(\vec{x}))]\!]) \\
= \; & (\langle id, [\![\Gamma'.E(\vec{x})]\!] \rangle \circ \langle [\![\Gamma.\vec{M}]\!] \rangle)^*([\![\Gamma', e'.\Box(e', \Phi(\vec{x}))]\!]) \\
= \; & \langle [\![\Gamma.\vec{M}]\!] \rangle^* \langle id, [\![\Gamma'.E(\vec{x})]\!] \rangle^*([\![\Gamma', e'.\Box(e', \Phi(\vec{x}))]\!]) \\
= \; & \langle [\![\Gamma.\vec{M}]\!] \rangle^*([\![\Gamma'.\Box(E(\vec{x}), \Phi(\vec{x}))]\!]).
\end{aligned}$$

$$\begin{aligned}
& [\![\Gamma.\vec{u}.(\Phi(\vec{u}) + \Psi(\vec{u}))(C(\vec{u}))(\vec{M})]\!] \\
= \; & [\![\Gamma.(\Phi(\vec{M}) + \Psi(\vec{M}))(C(\vec{M}))]\!] \\
= \; & \langle id, [\![\Gamma.C(\vec{M})]\!] \rangle^* [\![\Gamma, z'.(\Phi(\vec{M}) + \Psi(\vec{M}))(z')]\!] \\
= \; & \langle id, [\![\Gamma.C(\vec{M})]\!] \rangle^* ([\![\Gamma, x.\Phi(\vec{M})(x)]\!] + [\![\Gamma, y.\Psi(\vec{M})(y)]\!]) \\
\text{induction} \; = \; & \langle id, [\![\Gamma.C(\vec{M})]\!] \rangle^* ((\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^* \\
& ([\![\Gamma', x.\Phi(\vec{x})(x)]\!]) + (\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^*([\![\Gamma', y.\Psi(\vec{x})(y)]\!])) \\
= \; & \langle id, \langle [\![\Gamma.\vec{M}]\!] \rangle \rangle^* \circ (id \times [\![\Gamma'.C(\vec{x})]\!])^*((\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^* \\
& ([\![\Gamma', x.\Phi(\vec{x})(x)]\!]) + (\langle [\![\Gamma.\vec{M}]\!] \rangle \times id)^*([\![\Gamma', y.\Psi(\vec{x})(y)]\!])) \\
= \; & \langle [\![\Gamma.\vec{M}]\!] \rangle^* \circ \langle id, [\![\Gamma'.C(\vec{x})]\!] \rangle^*([\![\Gamma', x.\Phi(\vec{x})(x)]\!] + [\![\Gamma', y.\Psi(\vec{x})(y)]\!]) \\
= \; & \langle [\![\Gamma.\vec{M}]\!] \rangle^*([\![\Gamma'.(\Phi(\vec{x}) + \Psi(\vec{x}))(C(\vec{x}))]\!]).
\end{aligned}$$

□

The Soundness Theorem

The categorical semantics of the FIX logic is sound; indeed we have:

Proposition 5.2.2 ["Soundness for FIX Theories"] Let $\mathcal{C}$ be a FIX hyperdoctrine, Th a FIX theory and $\mathbf{M}$ a model of Th in $\mathcal{C}$. Then $\mathbf{M}$ satisfies any sequent in context which is a theorem of Th.

Proof We need to check that the collection of sequents in context which are satisfied by $\mathbf{M}$ is closed under the rules for generating sequents in context. We give just one example of this, by checking the soundness of Existential Modality Entailment. We have

$$\bigwedge_{\Theta \in \Lambda} [\![\Gamma, x.\Theta(\mathsf{Val}(x))]\!] \wedge [\![\Gamma, x.\Phi(x)]\!] \leq [\![\Gamma, x.\Psi(\mathsf{Val}(x))]\!]$$

iff

$$\bigwedge_{\Theta \in \Lambda} (id \times \eta)^*[\![\Gamma, e.\Theta(e)]\!] \wedge [\![\Gamma, x.\Phi(x)]\!] \leq (id \times \eta)^*[\![\Gamma, e.\Psi(e)]\!].$$

Pullback functions preserve meet, so this is iff

$$(id \times \eta)^*[\![\Gamma, e.\Lambda(e)]\!] \wedge [\![\Gamma, x.\Phi(x)]\!] \leq (id \times \eta)^*[\![\Gamma, e.\Psi(e)]\!].$$

Using adjointness, this holds iff

$$\exists(id \times \eta)((id \times \eta)^*[\![\Gamma, e.\Lambda(e)]\!] \wedge [\![\Gamma, x.\Phi(x)]\!]) \leq [\![\Gamma, e.\Psi(e)]\!]$$

and from Frobenius Reciprocity iff

$$[\![\Gamma, e.\Lambda(e)]\!] \wedge \exists(id \times \eta)[\![\Gamma, x.\Phi(x)]\!] \leq [\![\Gamma, e.\Psi(e)]\!]$$

iff

$$[\![\Gamma, e.\Lambda(e)]\!] \wedge [\![\Gamma, e.\Diamond(e, \Phi)]\!] \leq [\![\Gamma, e.\Psi(e)]\!]$$

which is what we want. □

5.3 The Categorical Logic Correspondence

Proposition 5.3.1 For each FIX theory Th over some FIX signature Sg, we may construct a syntactic FIX hyperdoctrine, which we shall denote by $\mathcal{C}(Th)$ or sometimes $\mathcal{F}$.

Proof The objects of the base FIX category are the types of Sg. The morphisms are equivalence classes of terms in a single variable context. We set

$$\mathcal{F}(\alpha, \beta) \stackrel{\text{def}}{=} \{M(x) \mid x{:}\,\alpha \vdash M(x){:}\,\beta\}/=$$

where $M(x) = M'(y)$ iff $x{:}\,\alpha \vdash M(x) =_\beta M'(x)$ is a theorem of Th. We now define an $\mathcal{F}$ indexed poset. For each object $\alpha \in \mathcal{F}$ the underlying set of the partial order $\mathcal{F}(\alpha)$ consists of equivalence classes of propositions in a single variable context, $x{:}\,\alpha \vdash \Phi(x)$ prop. We shall often omit the context itself; with this convention we impose a preorder by asking that

$$\Phi(x) \leq \Psi(y) \quad \text{iff} \quad \Phi(x) \vdash \Psi(x).$$

The required partial order is the poset reflection of the preorder. Given a morphism $F\colon \alpha \to \beta$ in $\mathcal{F}$, the pullback function $F^*\colon \mathcal{F}(\beta) \to \mathcal{F}(\alpha)$ is defined by substitution: $F^*(\Phi(y)) \stackrel{\text{def}}{=} \Phi(F)$. We check that the conditions defining a FIX hyperdoctrine are satisfied. For condition (1), it is obvious that that the fibres are meet semi lattices and that the pullback functions preserve the structure. For condition (2), the right adjoint to $\mathsf{Fst}(z)^*\colon \mathcal{F}(\gamma) \to \mathcal{F}(\gamma \times \alpha)$, will be written $\forall\pi\colon \mathcal{F}(\gamma \times \alpha) \to \mathcal{F}(\gamma)$ where

$$\forall\pi(\Phi(z)) \stackrel{\text{def}}{=} \forall_\alpha(x.\Phi(\langle y, x\rangle)).$$

That this is the required adjoint is well known; RBC is simple to check. We check condition (3) with some care. The right adjoint to

$$\langle \mathsf{Fst}(u), \mathsf{Val}(\mathsf{Snd}(u))\rangle^*\colon \mathcal{F}(\gamma \times T\alpha) \to \mathcal{F}(\gamma \times \alpha)$$

will be written $\Box\colon \mathcal{F}(\gamma \times \alpha) \to \mathcal{F}(\gamma \times T\alpha)$ where

$$\Box(\Phi(u)) \stackrel{\text{def}}{=} \Box(\mathsf{Snd}(z), x.\Phi(\langle \mathsf{Fst}(z), x\rangle)).$$

We check that this definition gives a right adjoint, that is

$$\frac{\Phi(z) \vdash \Box(\Psi(u))}{\langle \mathsf{Fst}(u), \mathsf{Val}(\mathsf{Snd}(u))\rangle^*(\Phi(z)) \vdash \Psi(u)}$$

We have

$$\dfrac{\dfrac{\dfrac{u, \Phi(\langle \mathsf{Fst}(u), \mathsf{Val}(\mathsf{Snd}(u))\rangle) \vdash \Psi(u)}{x, y, \Phi(\langle y, \mathsf{Val}(x)\rangle) \vdash \Psi(\langle y, x\rangle)}\,(\mathtt{sub})}{e, y, \Phi(\langle y, e\rangle) \vdash \Box(e, x.\Psi(\langle y, x\rangle))}\,(\Box\mathtt{i})}{z, \Phi(\langle \mathsf{Fst}(z), \mathsf{Snd}(z)\rangle) \vdash \Box(\mathsf{Snd}(z), x.\Psi(\langle \mathsf{Fst}(z), x\rangle))}$$

The converse direction is equally easy; we omit the remaining details for condition (3). For condition (4) we define an operation

$$+\colon \mathcal{F}(\gamma \times \alpha) \times \mathcal{F}(\gamma \times \alpha') \to \mathcal{F}(\gamma \times (\alpha + \alpha')).$$

For elements $\Phi(v) \in \mathcal{F}(\gamma \times \alpha)$ and $\Psi(w) \in \mathcal{F}(\gamma \times \alpha')$ we set

$$\Phi(v) + \Psi(w) \stackrel{\text{def}}{=} (\Phi + \Psi)(P(u)),$$

where $P(u)$ is the isomorphism $P(u)\colon \gamma \times (\alpha + \alpha') \to (\gamma \times \alpha) + (\gamma \times \alpha')$. To see that this satisfies condition (4), take morphisms $M(v)\colon \gamma \times \alpha \to \delta$ and $N(w)\colon \gamma \times \alpha' \to \delta$ together with elements $\Theta(z) \in \mathcal{F}(\gamma \times \delta)$ and $\Pi(u) \in \mathcal{F}(\gamma \times (\alpha + \alpha'))$. Then it remains to show that

$$\frac{\begin{cases} \langle \mathsf{Fst}(v), \mathsf{Inl}(\mathsf{Snd}(v))\rangle^*\Pi(u) \;\&\; \Phi(v) \vdash \langle \mathsf{Fst}(v), M(v)\rangle^*\Theta(z) \\ \langle \mathsf{Fst}(w), \mathsf{Inl}(\mathsf{Snd}(w))\rangle^*\Pi(u) \;\&\; \Psi(w) \vdash \langle \mathsf{Fst}(w), N(w)\rangle^*\Theta(z) \end{cases}}{\Pi(u) \;\&\; (\Phi + \Psi)(P(u)) \vdash \langle \mathsf{Fst}(u), \{M, N\}(P(u))\rangle^*\Theta(z)}$$

which follows from careful application of the rule (+ad) and the Coproduct Equations rules. Conditions (5) and (6) are easy to verify. Moving on to condition (7), the left adjoint to $(id \times \Delta)^*: \mathcal{F}(\gamma \times \alpha \times \alpha) \to \mathcal{F}(\gamma \times \alpha)$, is defined by

$$\exists(id \times \Delta)(\Phi(u)) \stackrel{\text{def}}{=}$$
$$\Phi(\langle \mathsf{Fst}(z), \langle \mathsf{Fst}(\mathsf{Snd}(z)), \mathsf{Snd}(\mathsf{Snd}(z))\rangle\rangle) \;\&\; \mathsf{Fst}(\mathsf{Snd}(z)) = \mathsf{Snd}(\mathsf{Snd}(z)).$$

We have now given all the structure needed to specify $\mathcal{F}$; the remaining details are routine and omitted. Note that condition (10) is immediate from the construction of $\mathcal{F}$. □

Proposition 5.3.2 Given a FIX hyperdoctrine $\mathcal{C}$, then we can define a FIX theory which we denote by $Th(\mathcal{C})$.

Proof The basic ground types and basic function symbols for the base FIX category are exactly those defined in Section 2.5. For each object $A_1 \times \ldots \times A_n$ in $\mathcal{C}$ there are relation symbols $R: A_1, \ldots, A_n$ which are copies of the elements in the fibre $\mathcal{C}(A_1 \times \ldots \times A_n)$. This gives us data for a FIX signature; there is an obvious canonical structure for this signature in $\mathcal{C}$ which we denote by $\mathbf{G}$. Then the axioms of the theory $Th(\mathcal{C})$ are exactly those sequents in context which are satisfied by the canonical structure; the theorems of $Th(\mathcal{C})$ are generated by the usual rules of the FIX logic. □

Now we can state the categorical logic correspondence:

Theorem 5.3.3 Let $\mathcal{C}$ be a FIX hyperdoctrine; then there is an equivalence of hyperdoctrines (i.e. an equivalence of indexed posets)

$$Eq : \mathcal{C}(Th(\mathcal{C})) \simeq \mathcal{C} : Eq^{-1},$$

where Eq is a FIX hyperdoctrine morphism. Thus there is a categorical equivalence of base categories, together with an isomorphism of posets

$$Eq_\alpha : \mathcal{C}(Th(\mathcal{C}))(\alpha) \to \mathcal{C}(Eq(\alpha)) : (Eq^{-1})_{Eq(\alpha)}.$$

Proof For the definition of Eq and Eq^{-1} see Theorem 3.4.3. Define $Eq_\alpha(\Phi(x)) \stackrel{\text{def}}{=} [\![x.\Phi(x)]\!]_{\mathbf{G}}$ and $(Eq^{-1})_{[\![\alpha]\!]_{\mathbf{G}}}(R) \stackrel{\text{def}}{=} R(x)$ where $R \in \mathcal{C}([\![\alpha]\!]_{\mathbf{G}})$. We omit the routine details of the proof. □

5.4 The Logical Relations Hyperdoctrine

Now that we have formalised the correspondence between FIX theories and FIX hyperdoctrines, we define a new FIX hyperdoctrine and use it, together with its corresponding theory/logic, to prove Theorems 4.5.4, 4.5.5 and 4.5.6. We shall write $\mathcal{F}$ for the FIX hyperdoctrine constructed from the pure FIX logic.

Definition 5.4.1 Let $\Gamma: \mathcal{F} \to \omega\mathcal{C}po$ denote the functor which assigns to each object $\alpha \in \mathcal{F}$ its set $\Gamma(\alpha)$ of global elements equipped with the *discrete* partial order. We construct a new FIX hyperdoctrine, denoted by $\mathcal{L}r$, using a construction that is closely allied to the theory of logical relations. An object of $\mathcal{L}r$ is a triple $(D, \lhd, \alpha)$, where $D \in \omega\mathcal{C}po$, $\alpha \in \mathcal{F}$ and $\lhd$ is an inclusive subset of $D \times \Gamma(\alpha)$. A morphism $(D, \lhd, \alpha) \to (D', \lhd', \alpha')$ in $\mathcal{L}r$ is a pair (f, F), where $f: D \to D'$ in $\omega\mathcal{C}po$ and $F: \alpha \to \alpha'$ in $\mathcal{F}$, satisfying the following condition:

$$\forall d \in D. M \in \Gamma(\alpha). d \lhd M \supset f(d) \lhd' FM.$$

It remains to define a $\mathcal{L}r$ indexed poset. We shall denote the fibre at an object $(D, \lhd, \alpha)$ by $\mathcal{L}r(D, \lhd, \alpha)$. The elements of the fibre consist of all triples $(S, \trianglelefteq, \Phi(x))$, where

1. $S \in \mathcal{I}(D)$, i.e. S runs over the inclusive subsets of the ωcpo D.
2. $\Phi(x) \in \mathcal{F}(\alpha)$.
3. $\trianglelefteq \in \mathcal{I}([S \times \Gamma_{\Phi(x)}(\alpha)] \cap \lhd)$ where $\Gamma_{\Phi(x)}(\alpha) \stackrel{\text{def}}{=} \{M \in \Gamma(\alpha) \mid \vdash \Phi(M)\}$; note that $[S \times \Gamma_{\Phi(x)}(\alpha)] \cap \lhd$ is an ωcpo.

The order is given by inclusion in the first and second coordinates, and by entailment in the third. Given a morphism $(f, F): (D', \lhd', \alpha') \to (D, \lhd, \alpha)$ in $\mathcal{L}r$, we define the pullback function

$$(f, F)^*: \mathcal{L}r(D, \lhd, \alpha) \to \mathcal{L}r(D', \lhd', \alpha')$$

by

$$(f, F)^*(S, \trianglelefteq, \Phi(x)) \stackrel{\text{def}}{=} (f^{-1}(S), \trianglelefteq^*, \Phi(F))$$

where $\trianglelefteq^* \stackrel{\text{def}}{=} \{(d, M) \in [f^{-1}(S) \times \Gamma_{\Phi(F)}(\alpha')] \cap \lhd' \mid f(d) \trianglelefteq FM\}$.

This completes the definition of the logical relations hyperdoctrine, $\mathcal{L}r$. Clearly we need to see that $\mathcal{L}r$ really is a FIX hyperdoctrine:

Proposition 5.4.2 The construction of $\mathcal{L}r$ detailed in Definition 5.4.1 gives rise to a FIX hyperdoctrine.

Proof It is a simple exercise to verify that this definition makes sense. We check that each $(f, F)^*$ is indeed monotone. Suppose that $(S_1, \trianglelefteq_1, \Phi_1(x)) \leq (S_2, \trianglelefteq_2, \Phi_2(x))$ in the fibre $\mathcal{L}r(D, \lhd, \alpha)$.Then,

- $S_1 \subset S_2$ and so $f^{-1}(S_1) \subset f^{-1}(S_2)$.
- $\Phi_1(x) \vdash \Phi_2(x)$ and so $\Phi_1(F) \vdash \Phi_2(F)$ by substitution.
- To see that $\trianglelefteq_1^* \subset \trianglelefteq_2^*$ note that $\Phi_1(FM) \vdash \Phi_2(FM)$.

We also need to see that given

$$(D'', \lhd'', \alpha'') \xrightarrow{(f',F')} (D', \lhd', \alpha') \xrightarrow{(f,F)} (D, \lhd, \alpha)$$

it is the case that $(f', F')^* \circ (f, F)^* = (ff', FF')^*$. The details are tedious but easy and so they are omitted. Now we have to check that the definitions of objects and morphisms yield a base FIX category, and that the $\mathcal{L}r$ indexed poset does indeed constitute a FIX hyperdoctrine. Firstly we check that we have a FIX category; most of the details are simple calculations, once it is clear how one defines the various categorical constructs.

The terminal object is $(1, \lhd_{unit}, unit)$ where $* \lhd_{unit} id_{unit}$. The binary product is given by

$$(D, \lhd, \alpha) \times (D', \lhd', \alpha') \stackrel{\text{def}}{=} (D \times D', \lhd \times \lhd', \alpha \times \alpha'),$$

where (putting $\lhd_\times \stackrel{\text{def}}{=} \lhd \times \lhd'$),

$$(d, d') \lhd_\times N \text{ iff } d \lhd \mathsf{Fst}(N) \text{ and } d' \lhd' \mathsf{Snd}(N).$$

It is clear that $\lhd_\times$ is inclusive, and easy to check the remaining details. Exponentials of objects are defined by

$$(D', \lhd', \alpha',) \to (D, \lhd, \alpha) \stackrel{\text{def}}{=} (D' \to D, \lhd' \to \lhd, \alpha' \to \alpha),$$

where

$$f \lhd' \to \lhd F \quad \text{iff} \quad \forall d' \in D'. \forall L' \in \Gamma(\alpha'). d' \lhd' L' \supset f(d') \lhd ap \circ \langle F, L' \rangle$$

and ap is the evaluation morphism in $\mathcal{F}$. The transpose rule is given by

$$\frac{(f, F): (D \times D', \lhd_\times, \alpha \times \alpha') \to (D'', \lhd'', \alpha'')}{(cur(f), cur(F)): (D, \lhd, \alpha) \to (D' \to D'', \lhd' \to \lhd'', \alpha' \to \alpha'')}$$

and the evaluation morphism is (ap, ap). Finite coproducts are also defined in the same (hopefully now familiar) coordinatewise/logical relations manner. The NNO of $\mathcal{L}r$ is specified by $(\mathbb{N}, \lhd_{nat}, nat)$, where $n \lhd_{nat} N$ iff $N = \mathsf{Suc}^n(\mathsf{O})$,[1] and the zero and successor morphisms are the expected coordinatewise ones. We now show that for a particular choice of monad, the category $\mathcal{L}r$ does indeed become a let category. The action of the monad on objects is specified by $T(D, \lhd, \alpha) \stackrel{\text{def}}{=} (D_\perp, \lhd_T, T\alpha)$, where

$$e \lhd_T E \quad \text{iff} \quad \forall d \in D. [d] = e \supset \exists M \in \Gamma(\alpha). d \lhd M \;\&\; \eta_\alpha M = E,$$

and $\eta_{(D, \lhd, \alpha)} \stackrel{\text{def}}{=} (\iota, \eta_\alpha): (D, \lhd, \alpha) \to (D_\perp, \lhd, T\alpha)$, with $\iota: D \to D_\perp$ the canonical inclusion. Finally the lifting rule is

$$\frac{(f, F): (D \times D', \lhd_\times, \alpha \times \alpha') \to (D''_\perp, \lhd''_T, T\alpha'')}{(f_\perp, lift(F)): (D \times D'_\perp, \lhd \times \lhd'_T, \alpha \times T\alpha') \to (D''_\perp, \lhd''_T, T\alpha'')}$$

[1] We shall often drop the sequent symbol $\vdash$ from equalities such as $\vdash N = \mathsf{Suc}^n(\mathsf{O})$

where of course $f_\perp(d,[d']) = f(d,d')$ and $f_\perp(d,\perp) = \perp$.
Now we show that $\mathcal{L}r$ does indeed possess a FPO. This will be determined up to isomorphism; thus as for the previous constructs we exhibit a candidate and show that it satisfies the required properties. The expected candidate for the FPO would be $(\Omega, \lhd_{fix}, fix)$, with structure morphism (σ,σ). By definition of the action of the monad on objects, in the relation $\lhd_T^{fix}$ one has $\perp \lhd_T^{fix} M$, for any $M \in \Gamma(Tfix)$. As (σ,σ) must preserve the relation, then $0 \lhd_{fix} \sigma M$ must hold, and the action of the monad yields $[0] \lhd_T^{fix} \eta\sigma M$. Once again (σ,σ) preserves this, so we must have $1 \lhd_{fix} \sigma\eta\sigma M$. In general we are forced to have $n \lhd_{fix} (\sigma\eta)^n\sigma M$. Finally, considering that the relation $\lhd_{fix}$ has to be a certain inclusive subset, we are led to the following definition:
$(\Omega, \lhd_{fix}, fix)$ is a FPO for T over $\mathcal{L}r$, where

- $n \lhd_{fix} N$ iff $\exists M \in \Gamma(fix).N = (\sigma\eta)^n M$, and
- $\top \lhd_{fix} N$ iff $\forall n \in \Omega \setminus \{\top\}.n \lhd_{fix} N$.

We check that the relation $\lhd_{fix}$ is inclusive. Take a chain in $\lhd_{fix}$, say $\{n_r \lhd_{fix} N \mid r \in \mathbb{N}\}$ where $N \in \Gamma(fix)$. We need to check that $\bigvee\{n_r \mid r \in \mathbb{N}\} \lhd_{fix} N$. If $\bigvee\{n_r \mid r \in \mathbb{N}\}$ is not $\top$ we are done. Otherwise, given any $n \in \Omega \setminus \{\top\}$, we can choose $r \in \mathbb{N}$ such that $n_r \geq n$. As $n_r \lhd_{fix} N$, we get

$$N = (\sigma\eta)^{n_r} M = (\sigma\eta)^n (\sigma\eta)^{n_r - n} M,$$

and so $n \lhd_{fix} N$. As n was arbitrary, we are done. Now we check that (σ,σ) is a morphism in $\mathcal{L}r$, where $(\sigma,\sigma)\colon (\Omega_\perp, \lhd_T^{fix}, Tfix) \to (\Omega, \lhd_{fix}, fix)$. We have three cases to cover.

1. If $\perp \lhd_T^{fix} E$ then $\sigma(\perp) = 0 \lhd_{fix} \sigma E$.
2. If $[n] \lhd_T^{fix} E$ note that $E = \eta(\sigma\eta)^n M$ for some M.
3. Suppose that $[\top] \lhd_T^{fix} E$. Then $\top \lhd_{fix} N$ and hence $\forall n \in \Omega \setminus \{\top\}.n \lhd_{fix} N$. In particular, we have $n-1 \lhd_{fix} N$, and so there is some $M \in \Gamma(fix)$ for which $N = (\sigma\eta)^{n-1} M$, giving $\sigma\eta N = (\sigma\eta)^n M$. So we have $\forall n \in \Omega \setminus \{\top\}.n \lhd_{fix} \sigma\eta N$, that is $\top \lhd_{fix} \sigma E$.

Finally, we have to verify that our definition yields an initial T algebra in $\mathcal{L}r$. Take $(f,F)\colon (D_\perp, \lhd_T, T\alpha) \to (D, \lhd, \alpha)$. The unique mediating morphism for (σ,σ) has to be $(\tilde{f},\tilde{F}) \stackrel{\text{def}}{=} (it(f), it(F))$ whose coordinates are the mediating morphisms in $\omega\mathcal{C}po$ and $\mathcal{F}$. Firstly we check that it is a morphism in $\mathcal{L}r$. Suppose that $n \lhd_{fix} N$. Then for some M we get $N = (\sigma\eta)^n M$. From the definition of the $\lhd_T$ relation, we get $\perp \lhd_T let(\eta\tilde{F})\sigma^{-1}M$ and so $f(\perp) \lhd F let(\eta\tilde{F})\sigma^{-1}M = \tilde{F}M$. Now suppose that $f^r(\perp) \lhd \tilde{F}(\sigma\eta)^{r-1}M$, where $r \leq n-1$. Clearly $[f^r(\perp)] \lhd_T \eta\tilde{F}(\sigma\eta)^{r-1}M$, and so $f^{r+1}(\perp) \lhd \tilde{F}(\sigma\eta)^r M$. Inductively we have $f^{n+1}(\perp) \lhd \tilde{F}(\sigma\eta)^n M = \tilde{F}N$, which is

what we had to prove. Finally, if $\top \lhd_{fix} N$ we need $\bigvee f^n(\bot) \lhd \tilde{F}N$, which follows from inclusivity of $\lhd$. We omit to verify that the morphisms

$$([\top],\omega)\colon (1, \lhd_{unit}, unit) \to (\Omega_\bot, \lhd_T^{fix}, Tfix) \qquad (\sigma,\sigma)\colon (\Omega_\bot, \lhd_T^{fix}, Tfix) \to (\Omega, \lhd_{fix}, fix)$$

constitute a FPO in $\mathcal{L}r$.

We now verify conditions (1) to (10) of Definition 5.1.1. It is easy to see that condition (1) holds. We check condition (2) in detail. Firstly we define the right adjoint to

$$(\pi, \mathsf{Fst}(z))^*\colon \mathcal{L}r(C, \lhd', \gamma) \to \mathcal{L}r(C \times D, \lhd' \times \lhd, \gamma \times \alpha)$$

which we denote by

$$\forall\pi\colon \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha) \to \mathcal{L}r(C, \lhd', \gamma)$$

where we set $\forall\pi(S, \trianglelefteq_\times, \Phi(z)) \stackrel{\text{def}}{=} (\forall\pi(S), \trianglelefteq^\circ_\times, \forall\pi(\Phi(z)))$ with

$$\begin{aligned}\trianglelefteq^\circ_\times \quad \stackrel{\text{def}}{=} \quad & \{(c,N) \in [\forall\pi(S) \times \Gamma_{\forall\pi(\Phi(z))}(\gamma)] \cap \lhd' \mid \\ & \forall d \in D. \forall M \in \Gamma(\alpha). d \lhd M \supset (c,d) \trianglelefteq_\times \langle M, N\rangle\}.\end{aligned}$$

This makes sense. For certainly $\forall\pi(S) \in \mathcal{I}(C)$ and $\forall\pi(\Phi(z)) \in \mathcal{F}(\gamma)$. Let us see that $\trianglelefteq^\circ_\times \in \mathcal{I}([\forall\pi(S) \times \Gamma_{\forall\pi(\Phi(z))}(\gamma)] \cap \lhd')$. With the obvious notation, let $\{(c_i, N) \mid i \in \mathbb{N}\}$ be a chain in $\trianglelefteq^\circ_\times$ and so $\forall d \in D. \forall M \in \Gamma(\alpha). d \lhd M$ we have $\{((c_i, d), \langle N, M\rangle) \mid i \in \mathbb{N}\}$ in $\trianglelefteq_\times$. But $\trianglelefteq_\times$ is inclusive, so the supremum of the latter chain lies in $\trianglelefteq_\times$, for all d and M such that $d \lhd M$. But this is exactly that $(\bigvee\{c_i \mid i \in \mathbb{N}\}, N) \in \trianglelefteq^\circ_\times$. Finally, it is clear that $\forall\pi$ is monotone. Now we verify that $(\pi, \mathsf{Fst}(z)) \dashv \forall\pi$. Take $(U, \trianglelefteq', \Psi(y)) \in \mathcal{L}r(C, \lhd', \gamma)$ and $(S, \trianglelefteq_\times, \Phi(z)) \in \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha)$. Then we need to check that

$$\frac{(\pi^{-1}(U), (\trianglelefteq')^*, \Psi(\mathsf{Fst}(z))) \leq (S, \trianglelefteq_\times, \Phi(z))}{(U, \trianglelefteq', \Psi(y)) \leq (\forall\pi(S), \trianglelefteq^\circ_\times, \forall\pi(\Phi(z)))}$$

It is clear that all is well in the first and third coordinates, as both $\omega\mathcal{C}po$ and $\mathcal{F}$ are FIX hyperdoctrines. All we need to do is examine the second coordinates.

Suppose that $(\trianglelefteq')^* \subset \trianglelefteq_\times$. Hence it remains to show $\trianglelefteq' \subset \trianglelefteq^\circ_\times$. Let $c \trianglelefteq' N$. Thus we need

1. $c \in \forall\pi(S)$,
2. $N \in \Gamma_{\forall(x.\Phi(\langle y,x\rangle))}(\gamma)$,
3. $c \lhd' N$,
4. $\forall d \in D. \forall M \in \Gamma(\alpha). d \lhd M \supset (c,d) \trianglelefteq_\times \langle M, N\rangle$.

We check each of these in turn.

1. Clearly $c \in U \subset \forall\pi(S)$.

2. We are given $\vdash \Psi(N)$ and $\Psi(y) \vdash \forall_\alpha(x.\Phi(\langle y, x\rangle))$, hence $\vdash \forall_\alpha(x.\Phi(\langle N, x\rangle))$.

3. Immediate.

4. By definition

$$(\trianglelefteq')^* = \{((c,d),\langle N, M\rangle) \in [\pi^{-1}(U) \times \Gamma_{\Psi(\mathsf{Fst}(z))}(\gamma \times \alpha)] \cap \lhd_\times |\ c \trianglelefteq' N\}.$$

Suppose that $d \lhd M$. We have $c \in U$, so $(c,d) \in \pi^{-1}(U)$ and $\vdash \Psi(N)$ implying $\langle N, M\rangle \in \Gamma_{\Psi(\mathsf{Fst}(z))}(\gamma \times \alpha)$. Also $c \lhd' N$ and so $(c,d) \lhd_\times \langle N, M\rangle$. With these facts, we see that $(c,d)(\trianglelefteq')^*\langle N, M\rangle$ and using the hypothesis we are done.

Now suppose that $\trianglelefteq' \subset \trianglelefteq^\circ_\times$; it remains to show $(\trianglelefteq')^* \subset \trianglelefteq_\times$. Let $(c,d)(\trianglelefteq')^*\langle N, M\rangle$. Then from the supposition we get $c \trianglelefteq^\circ_\times N$, which means that $(c,d) \trianglelefteq_\times \langle N, M\rangle$. So we do get the required right adjoint.

We must verify that RBC holds, namely that the square

$$\begin{array}{ccc} \mathcal{L}r(C \times D, \lhd' \times \lhd, \gamma \times \alpha) & \xrightarrow{\forall\pi} & \mathcal{L}r(C, \lhd', \gamma) \\ \Big\downarrow{\scriptstyle (f \times id, F \times id)^*} & & \Big\downarrow{\scriptstyle (f,F)^*} \\ \mathcal{L}r(C' \times D, \lhd'' \times \lhd, \gamma' \times \alpha) & \xrightarrow[\forall\pi]{} & \mathcal{L}r(C', \lhd'', \gamma') \end{array}$$

Of course, we already know that everything is fine in the first and third coordinates. We just need to check the second coordinate. We have that

$$\begin{aligned} \trianglelefteq^\circ_\times &= \{(c,N) \in [\forall\pi(S) \times \Gamma_{\forall\pi(\Phi(z))}(\gamma)] \cap \lhd' |\ \forall d.\forall M.d \lhd M \supset (c,d) \trianglelefteq_\times \langle N, M\rangle\} \\ \trianglelefteq^*_\times &= \{((c',d),\langle N', M\rangle) \in [(f \times id)^{-1}(S) \times \Gamma_{(F\times id)^*(\Phi(z))}(\gamma' \times \alpha)] \cap \lhd'' \times \lhd | \\ &\quad (f \times id)(c',d) \trianglelefteq_\times (F \times id)\langle N', M\rangle\} \\ (\trianglelefteq^\circ_\times)^* &= \{(c',N') \in [f^{-1} \circ \forall\pi(S) \times \Gamma_{F^* \circ \forall\pi(\Phi(z))}(\gamma')] \cap \lhd'' |\ f(c') \trianglelefteq^\circ_\times FN'\} \\ (\trianglelefteq^*_\times)^\circ &= \{(c',N') \in [\forall\pi \circ (f \times id)^{-1}(S) \times \Gamma_{\forall\pi\circ(F\times id)^*(\Phi(z))}(\gamma')] \cap \lhd'' | \\ &\quad \forall d.\forall M.d \lhd M \supset (c',d) \trianglelefteq^*_\times \langle N', M\rangle\} \end{aligned}$$

Suppose that $c'(\trianglelefteq^\circ_\times)^*N'$. We show $c'(\trianglelefteq^*_\times)^\circ N'$. Hence we need to see that $d \lhd M$ implies $(c',d) \trianglelefteq^*_\times \langle N', M\rangle$. This amounts to showing

1. $(c',d) \in (f \times id)^{-1}(S)$.

2. $\langle N', M\rangle \in \Gamma_{\Phi(\langle F(\mathsf{Fst}(z')),\mathsf{Snd}(z')\rangle)}(\gamma' \times \alpha)$.

3. $(c',d) \lhd'' \times \lhd \langle N', M\rangle$.

4. $(f(c'),d) \trianglelefteq_\times \langle FN', M\rangle$.

We check each of these:

1. Note that $c' \in f^{-1} \circ \forall\pi(S) = \forall\pi \circ (f \times id)^{-1}(S)$.
2. We have $N' \in \Gamma_{\forall_\alpha(x.\Phi(\langle F,x\rangle))}$ and hence $\vdash \Phi(\langle FN', M\rangle)$.
3. Immediate from $c' \lhd'' N'$.
4. Immediate from $f(c') \trianglelefteq^\circ_\times FN'$.

Thus $(\trianglelefteq^\circ_\times)^* \subset (\trianglelefteq^*_\times)^\circ$; the reverse inclusion is similar.
We move on to condition (3). The right adjoint to

$$(id \times \iota, id \times \eta)^*: \mathcal{L}r(C \times D_\perp, \lhd' \times \lhd_T, \gamma \times T\alpha) \to \mathcal{L}r(C \times D, \lhd' \times \lhd, \gamma \times \alpha)$$

which we shall write as

$$\Box: \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha) \to \mathcal{L}r(C \times D_\perp, \lhd^T_\times, \gamma \times T\alpha)$$

is defined by $\Box(S, \trianglelefteq_\times, \Phi(u)) \stackrel{\text{def}}{=} (\Box(S), \trianglelefteq^\circ_\times, \Box(\Phi(u)))$ where $\Box(S)$ and $\Box(\Phi(u))$ have the expected meaning, and where

$$\begin{aligned}\trianglelefteq^\circ_\times \quad \stackrel{\text{def}}{=} \quad & \{((c,e),\langle N,E\rangle) \in [\Box(S) \times \Gamma_{\Box(\Phi(u))}(\gamma \times T\alpha)] \cap \lhd^T_\times | \\ & \forall d.\forall M.d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \supset (c,d) \trianglelefteq_\times \langle N, M\rangle\}.\end{aligned}$$

We omit to check that the definition of $\Box$ makes sense, and that it is the required adjoint. We shall just check that RBC does indeed hold, i.e. the following square commutes:

$$\begin{array}{ccc} \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha) & \xrightarrow{\Box} & \mathcal{L}r(C \times D_\perp, \lhd^T_\times, \gamma \times T\alpha) \\ \Big\downarrow{\scriptstyle ((f \times id),(F \times id))^*} & & \Big\downarrow{\scriptstyle ((f \times id),(F \times id))^*} \\ \mathcal{L}r(C' \times D, \lhd'' \times \lhd, \gamma' \times \alpha) & \xrightarrow[\Box]{} & \mathcal{L}r(C' \times D_\perp, \lhd'' \times \lhd_T, \gamma' \times T\alpha) \end{array}$$

Once again, all we need to do is check things work in the second coordinate. We have:

$$\begin{aligned}\trianglelefteq^\circ_\times \quad = \quad & \{((c,e),\langle N,E\rangle) \in [\Box(S) \times \Gamma_{\Box(\Phi(u))}(\gamma \times T\alpha)] \cap \lhd^T_\times | \\ & \forall d.\forall M.d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \supset (c,d) \trianglelefteq_\times \langle N, M\rangle\}\end{aligned}$$

$$\begin{aligned}\trianglelefteq^*_\times \quad = \quad & \{((c',d),\langle N',M\rangle) \in [(f \times id)^{-1}(S) \times \Gamma_{(F\times id)^*(\Phi(u))}(\gamma' \times \alpha)] \cap \lhd'' \times \lhd | \\ & (f(c'),d) \trianglelefteq_\times \langle FN', M\rangle\}\end{aligned}$$

$$\begin{aligned}(\trianglelefteq^\circ_\times)^* \quad = \quad & \{((c',e),\langle N',E\rangle) \in [(f \times id)^{-1} \circ \Box(S) \times \Gamma_{(F\times id)^* \circ \Box(\Phi(u))}(\gamma' \times T\alpha)] \\ & \cap \lhd'' \times \lhd_T | \; (f(c'),e) \trianglelefteq^\circ_\times \langle FN', E\rangle\}\end{aligned}$$

$$\begin{aligned}(\trianglelefteq^*_\times)^\circ \quad = \quad & \{((c',e),\langle N',E\rangle) \in [\Box \circ (f \times id)^{-1}(S) \times \Gamma_{\Box\circ(F\times id)^*(\Phi(u))}(\gamma' \times T\alpha)] \\ & \cap \lhd'' \times \lhd_T | \; \forall d.\forall M.d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \supset (c',d) \trianglelefteq^*_\times \langle N', M\rangle\}\end{aligned}$$

Suppose that $(c', e)(\trianglelefteq^{\circ}_{\times})^{*}\langle N', E\rangle$. Then it remains to prove $(c', e)(\trianglelefteq^{*}_{\times})^{\circ}\langle N', E\rangle$. This means given any $d \in D$ and $M \in \Gamma(\alpha)$ for which $d \lhd M$, $e = [d]$ and $E = \mathsf{Val}(M)$ we must show

1. $(c', d) \in (f \times id)^{-1}(S)$.
2. $\langle N', M\rangle \in \Gamma_{(F\times id)^{*}(\Phi(u))}(\gamma' \times \alpha)$.
3. $(c', d) \lhd'' \times \lhd \langle N', M\rangle$.
4. $(f(c'), d) \trianglelefteq_{\times} \langle FN', M\rangle$.

Using the hypothesis, we have $(f(c'), e) \trianglelefteq^{\circ}_{\times} \langle FN', E\rangle$ and hence

$$d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \supset (f(c'), d) \trianglelefteq_{\times} \langle FN', M\rangle.$$

We check each of 1 to 4:

1. $(c', [d]) \in (f \times id)^{-1} \circ \Box(S) = \Box \circ (f \times id)^{-1}(S)$ and so $(c', d) \in (f \times id)^{-1}(S)$.
2. $\langle N', E\rangle \in \Gamma_{\Box(\mathsf{Snd}(z),x.\Phi(\langle F\mathsf{Fst}(z),x\rangle))}(\gamma' \times T\alpha)$, which means that
$$\vdash \Box(E, x.\Phi(\langle FN', x\rangle)).$$
But $E = \mathsf{Val}(M)$, thus appealing to Proposition 4.5.1 we have $\vdash \Phi(\langle FN', M\rangle)$.
3. By hypothesis.
4. Immediate.

Suppose that $(c', e)(\trianglelefteq^{*}_{\times})^{\circ}\langle N', E\rangle$. It remains to prove $(c', e)(\trianglelefteq^{\circ}_{\times})^{*}\langle N', E\rangle$. Thus given $d \in D$ and $M \in \Gamma(\alpha)$ for which $e = [d]$, $E = \mathsf{Val}(M)$ and $d \lhd M$, we need to show

1. $(f(c'), e) \in \Box(S)$,
2. $\langle FN', E\rangle \in \Gamma_{\Box(\Phi(u))}(\gamma \times T\alpha)$,
3. $(f(c'), e) \lhd^{T}_{\times} \langle FN', E\rangle$,
4. $(f(c'), d) \trianglelefteq_{\times} \langle FN', M\rangle$.

Using the hypothesis, we get

$$d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \supset (f(c'), d) \trianglelefteq_{\times} \langle FN', M\rangle.$$

We check each of 1 to 4:

1. $(c', [d]) \in \Box \circ (f \times id)^{-1}(S) = (f \times id)^{-1} \circ \Box(S)$.
2. We have $\langle N', E\rangle \in \Gamma_{\Box(\mathsf{Snd}(z),x.\Phi(\langle F\mathsf{Fst}(z),x\rangle))}(\gamma' \times T\alpha)$, and so
$$\vdash \Box(E, x.\Phi(\langle FN', x\rangle)).$$
But $E = \mathsf{Val}(M)$ and so $\vdash \Phi(\langle FN', M\rangle)$ by appeal to Proposition 4.5.1.

3. Trivial.

4. Immediate.

We now define the left adjoint to

$$(id \times \iota, id \times \eta)^*: \mathcal{L}r(C \times D_\perp, \lhd' \times \lhd_T, \gamma \times T\alpha) \to \mathcal{L}r(C \times D, \lhd' \times \lhd, \gamma \times \alpha).$$

Denote this by

$$\Diamond: \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha) \to \mathcal{L}r(C \times D_\perp, \lhd^T_\times, \gamma \times T\alpha)$$

where we set $\Diamond(S, \trianglelefteq_\times, \Phi(u)) \stackrel{\text{def}}{=} (\Diamond(S), \trianglelefteq^\circ_\times, \Diamond(\Phi(u)))$ with

$$\begin{aligned} \trianglelefteq^\circ_\times \stackrel{\text{def}}{=} \{((c,e),\langle N,E\rangle) \in [\Diamond(S) \times \Gamma_{\Diamond(\Phi(u))}(\gamma \times T\alpha)] \cap \lhd^T_\times \mid \\ \exists d.\exists M.d \lhd M \;\&\; e = [d] \;\&\; E = \mathsf{Val}(M) \;\&\; (c,d) \trianglelefteq_\times \langle N,M\rangle)\} \end{aligned}$$

We omit to check that this is well defined and that LBC is satisfied, but give brief details of the Frobenius Reciprocity condition. Take

$$(U, \trianglelefteq^T_\times, \Psi(z)) \in \mathcal{L}r(C \times D_\perp, \lhd^T_\times, \gamma \times T\alpha)$$

and

$$(S, \trianglelefteq_\times, \Phi(u)) \in \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha).$$

Observe that $U \cap \Diamond(S) = \Diamond((id \times \iota)^{-1}(U) \cap S)$ and

$$\Psi(\langle y,e\rangle) \;\&\; \Diamond(e, x.\Phi(\langle y,x\rangle)) \dashv \vdash \Diamond(e, x.\Psi(\langle y, \mathsf{Val}(x)\rangle) \;\&\; \Phi(\langle y,x\rangle)).$$

We require $((\trianglelefteq^T_\times)^* \cap \trianglelefteq_\times)^\circ = \trianglelefteq^T_\times \cap \trianglelefteq^\circ_\times$. Suppose that $(c,e)((\trianglelefteq^T_\times)^* \cap \trianglelefteq_\times)^\circ\langle N,E\rangle$. Then there exist d and M for which $(c,d)((\trianglelefteq^T_\times)^* \cap \trianglelefteq_\times)\langle N,M\rangle$. Unravelling the supposition and appealing to the above observation, we conclude that $(c,e) \in U \cap \Diamond(S)$ and $\vdash \Psi(\langle N,E\rangle) \;\&\; \Diamond(E, x.\Phi(\langle N,x\rangle))$. Using Proposition 4.5.1 we may deduce $\vdash \Psi(\langle N,E\rangle)$ and $\vdash \Phi(\langle N,M\rangle)$. So we can conclude that $(c,e) \trianglelefteq^T_\times \langle N,E\rangle$ and $(c,e) \trianglelefteq^\circ_\times \langle N,E\rangle$.

Suppose that $(c,e)(\trianglelefteq^T_\times \cap \trianglelefteq^\circ_\times)\langle N,E\rangle$. We need to find d and M for which $(c,d)(\trianglelefteq^T_\times)^*\langle N,M\rangle$ and $(c,d) \trianglelefteq_\times \langle N,M\rangle$. Now $(c,e) \trianglelefteq^\circ_\times \langle N,E\rangle$ implies that there are d and M for which $d \lhd M$, $e = [d]$ and $E = \mathsf{Val}(M)$. From the observation above, we deduce that $(c,e) \in \Diamond((id \times \iota)^{-1} \cap S)$ and $\vdash \Diamond(E, x.\Psi(\langle N, \mathsf{Val}(x)\rangle) \;\&\; \Phi(\langle N,x\rangle))$. Thus $(c,d) \in (id \times \iota)^{-1}(U) \cap S$ and appealing to Proposition 4.5.1 we deduce $\vdash \Psi(\langle N,E\rangle) \;\&\; \Phi(\langle N,M\rangle)$. Using these conclusions along with the supposition we are done.

Now for condition (4). We define the operation $+$

$$\begin{aligned} +: \mathcal{L}r(C \times D, \lhd_C \times \lhd, \gamma \times \alpha) \times \mathcal{L}r(C \times D', \lhd_C \times \lhd', \gamma \times \alpha') \to \\ \mathcal{L}r(C \times (D + D'), \lhd_C \times (\lhd + \lhd'), \gamma \times (\alpha + \alpha')) \end{aligned}$$

by sending the elements

$$\begin{aligned}(I, \trianglelefteq_1, \Phi(v)) &\in \mathcal{L}r(C \times D, \lhd_C \times \lhd, \gamma \times \alpha)\\ (J, \trianglelefteq_2, \Psi(w)) &\in \mathcal{L}r(C \times D', \lhd_C \times \lhd', \gamma \times \alpha')\end{aligned}$$

to $(I + J, \trianglelefteq_3, \Phi(v) + \Psi(w))$ where

$$\begin{aligned}\trianglelefteq_3 \stackrel{\text{def}}{=} \{((c,e),\langle N,E\rangle) \in [(I+J) \times \Gamma_{\Phi(v)+\Psi(w)}(\gamma \times (\alpha+\alpha'))] \cap \lhd_C \times (\lhd + \lhd') \mid \\ \exists d.\exists M.d \lhd M \;\&\; e = i(d) \;\&\; E = \mathsf{Inl}(M) \;\&\; (c,d) \trianglelefteq_1 \langle N,M\rangle\end{aligned}$$

or

$$\exists d'.\exists M'.d' \lhd' M' \;\&\; e = j(d') \;\&\; E = \mathsf{Inr}(M') \;\&\; (c,d') \trianglelefteq_2 \langle N,M'\rangle\}.$$

We omit to check that condition (4) is satisfied and move to condition (5). The left adjoint to

$$\begin{aligned}(id \times \Delta, id \times \Delta)^*\colon \mathcal{L}r(C \times D \times D, \lhd' \times \lhd \times \lhd, \gamma \times \alpha \times \alpha) \to \\ \mathcal{L}r(C \times D, \lhd' \times \lhd, \gamma \times \alpha)\end{aligned}$$

denoted by

$$\exists(id \times \Delta)\colon \mathcal{L}r(C \times D, \lhd_\times, \gamma \times \alpha) \to \mathcal{L}r(C \times D \times D, \lhd'_\times, \gamma \times \alpha \times \alpha)$$

is defined by setting

$$\exists(id \times \Delta)(S, \trianglelefteq_\times, \Phi(u)) \stackrel{\text{def}}{=} (\exists(id \times \Delta)(S), \trianglelefteq^\circ_\times, \exists(id \times \Delta)(\Phi(u))),$$

where

$$\begin{aligned}\trianglelefteq^\circ_\times \stackrel{\text{def}}{=} \{((c,e),\langle N,E\rangle) \in [\exists(id \times \Delta)(S) \times \Gamma_{\exists(id\times\Delta)(\Phi(u))}(\gamma \times \alpha \times \alpha)] \cap \lhd'_\times \mid \\ \exists d.\exists M.d \lhd M \;\&\; e = \Delta(d) \;\&\; E = \Delta M \;\&\; (c,d) \trianglelefteq_\times \langle N,M\rangle\}.\end{aligned}$$

Let us check that we have defined the required adjoint. Take an element

$$(U, \trianglelefteq'_\times, \Psi(z)) \in \mathcal{L}r(C \times D \times D, \lhd'_\times, \gamma \times \alpha \times \alpha).$$

Of course we only need to check details in the middle coordinate. The action of $(id \times \Delta, id \times \Delta)^*$ on this element gives

$$\begin{aligned}(\trianglelefteq'_\times)^* \stackrel{\text{def}}{=} \{((c,d),\langle N,M\rangle) \in [(id \times \Delta)^{-1}(U) \times \Gamma_{(id\times\Delta)^*(\Psi(z))}(\gamma \times \alpha)] \cap \lhd_\times \mid \\ (c,d,d) \trianglelefteq'_\times \langle N,M,M\rangle\}.\end{aligned}$$

So we need to verify that

$$\frac{\trianglelefteq_\times \subset (\trianglelefteq'_\times)^*}{\trianglelefteq^\circ_\times \subset \trianglelefteq'_\times}$$

We check one direction. Suppose $\trianglelefteq^\circ_\times \subset \trianglelefteq'_\times$ and take $(c,d) \trianglelefteq_\times \langle N,M\rangle$. Then $(c,d) \in S \subset (id \times \Delta)^{-1}(U)$ and $\vdash \Phi(\langle N,M\rangle)$ hence $\vdash \Psi(\langle N,M,M\rangle)$. It is clear that

$(c,d) \lhd_\times \langle N,M\rangle$. So it remains to show that $(c,d,d) \trianglelefteq'_\times \langle N,M,M\rangle$. It will be enough to show $(c,d,d) \in \exists(id\times\Delta)(S)$ and $\vdash \Phi(\langle N,M\rangle)$ & $M = M$. But this is clear, for we have $(c,d) \in S$ and $\langle N,M\rangle \in \Gamma_{\Phi(u)}(\gamma\times\alpha)$. Finally, for Frobenius reciprocity, write $(C\times D, \trianglelefteq_\top, \top)$ for the greatest element in $\mathcal{L}r(C\times D, \lhd, \gamma\times\alpha)$. Then it is easy to see that

$$\begin{aligned}((\trianglelefteq'_\times)^*)^\circ &= \{((c,d,d),\langle N,M,M\rangle) \in \exists(id\times\Delta)\circ(id\times\Delta)^{-1}(U)\times \\ &\quad \Gamma_{\exists(id\times\Delta)\circ(id\times\Delta)^*(\Psi(z))}(\gamma\times\alpha\times\alpha)\cap \lhd'_\times |\ (c,d,d) \trianglelefteq'_\times \langle N,M,M\rangle\} \\ &= \trianglelefteq^\circ_\top \cap \trianglelefteq'_\times .\end{aligned}$$

We have now shown how to define all of the adjoints and operations needed to verify that $\mathcal{L}r$ is a FIX hyperdoctrine; we omit the remaining details. □

5.5 Proving Existence and Disjunction Properties

We shall soon prove the Existence Property, Disjunction Property, and Standardness of *nat*; these results were stated in Section 4.5.

The Initial Model of the FIX Logic

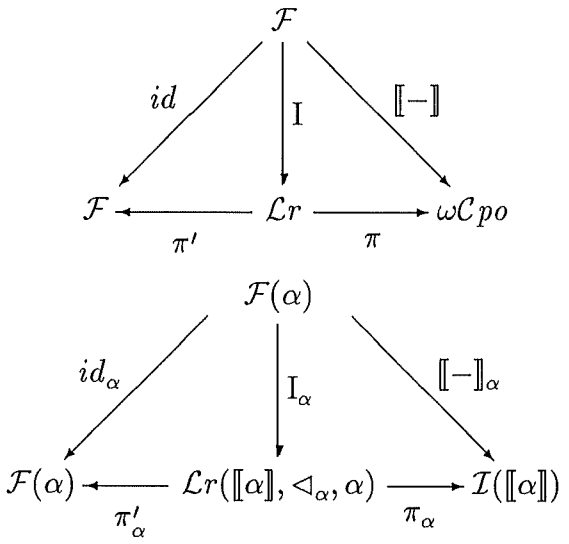
Proposition 5.5.1 The FIX hyperdoctrine $\mathcal{F}$, arising from the pure FIX logic, is (essentially) initial amongst all FIX hyperdoctrines.

Proof This is immediate from the definition of FIX hyperdoctrine morphism. □

We shall need the following observations: Using the initiality of $\mathcal{F}$, we see that there are FIX hyperdoctrine morphisms $[\![-]\!]$ and I, together with obvious projections π and π' where

$$\begin{array}{ll}[\![-]\!]\colon \mathcal{F} \to \omega\mathcal{C}po & \pi\colon \mathcal{L}r \to \omega\mathcal{C}po \\ \mathrm{I}\colon \mathcal{F} \to \mathcal{L}r & \pi'\colon \mathcal{L}r \to \mathcal{F}.\end{array}$$

These FIX hyperdoctrine morphisms satisfy the following commutative diagrams:



The Disjunction Property

We prove Theorem 4.5.5. First note that the closed term $E\colon \alpha+\beta$ corresponds to a morphism $E\colon unit \to \alpha+\beta$ in $\mathcal{F}$. The action of the base functor component of $\mathrm{I}\colon \mathcal{F} \to \mathcal{L}r$ on this morphism, using the above commutative diagrams, is

$$\mathrm{I}(E) = ([\![E]\!], E)\colon (1, \lhd_{unit}, unit) \to ([\![\alpha]\!] + [\![\beta]\!], \lhd_{\alpha+\beta}, \alpha+\beta).$$

Also, the following square commutes:

$$\begin{array}{ccc} \mathcal{F}(\alpha+\beta) & \xrightarrow{\mathrm{I}_{\alpha+\beta}} & \mathcal{L}r([\![\alpha]\!] + [\![\beta]\!], \lhd_{\alpha+\beta}, \alpha+\beta) \\ \Big\downarrow{\scriptstyle E^*} & & \Big\downarrow{\scriptstyle ([\![E]\!], E)^*} \\ \mathcal{F}(unit) & \xrightarrow[\mathrm{I}_{unit}]{} & \mathcal{L}r(1, \lhd_{unit}, unit) \end{array}$$

The theorem follows by observing the effect of the two possible routes of the square. Let z be a variable of type $\alpha+\beta$, and consider $(\Phi+\Psi)(z) \in \mathcal{F}(\alpha+\beta)$. Then we have

$$\begin{aligned} \mathrm{I}_{unit}(E^*((\Phi+\Psi)(z))) &= \mathrm{I}((\Phi+\Psi)(E)) \\ &= ([\![(\Phi+\Psi)(E)]\!], \trianglelefteq_{unit}, (\Phi+\Psi)(E)) \end{aligned}$$

where because $\vdash (\Phi+\Psi)(E)$ by hypothesis and I_{unit} preserves greatest elements, the relation $\trianglelefteq_{unit}$ must be non-empty. Also, we have

$$\begin{aligned} ([\![E]\!], E)^*(\mathrm{I}_{\alpha+\beta}((\Phi+\Psi)(z))) &= ([\![E]\!], E)^*([\![z.(\Phi+\Psi)(z)]\!], \trianglelefteq_{\alpha+\beta}, (\Phi+\Psi)(z)) \\ &= ([\![E]\!]^{-1}([\![z.(\Phi+\Psi)(z)]\!]), \trianglelefteq^*_{\alpha+\beta}, (\Phi+\Psi)(E)) \end{aligned}$$

where

$$\begin{aligned} \trianglelefteq^*_{\alpha+\beta} \;=\; & \{(*, id_{unit}) \in [[\![E]\!]^{-1}([\![z.(\Phi+\Psi)(z)]\!]) \times \Gamma_{(\Phi+\Psi)(E)}(unit)] \cap \lhd_{unit} | \\ & [\![E]\!](*) \trianglelefteq_{\alpha+\beta} E \circ id\}. \end{aligned}$$

But this relation is exactly $\trianglelefteq_{unit}$, hence is *non-empty*, yielding $[\![E]\!](*) \trianglelefteq_{\alpha+\beta} E$ which implies $[\![E]\!](*) \lhd_{\alpha+\beta} E$. By definition of the relation $\lhd_{\alpha+\beta}$ this means without loss of generality there is a global element $M \in \Gamma(\alpha)$, that is a closed term M, for which $\vdash E =_{\alpha+\beta} \mathsf{Inl}(M)$, and from this we may derive $\vdash \Phi(M)$ using Proposition 4.5.1.

The Existence Property

We prove Theorem 4.5.4. Take a proposition $\Diamond(e,\Phi) \in \mathcal{F}(T\alpha)$ and use the square

$$\begin{array}{ccc} \mathcal{F}(T\alpha) & \xrightarrow{\mathrm{I}_{T\alpha}} & \mathcal{L}r([\![\alpha]\!]_\perp, \lhd_{T\alpha}, T\alpha) \\ \Big\downarrow{\scriptstyle E^*} & & \Big\downarrow{\scriptstyle ([\![E]\!], E)^*} \\ \mathcal{F}(unit) & \xrightarrow[\mathrm{I}_{unit}]{} & \mathcal{L}r(1, \lhd_{unit}, unit) \end{array}$$

As above, this yields

$$\begin{aligned} \trianglelefteq_{unit} &= \trianglelefteq^*_{T\alpha} \\ &= \{(*, id) \in [[\![E]\!]^{-1}([\![e.\Diamond(e,\Phi)]\!]) \times \Gamma_{\Diamond(E,\Phi)}(unit)] \cap \lhd_{unit} \mid [\![E]\!](*) \trianglelefteq_{T\alpha} E\}. \end{aligned}$$

This has to be non empty and so $[\![E]\!](*) \trianglelefteq_{T\alpha} E$, which implies $[\![E]\!](*) \lhd_{T\alpha} E$. By assumption we have $\vdash \Diamond(E,\Phi)$ and therefore $\{*\} = [\![E]\!]^{-1}\eta([\![x.\Phi(x)]\!])$ implying that $[\![E]\!](*)$ is not bottom. Hence there is some closed M for which $\vdash E =_{T\alpha} \mathsf{Val}(M)$ and using Proposition 4.5.1 we have $\vdash \Phi(M)$.

A Formal Adequacy of the FIX Logic

We finish this section by remarking that the Existence Property expresses a *formal adequacy* of the FIX logic. Indeed, we have the following

Corollary 5.5.2 Given a closed term E of type $T\alpha$, it is provably equal to a value $\mathsf{Val}(M)$, where M is a closed term of type α, if and only if the $\omega\mathcal{C}po$ interpretation $[\![E]\!] \in [\![T\alpha]\!] = [\![\alpha]\!]_\perp$ is not $\perp$.

Proof Immediate from the proof of the Existence Property. □

5.6 Proving Standardness of the Natural Number Type

We prove Theorem 4.5.6. Let N be a closed term of type nat. Using the square

$$\begin{array}{ccc} \mathcal{F}(nat) & \xrightarrow{\mathrm{I}_{nat}} & \mathcal{L}r(\mathbb{N}, \lhd_{nat}, nat) \\ \Big\downarrow{\scriptstyle N^*} & & \Big\downarrow{\scriptstyle ([\![N]\!], N)^*} \\ \mathcal{F}(unit) & \xrightarrow[\mathrm{I}_{unit}]{} & \mathcal{L}r(1, \lhd_{unit}, unit) \end{array}$$

and arguing the same way as in the previous section, we conclude that $[\![N]\!](*) \trianglelefteq_{nat} N$ and from this we deduce $\vdash N =_{nat} \mathsf{Suc}^n(0)$, using the definition of the $\lhd_{nat}$ relation in the NNO of $\mathcal{L}r$.

Chapter 6
Applications of the FIX Logic

6.1 Introduction

In this chapter we shall define the syntax and operational semantics of two little programming languages, both of which are closely allied to Plotkin's PCF. PCF is an acronym for Programming Computable Functions. In essence, the syntax of PCF is that of simply typed lambda calculus (with ground types just the natural numbers and booleans) which has been enriched with explicit operations for arithmetic, a conditional at ground types and fixpoint operators. This syntax is then equipped with a call by name operational semantics, giving rise to the language PCF. PCF was first investigated by Plotkin and the results appear in [Plo77].

The two languages we investigate here, which we call QL and HPCF, resemble PCF in that their syntax consists essentially of simply typed lambda calculus with extra arithmetical, procedural and fixpoint features. They differ in having conditionals at higher types. The syntax of QL, while similar to that of PCF, makes use of higher order metaconstants. QL has recursive function declarations instead of fixpoint operators and the operational semantics is call by value. HPCF has a call by name operational semantics and apart from conditionals at higher types is identical to PCF.

We shall specify the syntax and semantics of these languages, then give a translation into a suitable FIX theory. For each language we state two adequacy results, one for static semantics and one for dynamic semantics, which shows that the translation preserves the structure of the original language. We emphasise that both QL and HPCF are no more than very simple adaptations of Plotkin's PCF. The intention of this chapter is just to investigate how well suited the FIX logic is for interpreting and reasoning about two quite standard languages. The FIX logic can be viewed as a metalogic in which we interpret both QL and HPCF; for an account of this style of programming language analysis see [Plo85].

6.2 The Language QL

We define the language QL by specifying the basic syntax of types and raw expressions; this syntax will then be given a static and dynamic semantics.

The Types and Expressions of QL

The types of QL are given by the grammar:

$$\sigma \ ::= \ \mathtt{bool} \mid \mathtt{nat} \mid \sigma \rightarrow \sigma$$

The (raw) expressions of QL are given by the grammar:

$\mathtt{m}$	$::=$	x	variables
	$\mid$	$\mathtt{tt}$	truth
	$\mid$	$\mathtt{ff}$	falsity
	$\mid$	$\mathtt{k}_n$	natural numbers
	$\mid$	$\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n})$	conditional
	$\mid$	$\mathtt{S}(\mathtt{m})$	successor
	$\mid$	$\mathtt{P}(\mathtt{m})$	predecessor
	$\mid$	$\mathtt{Z}(\mathtt{m})$	zero test
	$\mid$	$\mathtt{mn}$	application
	$\mid$	$\lambda x{:}\sigma.\mathtt{m}$	function definition
	$\mid$	$\mathtt{R}_{\tau,\sigma}(\mathtt{m},\mathtt{n})$	recursive functions

The Static Semantics of QL

The static semantics assigns types to expressions in context. Each judgement takes the form $\Gamma \vdash \mathtt{m}{:}\sigma$. The rules for deriving these judgements are given below. The context Γ consists of a list of typed variables (the variables are assumed distinct). Variables are bound in the usual way by lambda abstractions and recursive function declarations. Given a QL expression in context, $\Gamma \vdash \mathtt{m}{:}\sigma$, it is easy to see that the free variables of $\mathtt{m}$ all occur in Γ, and that the type σ assigned to the raw QL term $\mathtt{m}$ is unique. The types `nat` and `bool` will be referred to as *ground* types.

Variables

$$\frac{}{\Gamma, x{:}\sigma, \Gamma' \vdash x{:}\sigma}$$

Constants

$$\frac{}{\Gamma \vdash \mathtt{tt}{:}\mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{ff}{:}\mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{k}_n{:}\mathtt{nat}}$$

Conditional

$$\frac{\Gamma \vdash \mathtt{b}{:}\mathtt{bool} \quad \Gamma \vdash \mathtt{m}{:}\sigma \quad \Gamma \vdash \mathtt{n}{:}\sigma}{\Gamma \vdash \mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n}){:}\sigma}$$

Arithmetic

$$\frac{\Gamma \vdash \mathtt{m}{:}\mathtt{nat}}{\Gamma \vdash \mathtt{S}(\mathtt{m}){:}\mathtt{nat}} \qquad \frac{\Gamma \vdash \mathtt{m}{:}\mathtt{nat}}{\Gamma \vdash \mathtt{P}(\mathtt{m}){:}\mathtt{nat}} \qquad \frac{\Gamma \vdash \mathtt{m}{:}\mathtt{nat}}{\Gamma \vdash \mathtt{Z}(\mathtt{m}){:}\mathtt{bool}}$$

Functions

$$\frac{\Gamma \vdash \mathtt{m}: \sigma \to \tau \quad \Gamma \vdash \mathtt{n}: \sigma}{\Gamma \vdash \mathtt{mn}: \tau} \qquad \frac{\Gamma, x: \sigma \vdash \mathtt{m}: \tau}{\Gamma \vdash \lambda x: \sigma.\mathtt{m}: \sigma \to \tau}$$

Recursive Functions

$$\frac{\Gamma, f: \sigma \to \sigma', x: \sigma \vdash \mathtt{m}: \sigma' \quad \Gamma \vdash \mathtt{n}: \sigma}{\Gamma \vdash \mathtt{R}_{\sigma,\sigma'}(\mathtt{m},\mathtt{n}): \sigma'}$$

The Dynamic Semantics of QL

We call a QL expression $\mathtt{m}$ *closed* if $\vdash \mathtt{m}: \sigma$ is derivable for some (necessarily unique) type σ. The *canonical* QL expressions comprise the *subset* of closed expressions given by the grammar:

$$\mathtt{c} ::= \mathtt{tt} \mid \mathtt{ff} \mid \mathtt{k}_n \mid \lambda x: \sigma.\mathtt{m}$$

We now give the syntax of QL a call by value dynamic semantics via an *evaluation relation*, which will take the form $\mathtt{m} \Longrightarrow \mathtt{c}$, where $\mathtt{m}$ and $\mathtt{c}$ are closed QL expressions and $\mathtt{c}$ is canonical. The rules for generating the evaluation relation are given below:

Canonical Forms

$$\frac{\mathtt{c} \text{ canonical}}{\mathtt{c} \Longrightarrow \mathtt{c}}$$

Conditionals

$$\frac{\mathtt{b} \Longrightarrow \mathtt{tt} \quad \mathtt{m} \Longrightarrow \mathtt{c}}{\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n}) \Longrightarrow \mathtt{c}} \qquad \frac{\mathtt{b} \Longrightarrow \mathtt{ff} \quad \mathtt{n} \Longrightarrow \mathtt{c}}{\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n}) \Longrightarrow \mathtt{c}}$$

Arithmetic

$$\frac{\mathtt{m} \Longrightarrow \mathtt{k}_n}{\mathtt{S}(\mathtt{m}) \Longrightarrow \mathtt{k}_{n+1}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{k}_{n+1}}{\mathtt{P}(\mathtt{m}) \Longrightarrow \mathtt{k}_n} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{k}_0}{\mathtt{P}(\mathtt{m}) \Longrightarrow \mathtt{k}_0}$$

$$\frac{\mathtt{m} \Longrightarrow \mathtt{k}_0}{\mathtt{Z}(\mathtt{m}) \Longrightarrow \mathtt{tt}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{k}_{n+1}}{\mathtt{Z}(\mathtt{m}) \Longrightarrow \mathtt{ff}}$$

Functions

$$\frac{\mathtt{m} \Longrightarrow \lambda x: \sigma.\mathtt{m}' \quad \mathtt{n} \Longrightarrow \mathtt{c}' \quad \mathtt{m}'[\mathtt{c}'/x] \Longrightarrow \mathtt{c}}{\mathtt{mn} \Longrightarrow \mathtt{c}}$$

Recursive Functions

$$\frac{\mathtt{n} \Longrightarrow \mathtt{c}' \quad \mathtt{m}[\lambda x: \sigma.\mathtt{R}_{\sigma,\sigma'}(\mathtt{m},x)/f, \mathtt{c}'/x] \Longrightarrow \mathtt{c}}{\mathtt{R}_{\sigma,\sigma'}(\mathtt{m},\mathtt{n}) \Longrightarrow \mathtt{c}}$$

It is easy to see that the dynamic semantics is deterministic and if $\mathtt{m} \Longrightarrow \mathtt{c}$ then $\mathtt{m}$ and $\mathtt{c}$ have the same type.

6.3 Translation of QL into the FIX Logic

We shall give a translation of QL into a theory over FIX. We aim to give an interpretation of the language QL which will preserve all of its structure and properties. In fact the pure FIX logic will interpret QL; more formally, the FIX theory we consider consists simply of the FIX signature with no basic function symbols or relation symbols, together with no extralogical axioms. We shall not be too formal and simply refer to the FIX logic. The first step is to translate the static semantics of QL into suitable judgements in the FIX logic.

Interpretation of the Static Semantics

For each expression in context, $x_i{:}\,\sigma_i \vdash \mathtt{m}{:}\,\sigma$, we give a a term in context of FIX, and we think of this process as a translation of QL into FIX. The static typing judgement $x_1{:}\,\sigma_1, \ldots, x_n{:}\,\sigma_n \vdash \mathtt{m}{:}\,\sigma$ is translated to

$$x_1{:}\,[\![\sigma_1]\!]^v, \ldots, x_n{:}\,[\![\sigma_n]\!]^v \vdash \vec{u}.[\![\mathtt{m}]\!]^v(\vec{x}){:}\,T[\![\sigma]\!]^v,$$

where for any term $\mathtt{m}$ in a context of n variables $\{x_1, \ldots, x_n\}$, $[\![\mathtt{m}]\!]^v$ is an expression of the abstract syntax generated from the pure FIX logic with arity TERM and for which $\mathrm{FV}([\![\mathtt{m}]\!]^v) = \{u_1, \ldots, u_n\}$. Given a closed QL expression (in context) $\vdash \mathtt{m}{:}\,\sigma$, this is of course translated to a judgement $\vdash [\![\mathtt{m}]\!]^v{:}\,T[\![\sigma]\!]^v$. Note that the superscript v on the semantic bracket $[\![-]\!]^v$ refers to the fact that we are specifying a translation of a call by value language. We shall often refer informally to a call by value translation. In order to specify the translation, we shall define expressions of the abstract syntax generated from the object level signature of FIX which have arity TERM $\to$ TERM and which we shall denote by Pred and Zero. The (representatives for these) expressions are (using η equality in the meta λ calculus) defined by

$$\begin{aligned} \mathsf{Pred}(n) &\stackrel{\mathrm{def}}{=} \mathsf{Snd}((x.\langle \mathsf{Suc}(\mathsf{Fst}(x)), \mathsf{Fst}(x)\rangle)^n(\langle \mathsf{O}, \mathsf{O}\rangle)) \\ \mathsf{Zero}(n) &\stackrel{\mathrm{def}}{=} (x.\mathsf{Inr}_{unit}(\langle\rangle))^n(\mathsf{Inl}_{unit}(\langle\rangle)) \end{aligned}$$

Note that the judgements $n{:}\,nat \vdash \mathsf{Pred}(n){:}\,nat$ and $n{:}\,nat \vdash \mathsf{Zero}(n){:}\,unit + unit$ are FIX terms in context; moreover, it is not difficult to see that Pred and Zero have the properties we would expect of them. We also make the definition

$$\mathsf{Fix}_\alpha(f) \stackrel{\mathrm{def}}{=} \mathsf{It}_\alpha(e.\mathsf{Let}\,(e, x.fx), \sigma(\omega))$$

for which it is immediate that $f{:}\,T\alpha \to T\alpha \vdash \mathsf{Fix}_\alpha(f){:}\,T\alpha$ is a FIX term in context. The translation of QL into FIX is given below:

- $[\![\mathtt{nat}]\!]^v \stackrel{\mathrm{def}}{=} nat$

- $[\![\mathtt{bool}]\!]^v \stackrel{\text{def}}{=} unit + unit$
- $[\![\sigma \to \tau]\!]^v \stackrel{\text{def}}{=} [\![\sigma]\!]^v \to T[\![\tau]\!]^v$
- $[\![x]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(u)$ where u is a meta variable.
- $[\![\mathtt{tt}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inl}_{unit}(\langle\rangle))$
- $[\![\mathtt{ff}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inr}_{unit}(\langle\rangle))$
- $[\![\mathtt{k}_n]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Suc}^n(\mathsf{O}))$
- $[\![\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{b}]\!]^v, x.\{y.[\![\mathtt{m}]\!]^v, y.[\![\mathtt{n}]\!]^v\}(x))$
- $[\![\mathtt{S}(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Suc}(x)))$
- $[\![\mathtt{P}(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Pred}(x)))$
- $[\![\mathtt{Z}(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Zero}(x)))$
- $[\![\mathtt{mn}]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{m}]\!]^v, f.\mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.fx))$
- $[\![\lambda x{:}\,\sigma.\mathtt{m}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{[\![\sigma]\!]^v}(x.[\![\mathtt{m}]\!]^v))$
- $[\![\mathtt{R}_{\sigma,\sigma'}(\mathtt{m},\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, y.\mathsf{Y}_{[\![\sigma]\!]^v,[\![\sigma']\!]^v}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v))))y)$

Interpretation of the Dynamic Semantics

Clearly the minimal requirement of an interpretation of the dynamics semantics of QL is soundness, namely that if $\mathtt{m} \Longrightarrow \mathtt{c}$ then we have $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$ where the latter equality holds in FIX. Further, it would be pleasing if whenever $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$, there is a canonical $\mathtt{c}'$ for which $\mathtt{m} \Longrightarrow \mathtt{c}'$ and $\vdash [\![\mathtt{c}']\!]^v = [\![\mathtt{c}]\!]^v$, that is to say that FIX is computationally adequate for interpreting QL. We shall soon see that this is indeed the case, and in order to do this we shall need a little additional notation. For canonical closed terms $\mathtt{c}$ of QL, note that the interpretation takes the form $[\![\mathtt{c}]\!]^v \equiv \mathsf{Val}(\lceil \mathtt{c} \rceil)$ and we shall take this as an informal definition of $\lceil \mathtt{c} \rceil$. We translate the dynamic semantics of QL into judgements in FIX simply by taking each instance of the evaluation relation $\mathtt{m} \Longrightarrow \mathtt{c}$ to the judgement $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$.

6.4 Adequacy Results for QL

Static Adequacy for QL

Proposition 6.4.1 ["QL Static Adequacy"] The interpretation of the static semantics of QL in FIX is adequate, in the sense that $x_i{:}\,\sigma_i \vdash \mathtt{m}{:}\,\sigma$ is a well formed QL

expression in context iff

$$x_i\colon [\![\sigma_i]\!]^v \vdash \vec{u}.[\![\mathtt{m}]\!]^v(\vec{x})\colon T[\![\sigma]\!]^v$$

is derivable in FIX.

Proof Both directions proceed by structural induction. We give one example, for the backwards direction.
(*Case* $\mathtt{m}$ *is* $\mathtt{R}(\mathtt{m},\mathtt{n})$): From the definition of $[\![\mathtt{R}(\mathtt{m},\mathtt{n})]\!]^v$, the FIX logic rules and the induction hypothesis, we have

$$x_i\colon \sigma_i, f\colon \sigma'\to\sigma, x\colon \sigma' \vdash \mathtt{m}\colon \sigma \text{ and } x_i\colon \sigma_i \vdash \mathtt{n}\colon \sigma',$$

from which $x_i\colon \sigma_i \vdash \mathtt{R}(\mathtt{m},\mathtt{n})\colon \sigma$ is immediate. □

Dynamic Adequacy of QL

We shall require a Lemma based on Plotkin's methods given in [Plo85]. We write $D(-)$ for the composition $[\![-]\!] \circ \vec{u}.[\![-]\!]^v(\vec{x})\colon \mathrm{QL} \to \mathrm{FIX} \to \omega\mathcal{C}po$ where $[\![-]\!]$ is the standard domain theoretic semantics of FIX. We define a relation $\lhd_\sigma$ between elements $d \in D(\sigma)$ and canonical forms $\vdash \mathtt{c}\colon\sigma$ by induction on the structure of σ. In the definition which follows, $\trianglelefteq_\sigma$ is the relation between elements $e \in D(\sigma)_\perp$ and closed QL terms $\vdash \mathtt{m}\colon\sigma$ defined in terms of $\lhd_\sigma$ by asking that $e \trianglelefteq_\sigma \mathtt{m}$ iff $\forall d \in D(\sigma).e = [d] \supset \exists \mathtt{c}.\mathtt{m} \Longrightarrow \mathtt{c}$ & $d \lhd_\sigma \mathtt{c}$. We define:

- $i(*) \lhd_{\mathtt{bool}} \mathtt{tt}$ and $j(*) \lhd_{\mathtt{bool}} \mathtt{ff}$ where $i, j\colon 1 \to 1+1$ are coproduct insertions.
- $n \lhd_{\mathtt{nat}} \mathtt{k}_n$ where $n \in \mathbb{N}$.
- $f \lhd_{\sigma\to\tau} \lambda x\colon\sigma.\mathtt{m}$ iff $\forall d \in D(\sigma)\forall \mathtt{c}\colon\sigma.d \lhd_\sigma \mathtt{c} \supset f(d) \trianglelefteq_\tau (\lambda x\colon\sigma.\mathtt{m})\mathtt{c}$.

With this, we have

Lemma 6.4.2 Let $x_1\colon\sigma_1,\ldots,x_n\colon\sigma_n \vdash \mathtt{m}\colon\sigma$ be a QL term in context and suppose that for $i = 1,\ldots,n$ we have $d_i \in D(\sigma_i)$, $\vdash c_i\colon\sigma_i$ and $d_i \lhd_{\sigma_i} \mathtt{c}_i$. Then the continuous function $D(\Gamma \vdash \mathtt{m})\colon D(\sigma_1)\times\ldots\times D(\sigma_n) \to D(\sigma)_\perp$ satisfies $D(\Gamma \vdash \mathtt{m})(\vec{d}) \trianglelefteq_\sigma \mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$.

Proof The proof proceeds by induction on the structure of $\mathtt{m}$. We illustrate the proof with two cases
(*Case* $\mathtt{m}$ *is* $\lambda x\colon\sigma.\mathtt{m}$): Suppose that the conditions of the lemma are satisfied. Then we need to show that $D(\Gamma \vdash \lambda x\colon\sigma.\mathtt{m})(\vec{d}) \trianglelefteq_{\sigma\to\tau} \lambda x\colon\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$. Using the definition of $D(-)$ we can show that $D(\Gamma \vdash \lambda x\colon\sigma.\mathtt{m}) = \iota \circ cur(D(\Gamma, x\colon\sigma \vdash \mathtt{m}))$ where

$$\iota\colon D(\sigma)\to D(\tau)_\perp \to (D(\sigma)\to D(\tau)_\perp)_\perp.$$

Hence $D(\Gamma \vdash \lambda x\colon\sigma.\mathtt{m})(\vec{d}) = [cur(D(\Gamma, x\colon\sigma \vdash \mathtt{m}))(\vec{d})]$. By definition of the $\trianglelefteq$ relation we show that $cur(D(\Gamma, x\colon\sigma \vdash \mathtt{m}))(\vec{d}) \lhd_{\sigma\to\tau} \lambda x\colon\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$; thus if $d \lhd_\sigma \mathtt{c}$ it remains

to show $cur(D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m}))(\vec{d})(d) \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}])\mathtt{c}$. By the induction hypothesis, $D(\Gamma, x{:}\sigma \vdash \mathtt{m})(\vec{d}, d) \trianglelefteq_\tau \mathtt{m}[\vec{\mathtt{c}}/\vec{x}, \mathtt{c}/x]$ and so $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}, \mathtt{c}/x] \Longrightarrow \mathtt{c}'$ for some $\mathtt{c}'$ provided that $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m})(\vec{d})(d)$ is not $\bot$. But then $(\lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}])\mathtt{c} \Longrightarrow \mathtt{c}'$ and we are done.

(*Case* $\mathtt{m}$ *is* $\mathtt{mn}$): We need to show that $D(\Gamma \vdash \mathtt{mn})(\vec{d}) \trianglelefteq_\tau \mathtt{m}[\vec{\mathtt{c}}/\vec{x}]\mathtt{n}[\vec{\mathtt{c}}/\vec{x}]$ where, say, $\Gamma \vdash \mathtt{m}{:}\sigma{\to}\tau$ and $\Gamma \vdash \mathtt{n}{:}\sigma$. Suppose that $D(\Gamma \vdash \mathtt{mn})(\vec{d})$ is not $\bot$. One can check that neither are $D(\Gamma \vdash \mathtt{m})(\vec{d})$ or $D(\Gamma \vdash \mathtt{n})(\vec{d})$; let us write $[f]$ and $[d]$ for these. By the induction hypothesis we have $[f] \trianglelefteq_{\sigma\to\tau} \mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$ and $[d] \trianglelefteq_\sigma \mathtt{n}[\vec{\mathtt{c}}/\vec{x}]$. Hence $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}] \Longrightarrow \lambda x{:}\sigma.\mathtt{m}'$ and $\mathtt{n}[\vec{\mathtt{c}}/\vec{x}] \Longrightarrow \mathtt{c}$. This leads to $f(d) \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m}')\mathtt{c}$ and from the original supposition there is some $\mathtt{c}'$ for which $(\lambda x{:}\sigma.\mathtt{m}')\mathtt{c} \Longrightarrow \mathtt{c}'$. Thus we may deduce $\mathtt{m}'[\mathtt{c}/x] \Longrightarrow \mathtt{c}'$ and conclude $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]\mathtt{n}[\vec{\mathtt{c}}/\vec{x}] \Longrightarrow \mathtt{c}'$. □

We shall also need the following Lemma:

Lemma 6.4.3 With the call by value interpretation of QL, and $x{:}\sigma \vdash \mathtt{m}{:}\tau$, $\vdash \mathtt{c}{:}\sigma$ QL terms in context with $\mathtt{c}$ canonical, we have

$$\vdash [\![\mathtt{m}[\mathtt{c}/x]]\!]^v = [\![\mathtt{m}]\!]^v[\lceil \mathtt{c} \rceil/u],$$

where $[\![x]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(u)$ and $[\lceil \mathtt{c} \rceil/u]$ means substitution in the meta λ calculus.

Proof N.B. Recall Section 0.3 of Chapter 0. The proof is a trivial structural induction on $\mathtt{m}$. We illustrate with one example.

(*Case* $\mathtt{m}$ *is* $\mathtt{R}(\mathtt{m},\mathtt{n})$):

$$\begin{aligned} \vdash [\![\mathtt{R}(\mathtt{m},\mathtt{n})\ [\mathtt{c}/y]]\!]^v &= \mathsf{Let}\,([\![\mathtt{n}[\mathtt{c}/y]]\!]^v, u.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}[\mathtt{c}/y]]\!]^v))))u) \\ \text{which by induction is} &= \mathsf{Let}\,([\![\mathtt{n}]\!]^v, u.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v))))u)\ [\lceil \mathtt{c} \rceil/y] \\ &= [\![\mathtt{R}(\mathtt{m},\mathtt{n})]\!]^v[\lceil \mathtt{c} \rceil/y]. \end{aligned}$$

□

Theorem 6.4.4 ["QL Dynamic Adequacy"] The interpretation of QL in the FIX logic is *computationally adequate*; more precisely, given closed QL terms $\mathtt{m}$ and $\mathtt{c}$ for which $\mathtt{c}$ is canonical, then $\mathtt{m} \Longrightarrow \mathtt{c}$ implies $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$, and $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$ implies there is some canonical $\mathtt{c}'$ for which $\mathtt{m} \Longrightarrow \mathtt{c}'$.

Proof The proof in the forwards direction proceeds by induction on the derivation of $\mathtt{m} \Longrightarrow \mathtt{c}$; we give details for the cases of application and recursive function terms.

(*Case* $\mathtt{mn} \Longrightarrow \mathtt{c}$): Using minimality of $\Longrightarrow$ and the induction hypothesis, we obtain

$$\begin{aligned} \vdash [\![\mathtt{m}]\!]^v &= \mathsf{Val}(\lambda(x.[\![\mathtt{m}']\!]^v)) \\ \vdash [\![\mathtt{n}]\!]^v &= \mathsf{Val}(\lceil \mathtt{c}' \rceil) \\ \vdash [\![\mathtt{m}'[\mathtt{c}'/x]]\!]^v &= \mathsf{Val}(\lceil \mathtt{c} \rceil). \end{aligned}$$

Thus we have

$$\begin{aligned} \vdash [\![\mathtt{mn}]\!]^v &= \mathsf{Let}([\![\mathtt{m}]\!]^v, f.\mathsf{Let}([\![\mathtt{n}]\!]^v, x.fx)) \\ &= \lambda(x.[\![\mathtt{m}']\!]^v)\lceil \mathtt{c}' \rceil \\ &= [\![\mathtt{m}']\!]^v[\lceil \mathtt{c}' \rceil / x] \\ \text{which by Lemma 6.4.3} &= [\![\mathtt{c}]\!]^v, \end{aligned}$$

as required.

(*Case* $\mathtt{R}(\mathtt{m},\mathtt{n}) \Longrightarrow \mathtt{c}$): Using minimality of $\Longrightarrow$ and the induction hypothesis, we obtain

$$\begin{aligned} \vdash [\![\mathtt{n}]\!]^v &= \mathsf{Val}(\lceil \mathtt{c}' \rceil) \\ \vdash [\![\mathtt{m}[\lambda x{:}\sigma.\mathtt{R}(\mathtt{m},x)/f, \mathtt{c}'/x]]\!]^v &= \mathsf{Val}(\lceil \mathtt{c} \rceil). \end{aligned}$$

Let us put $M \stackrel{\text{def}}{=} \lambda(f.\lambda(x.[\![\mathtt{m}]\!]^v))$ and note that

$$\begin{aligned} \vdash [\![\lambda x{:}\sigma.\mathtt{R}(\mathtt{m},x)]\!]^v &= \mathsf{Val}(\lambda(x.\mathsf{Y}(M)x)) \\ &= \mathsf{Val}(\mathsf{Y}(M)). \end{aligned}$$

Thus we have

$$\begin{aligned} \vdash [\![\mathtt{R}(\mathtt{m},\mathtt{n})]\!]^v &= \mathsf{Let}([\![\mathtt{n}]\!]^v, y.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v)))y) \\ &= \lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v)))\ \mathsf{Y}(M)\ \lceil \mathtt{c}' \rceil \\ &= \lambda(x.[\![\mathtt{m}]\!]^v)[\mathsf{Y}(M)/f]\ \lceil \mathtt{c}' \rceil \\ &= \lambda(x.[\![\mathtt{m}]\!]^v[\mathsf{Y}(M)/f])\ \lceil \mathtt{c}' \rceil \\ &= [\![\mathtt{m}]\!]^v[\mathsf{Y}(M)/f][\lceil \mathtt{c}' \rceil / x] \\ &= [\![\mathtt{m}]\!]^v[\lceil \lambda x{:}\sigma.\mathtt{R}(\mathtt{m},x) \rceil / f, \lceil \mathtt{c}' \rceil / x] \\ \text{which by Lemma 6.4.3} &= [\![\mathtt{c}]\!]^v, \end{aligned}$$

and so we are done.

For the converse direction, suppose that $\vdash [\![\mathtt{m}]\!]^v =_\sigma [\![\mathtt{c}]\!]^v$. We have $[\![\mathtt{m}]\!]^v = \mathsf{Val}(\lceil \mathtt{c} \rceil)$ and hence it is the case that $D(\vdash \mathtt{m})(*)$ is not $\bot$, say $[d]$. Appeal to Lemma 6.4.2 to deduce that $[d] \trianglelefteq_\sigma \mathtt{m}$ and hence there is some canonical $\mathtt{c}'$ for which $\mathtt{m} \Longrightarrow \mathtt{c}'$ by the definition of $\trianglelefteq$. □

6.5 A Further PCF style language, HPCF

We define the language HPCF by specifying the basic syntax of types and raw expressions; this syntax will then be given a static and dynamic semantics.

The Types and Expressions of HPCF

The types of HPCF are given by the grammar:

$$\sigma ::= \mathtt{bool} \mid \mathtt{nat} \mid \sigma \to \sigma$$

The (raw) expressions of HPCF are given by the grammar:

$$
\begin{array}{rcll}
\mathtt{m} & ::= & x & \text{variables} \\
& | & \mathtt{tt} & \text{truth} \\
& | & \mathtt{ff} & \text{falsity} \\
& | & \mathtt{k}_n & \text{natural numbers} \\
& | & \mathtt{C}_\sigma & \text{conditional} \\
& | & \mathtt{S} & \text{successor} \\
& | & \mathtt{P} & \text{predecessor} \\
& | & \mathtt{Z} & \text{zero test} \\
& | & \mathtt{Y}_\sigma & \text{fixpoints} \\
& | & \mathtt{mn} & \text{application} \\
& | & \lambda x{:}\sigma.\mathtt{m} & \text{function definition}
\end{array}
$$

The Static Semantics of HPCF

Variables

$$\frac{}{\Gamma, x{:}\sigma, \Gamma' \vdash x{:}\sigma}$$

Constants

$$\frac{}{\Gamma \vdash \mathtt{tt}{:}\mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{ff}{:}\mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{k}_n{:}\mathtt{nat}}$$

Conditional

$$\frac{}{\Gamma \vdash \mathtt{C}_\sigma{:}\mathtt{bool}\to(\sigma\to(\sigma\to\sigma))}$$

Arithmetic

$$\frac{}{\Gamma \vdash \mathtt{S}{:}\mathtt{nat}\to\mathtt{nat}} \qquad \frac{}{\Gamma \vdash \mathtt{P}{:}\mathtt{nat}\to\mathtt{nat}} \qquad \frac{}{\Gamma \vdash \mathtt{Z}{:}\mathtt{nat}\to\mathtt{bool}}$$

Fixpoints

$$\frac{}{\Gamma \vdash \mathtt{Y}_\sigma{:}(\sigma\to\sigma)\to\sigma}$$

Functions

$$\frac{\Gamma \vdash \mathtt{m}{:}\sigma\to\tau \quad \Gamma \vdash \mathtt{n}{:}\sigma}{\Gamma \vdash \mathtt{mn}{:}\tau} \qquad \frac{\Gamma, x{:}\sigma \vdash \mathtt{m}{:}\tau}{\Gamma \vdash \lambda x{:}\sigma.\mathtt{m}{:}\sigma\to\tau}$$

The Dynamic Semantics of HPCF

The *canonical* HPCF expressions consist of the subset of closed expressions given by the grammar:

$$\mathtt{c} ::= \mathtt{tt} \mid \mathtt{ff} \mid \mathtt{C}_\sigma \mid \mathtt{k}_n \mid \mathtt{S} \mid \mathtt{P} \mid \mathtt{Z} \mid \mathtt{Y}_\sigma \mid \lambda x{:}\sigma.\mathtt{m} \mid \mathtt{C}_\sigma\mathtt{b} \mid \mathtt{C}_\sigma\mathtt{bm}$$

We now give the syntax of HPCF a call by name dynamic semantics. Apart from conditionals at higher types, HPCF is in every respect identical to Plotkin's language PCF. The dynamic semantics will be presented using an evaluation relation just as for QL:

Canonical Forms

$$\frac{\mathtt{c}\ \text{canonical}}{\mathtt{c} \Longrightarrow \mathtt{c}}$$

Conditionals

$$\frac{\mathtt{m} \Longrightarrow \mathtt{C}_\sigma}{\mathtt{mb} \Longrightarrow \mathtt{C}_\sigma\mathtt{b}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{C}_\sigma\mathtt{b}}{\mathtt{mn} \Longrightarrow \mathtt{C}_\sigma\mathtt{bn}}$$

$$\frac{\mathtt{m} \Longrightarrow \mathtt{C}_\sigma\mathtt{bm}' \quad \mathtt{b} \Longrightarrow \mathtt{tt} \quad \mathtt{m}' \Longrightarrow \mathtt{c}}{\mathtt{mn} \Longrightarrow \mathtt{c}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{C}_\sigma\mathtt{bm}' \quad \mathtt{b} \Longrightarrow \mathtt{ff} \quad \mathtt{n} \Longrightarrow \mathtt{c}}{\mathtt{mn} \Longrightarrow \mathtt{c}}$$

$$\frac{\mathtt{l} \Longrightarrow \mathtt{C}_\sigma \quad \mathtt{b} \Longrightarrow \mathtt{tt} \quad \mathtt{m} \Longrightarrow \mathtt{c}}{\mathtt{lbmn} \Longrightarrow \mathtt{c}} \qquad \frac{\mathtt{l} \Longrightarrow \mathtt{C}_\sigma \quad \mathtt{b} \Longrightarrow \mathtt{ff} \quad \mathtt{n} \Longrightarrow \mathtt{c}}{\mathtt{lbmn} \Longrightarrow \mathtt{c}}$$

Arithmetic

$$\frac{\mathtt{m} \Longrightarrow \mathtt{S} \quad \mathtt{n} \Longrightarrow \mathtt{k}_n}{\mathtt{mn} \Longrightarrow \mathtt{k}_{n+1}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{P} \quad \mathtt{n} \Longrightarrow \mathtt{k}_{n+1}}{\mathtt{mn} \Longrightarrow \mathtt{k}_n} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{P} \quad \mathtt{n} \Longrightarrow \mathtt{k}_0}{\mathtt{mn} \Longrightarrow \mathtt{k}_0}$$

$$\frac{\mathtt{m} \Longrightarrow \mathtt{Z} \quad \mathtt{n} \Longrightarrow \mathtt{k}_0}{\mathtt{mn} \Longrightarrow \mathtt{tt}} \qquad \frac{\mathtt{m} \Longrightarrow \mathtt{Z} \quad \mathtt{n} \Longrightarrow \mathtt{k}_{n+1}}{\mathtt{mn} \Longrightarrow \mathtt{ff}}$$

Fixpoints

$$\frac{\mathtt{m} \Longrightarrow \mathtt{Y}_\sigma \quad \mathtt{nY}_\sigma\mathtt{n} \Longrightarrow \mathtt{c}}{\mathtt{mn} \Longrightarrow \mathtt{c}}$$

Functions

$$\frac{\mathtt{m} \Longrightarrow \lambda x{:}\sigma.\mathtt{m}' \quad \mathtt{m}'[\mathtt{n}/x] \Longrightarrow \mathtt{c}}{\mathtt{mn} \Longrightarrow \mathtt{c}}$$

Remark 6.5.1 Plotkin originally specified the operational semantics of PCF via a single step reduction relation of the form $\mathtt{m} \longmapsto \mathtt{n}$ where $\mathtt{m}$ and $\mathtt{n}$ are closed terms. Clearly HPCF could be given an operational semantics in the same way: for details of the original specification of Plotkin's PCF in this style of semantics see [Plo77]. We omit the details, but remark that presenting HPCF in this style would lead to:

Proposition 6.5.2 Let $\mathtt{m}$ and $\mathtt{c}$ be closed HPCF terms with $\mathtt{c}$ canonical. Then $\mathtt{m} \Longrightarrow \mathtt{c}$ iff $\mathtt{m} \longmapsto^* \mathtt{c}$, where $\longmapsto^*$ is the reflexive transitive closure of $\longmapsto$. □

This can be proved succinctly through:

Lemma 6.5.3 If $\mathtt{m} \longmapsto \mathtt{n}$ then for all closed canonical $\mathtt{c}$, we have $\mathtt{n} \Longrightarrow \mathtt{c}$ implies $\mathtt{m} \Longrightarrow \mathtt{c}$. □

6.6 Translation of HPCF into the FIX Logic

Interpretation of the Static Semantics

For each expression in context, $x_i{:}\sigma_i \vdash \mathtt{m}{:}\sigma$ of HPCF, we give a translation into a term in context of FIX. The static typing judgement $x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n \vdash \mathtt{m}{:}\sigma$ is translated to

$$x_1{:}T[\![\sigma_1]\!]^n, \ldots, x_n{:}T[\![\sigma_n]\!]^n \vdash \vec{u}.[\![\mathtt{m}]\!]^n(\vec{x}){:}T[\![\sigma]\!]^n.$$

The translation of HPCF into FIX is given below:

- $[\![\mathtt{nat}]\!]^n \stackrel{\text{def}}{=} nat$
- $[\![\mathtt{bool}]\!]^n \stackrel{\text{def}}{=} unit + unit$
- $[\![\sigma{\to}\tau]\!]^n \stackrel{\text{def}}{=} T[\![\sigma]\!]^n{\to}T[\![\tau]\!]^n$
- $[\![x]\!]^n \stackrel{\text{def}}{=} u$ where u is a meta variable.
- $[\![\mathtt{tt}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inl}_{unit}(\langle\rangle))$
- $[\![\mathtt{ff}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inr}_{unit}(\langle\rangle))$
- $[\![\mathtt{k}_n]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Suc}^n(\mathsf{O}))$
- $[\![\mathtt{C}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{bool}]\!]^n}(b.\mathsf{Val}(\lambda_{T[\![\sigma]\!]^n}(z.(\mathsf{Val}(\lambda_{T[\![\sigma]\!]^n}(z'.\mathsf{Let}\,(b, x.\{y.z, y.z'\}(x)\ldots)$
- $[\![\mathtt{S}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Suc}(x)))))$
- $[\![\mathtt{P}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Pred}(x)))))$
- $[\![\mathtt{Z}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Zero}(x)))))$
- $[\![\mathtt{Y}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T(T[\![\sigma]\!]^n{\to}T[\![\sigma]\!]^n)}(y.\mathsf{Fix}_{[\![\sigma]\!]^n}(\lambda_{T[\![\sigma]\!]^n}(x.\mathsf{Let}\,(y, f.fx)))))$
- $[\![\mathtt{mn}]\!]^n \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{m}]\!]^n, f.f[\![\mathtt{n}]\!]^n))$
- $[\![\lambda x{:}\sigma.\mathtt{m}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{[\![T\sigma]\!]^n}(x.[\![\mathtt{m}]\!]^n))$

Note that this interpretation is one of a number of possibilities. Of course, for most of the syntax of HPCF there will only be one sensible translation. However, in the case of the fixpoint constants $\mathtt{Y}_\sigma$, there are two reasonable translations and (as we shall see) they have quite different properties. This said, the important requirement of any translation is that it preserves the structure and properties of the original language. In Section 6.8 we shall give an alternative translation of $\mathtt{Y}_\sigma$ and investigate its properties.

Interpretation of the Dynamic Semantics

This is the same as for QL: see Page 103

6.7 Adequacy Results for HPCF

Static Adequacy for HPCF

We prove the following Proposition, establishing that the translation of the static semantics of HPCF is, in a sense to be made precise, information preserving.

Proposition 6.7.1 ["HPCF Static Adequacy"] The interpretation of the static semantics of HPCF in FIX is adequate, in the sense that $x_i{:}\,\sigma_i \vdash \mathtt{m}{:}\,\sigma$ is a well formed HPCF expression in context iff

$$x_i{:}\,T[\![\sigma_i]\!]^n \vdash \vec{u}.[\![\mathtt{m}]\!]^n(\vec{x}){:}\,T[\![\sigma]\!]^n$$

is derivable in FIX.

Proof The forwards direction is an induction on the structure of the term $\mathtt{m}$; we illustrate one case.
(*Case* $\mathtt{m}$ *is* $\mathtt{Y}_\sigma\mathtt{m}$): By induction and the definition of the translation, we have

$$x_i{:}\,T[\![\sigma_i]\!]^n \vdash \vec{u}.[\![\mathtt{m}]\!]^n(\vec{x}){:}\,T(T[\![\sigma]\!]^n \to T[\![\sigma]\!]^n)$$

and thus (using the fact that the raw terms (represented by) $\vec{u}.[\![\mathtt{m}]\!]^n(\vec{x})$ and $[\![\mathtt{m}]\!]^n[\vec{x}/\vec{u}]$ are the same)

$$x_i{:}\,T[\![\sigma_i]\!]^n \vdash \mathsf{Let}\,([\![\mathtt{m}]\!]^n, x.\mathsf{Fix}(x)){:}\,T[\![\sigma]\!]^n.$$

From the definition of $[\![\mathtt{Y}_\sigma\mathtt{m}]\!]^n$ we are done. Clearly the reverse direction is equally easy. □

Dynamic Adequacy for HPCF

We shall need the following Lemma:

Lemma 6.7.2 With the call by name interpretation of HPCF, and $x{:}\sigma \vdash \mathtt{m}{:}\tau$, $\vdash \mathtt{n}{:}\sigma$ HPCF terms in context, we have

$$\vdash [\![\mathtt{m}[\mathtt{n}/x]]\!]^n = [\![\mathtt{m}]\!]^n[[\![\mathtt{n}]\!]^n/u],$$

where $[\![x]\!]^v \stackrel{\text{def}}{=} u$.

Proof Trivial induction. □

Theorem 6.7.3 ["HPCF Dynamic Adequacy"] The translation of HPCF into the FIX logic is *computationally adequate*; more precisely, given closed HPCF terms $\mathtt{m}$ and $\mathtt{c}$ where $\mathtt{c}$ is canonical, then $\mathtt{m} \Longrightarrow \mathtt{c}$ implies $\vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{c}]\!]^n$ and if $\vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{c}]\!]^n$ then there is a canonical $\mathtt{c}'$ for which $\mathtt{m} \Longrightarrow \mathtt{c}'$.

Proof The "only if" uses rule induction on the derivation of the evaluation relation. We shall just give two cases, namely for application and fixpoint terms.

(*Case* Functions): Using minimality of $\Longrightarrow$ and the induction hypothesis, we obtain

$$\begin{array}{l} \vdash [\![\mathtt{m}]\!]^n = \mathsf{Val}(\lambda(x.[\![\mathtt{m}']\!]^n)) \\ \vdash [\![\mathtt{m}'[\mathtt{n}/x]]\!]^n = [\![\mathtt{c}]\!]^n. \end{array}$$

Hence we get

$$\begin{array}{rrcl} & \vdash [\![\mathtt{mn}]\!]^n & \stackrel{\text{def}}{=} & \mathsf{Let}\,([\![\mathtt{m}]\!]^n, f.f[\![\mathtt{n}]\!]^n) \\ & & = & \lambda(x.[\![\mathtt{m}']\!]^n)[\![\mathtt{n}]\!]^n \\ & & = & [\![\mathtt{m}']\!]^n[[\![\mathtt{n}]\!]^n/x] \\ \text{which via Lemma 6.7.2} & & = & [\![\mathtt{c}]\!]^n \end{array}$$

as required.

(*Case* Fixpoints): Using minimality of $\Longrightarrow$, the induction hypothesis and the translation of application terms, we have

$$\begin{array}{l} \vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{Y}_\sigma]\!]^n \\ \vdash \mathsf{Let}\,([\![\mathtt{n}]\!]^n, g.g[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n) = [\![\mathtt{c}]\!]^n. \end{array}$$

Hence we get

$$\begin{array}{rcl} [\![\mathtt{mn}]\!]^n & = & \mathsf{Let}\,([\![\mathtt{Y}_\sigma]\!]^n, f.f[\![\mathtt{n}]\!]^n) \\ & = & \mathsf{Fix}(\lambda(x.\mathsf{Let}\,([\![\mathtt{n}]\!]^n, f.fx))) \\ & = & \mathsf{Let}\,([\![\mathtt{n}]\!]^n, f.f[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n) \\ & = & [\![\mathtt{c}]\!]^n, \end{array}$$

which is what we had to prove.

To prove the converse direction we could use a method similar to the one used in the proof of QL Dynamic Adequacy. The details are omitted. □

6.8 An Alternative Translation of Fixpoints

All of the results of Sections 6.6 and 6.7 remain true for a slightly different translation of the fixpoint constants $\mathtt{Y}_\sigma$. However, the proof of computational adequacy of the translation is not so straightforward as before. We present a proof which uses the Existence Property of the FIX logic which was stated on Page 73.

The translation of the fixpoint constants $\mathtt{Y}_\sigma$ now takes the form

$$[\![\mathtt{Y}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T([\![\sigma]\!]^n \to T[\![\sigma]\!]^n)}(y.\mathsf{Let}\,(y, x.\mathsf{Fix}_{[\![\sigma]\!]^n}(x)))).$$

In order to prove a computational adequacy result which uses this new translation, we shall need

Lemma 6.8.1 Suppose that

$$\begin{array}{rcl} \Gamma & \vdash & E{:}T\alpha \\ \Gamma, x{:}\alpha & \vdash & F(x){:}T\beta \\ \Gamma, y{:}\beta & \vdash & \Phi(y)\ \text{prop} \end{array}$$

are well formed judgements in FIX. Then we have

$$\frac{\Gamma, \Lambda \vdash \Diamond(\mathsf{Let}\,(E, F), \Phi)}{\Gamma, \Lambda \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}$$

ProofThe labelling of steps in the prooftrees is informal and for guidance only. We have

$$\dfrac{\dfrac{\dfrac{(\Diamond\mathtt{i})}{\Gamma, x{:}\alpha, y{:}\beta, \Lambda, \Phi(y), F(x) = \mathsf{Val}(y) \vdash \Diamond(F(x), \Phi)}}{\Gamma, y{:}\beta, \Lambda, \Diamond(E, x.\Phi(y)\ \&\ F(x) = \mathsf{Val}(y)) \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}\,(*)}{\Gamma, y{:}\beta, \Lambda, \Phi(y)\ \&\ \Diamond(E, x.F(x) = \mathsf{Val}(y)) \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}\,(\mathtt{fr})$$

and

$$\frac{(\mathtt{mod})(\ \&\ \mathtt{ad})}{\Gamma, y{:}\beta, \Lambda, \mathsf{Let}\,(E, F) = \mathsf{Val}(y), \Phi(y) \vdash \Phi(y)\ \&\ \Diamond(E, x.F(x) = \mathsf{Val}(y))}$$

where the step $(*)$ follows from Lemma 4.5.2 and rule $(\mathtt{fr})$ is proved Proposition 4.5.3. Applying $(\mathtt{cut})$ to the above conclusions we have

$$\Gamma, y{:}\beta, \Lambda, \mathsf{Let}\,(E, F) = \mathsf{Val}(y), \Phi(y) \vdash \Diamond(E, x.\Diamond(F(x), \Phi)).$$

Using this together with the hypothesis $\Gamma, \Lambda, \vdash \Diamond(\mathsf{Let}\,(E, F), \Phi)$ and $(\Diamond\mathtt{e})$ we are done. □

Now we can prove computational adequacy:

Theorem 6.8.2 Theorem 6.7.3 remains true if we replace the translation of the constants $\mathtt{Y}_\sigma$ given on Page 109 with that given on Page 112.

Proof Clearly the change to the original proof will only involve the fixpoint constants. Indeed, for the "only if" direction:

(*Case* Fixpoints): Applying minimality of $\Longrightarrow$, the induction hypothesis, and the translation of application terms, we have

$$\begin{array}{l} \vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{Y}_\sigma]\!]^n \\ \vdash \mathsf{Let}\,([\![\mathtt{n}]\!]^n, f.f[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n) = [\![\mathtt{c}]\!]^n, \end{array}$$

and thus

$$\vdash \Diamond(\mathsf{Let}\,([\![\mathtt{n}]\!]^n, f.f[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n), x.x = \lceil \mathtt{c} \rceil).$$

Applying Lemma 6.8.1 we obtain

$$\vdash \Diamond([\![\mathtt{n}]\!]^n, y.\Diamond(y[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n, x.x = \lceil \mathtt{c} \rceil)).$$

Appealing to the Existence Property (Theorem 4.5.4), there is a closed term N for which $\vdash [\![\mathtt{n}]\!]^n = \mathsf{Val}(N)$ and $\vdash \Diamond(N[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n, x.x = \lceil \mathtt{c} \rceil)$, that is $\vdash N[\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n = [\![\mathtt{c}]\!]^n$. Via the definition of $[\![\mathtt{Y}_\sigma]\!]^n$ we see that $\vdash [\![\mathtt{Y}_\sigma\mathtt{n}]\!]^n = \mathsf{Fix}(N)$, yielding

$$\begin{array}{rcl} \vdash [\![\mathtt{mn}]\!]^n & = & \mathsf{Let}\,([\![\mathtt{Y}_\sigma]\!]^n, f.f[\![\mathtt{n}]\!]^n) \\ & = & \mathsf{Let}\,([\![\mathtt{n}]\!]^n, x.\mathsf{Fix}(x)) \\ & = & N\mathsf{Fix}(N) \\ & = & [\![\mathtt{c}]\!]^n. \end{array}$$

as required. The details for the converse direction are omitted; the proof uses a technique similar to that adopted in proving QL Dynamic Adequacy. □

Chapter 7
Representations of Scott Predomains

7.1 Scott Domains and Information Systems

It is well known that the class of Scott domains together with Scott continuous functions form a category which is equivalent to the category of information systems together with approximable maps. Note that here the Scott continuous functions are those set functions which preserve filtered colimits (i.e. directed suprema). For details see both [Sco82] and [WL83]. Of course, it is by definition that a Scott domain has a least element. We now extend Scott's results to structures which are just like Scott domains but which do not necessarily possess a least element; we shall call these Scott predomains. The literature describes many different kinds of domain and a number of the definitions are non-standard. For this reason we elaborate on precisely what we mean by a Scott predomain.

7.2 Scott Predomains and Preinformation Systems

The Category of Scott Predomains

Definition 7.2.1 If P is any poset then a subset S is *bounded* iff S is non-empty and we have $\exists p \in P.S \leq p$. We write $Bd(S)$ for this. We say that P is *bounded cocomplete* if every bounded subset has a supremum.

An ω *cocomplete partial order* is a poset which possesses suprema (colimits) of ω diagrams. We refer to these as ωcpo's and often call ω diagrams ω *chains.*

A *directed cocomplete partial order* D is a poset which has suprema of directed diagrams (recall that a *directed diagram* is a functor $f: F \to D$ where F is a poset which is a filtered category). We refer to these as dcpo's. The image of such a directed diagram will be referred to as a *directed subset* of D. If S is a directed subset of D we shall write $\bigvee^{\partial} S$ for its supremum; we shall suppose that part of the force of this notation is that S is directed. For other subsets we write $\bigvee S$ for the supremum. Finally, note that any dcpo is an ωcpo.

An element $d \in D$ is *finite* iff $d \leq \bigvee^{\partial} S$ implies $\exists s \in S.d \leq s$ for all directed S. The set of finite elements of D will be written as D°. A dcpo D is *algebraic* if for every d in D we have $d = \bigvee^{\partial}\{e \mid e \in D^{\circ} \;\&\; e \leq d\}$; note that by definition, for any element d of D there is a finite element below d.

A *Scott predomain* is a bounded cocomplete algebraic dcpo. We shall say that a set function between Scott predomains which preserves suprema of directed subsets is

Scott continuous. (Note that as a functor between categories, this coincides with the categorical notion of continuity as preservation of filtered colimits).

Proposition 7.2.2 Scott predomains and Scott continuous functions form a category, $\mathcal{Spd}$. □

Definition 7.2.3 Let $\mathcal{C}$ be a let category. Suppose that for all objects B and C the functor $\mathcal{C}((-) \times B, TC) : \mathcal{C}^{op} \to \mathcal{S}et$ is represented by an object $B \multimap C$. Then we shall say that $\mathcal{C}$ has *T-exponentials.*

Proposition 7.2.4 The category $\mathcal{Spd}$ is a let category with respect to lifting of domains, which has (finite products), stable finite coproducts, $\perp$-exponentials, NNO and FPO.

Proof We just sketch the details. The forgetful functor $U : \mathcal{Spd} \to \mathcal{S}et$ creates stable finite (co)products in $\mathcal{Spd}$. Given a Scott predomain D we define the lifted Scott predomain $D_\perp$ as expected. There is an obvious inclusion $i : D \to D_\perp$ and for a Scott continuous function $f : D \times D' \to D''_\perp$ there is $f^* : D \times D'_\perp \to D''_\perp$ sending $(d, \perp)$ to $\perp$. It is easy to see that this gives rise to a let category. Now we show that $\mathcal{Spd}((-) \times B, C_\perp)$ is representable, say by a Scott predomain $B \multimap C$. We define the underlying set of $B \multimap C$ to be the set of Scott continuous functions from B to $C_\perp$ which is a poset ordered pointwise. $B \multimap C$ is a dcpo: For let $F \subset B \multimap C$ be a directed subset. Then setting $(\bigvee^\partial F)(b) \stackrel{\text{def}}{=} \bigvee^\partial Fb$ yields the supremum of F. Now let F be non-empty and bounded by f. Then we have $Fb \leq fb$ for any $b \in B$; as $C_\perp$ is bounded cocomplete $\bigvee Fb$ exists in $C_\perp$ and hence we have $\bigvee F$ in $B \multimap C$. Finally we have to show that $B \multimap C$ is algebraic. Consider the set function defined by

$$[b, c](x) \stackrel{\text{def}}{=} \begin{cases} c & \text{if } b \leq x \\ \perp & \text{otherwise} \end{cases}$$

where $b \in B^\circ$, $c \in C^\circ_\perp$ and $x \in B$. Then it is easy to check that every $[b, c]$ is Scott continuous and finite; indeed all finite elements of $B \multimap C$ arise in this way and it is the case that if $f \in B \multimap C$ then $f = \bigvee^\partial \{[b, c] \mid [b, c] \leq f\}$. We omit details: the essence of the proof can be found in [Sco71]. Recall that the category $\mathcal{C}$ of dcpo's and Scott continuous functions is a ccc. Of course $\mathcal{Spd}$ is a full subcategory of $\mathcal{C}$ and thus the natural isomorphism

$$\mathcal{Spd}((-) \times B, C_\perp) \cong \mathcal{Spd}((-), B \multimap C)$$

is immediate. Finally, the expected candidates for the NNO and FPO are easily seen to work. □

Remark 7.2.5 Note that the category $\mathcal{Spd}$ is *not* cartesian closed. For more details about cartesian closure of categories of domains see [Jun88].

The Category of Preinformation Systems

Information systems provide a form of representation theorem for Scott domains. In essence, every such domain corresponds in a natural way to a set of sets which is ordered by inclusion. We shall describe a version of the original information systems from which we may derive a similar representation theorem for Scott predomains.

Definition 7.2.6 The category of *preinformation systems*, $p\mathcal{I}n\mathcal{S}$ consists of objects $A \stackrel{\text{def}}{=} (A, \downarrow, \vdash)$ triples which are either $(\emptyset, \emptyset, \emptyset)$, or else A is a nonempty set, $\downarrow$ is a nonempty set of finite subsets of A, and $\vdash$ is a subset of $\downarrow \times A$. The three coordinates of the triple are respectively known as *the tokens*, *the consistent sets* and the *the entailment relation.* Note that we shall confuse the preinformation system A with the token set A. These triples satisfy the following data:

1. $\emptyset \notin \downarrow$.
2. $X \subset^f Y\downarrow$ implies $X\downarrow$, where $\subset^f$ denotes non-empty finite subset and $X\downarrow$ means $X \in \downarrow$.
3. $a \in A$ implies $\{a\}\downarrow$.
4. $X \vdash a$ implies $X \cup \{a\}\downarrow$.
5. $X\downarrow$ & $a \in X$ implies $X \vdash a$.
6. $X \vdash Y \vdash a$ implies $X \vdash a$ where $X \vdash Y$ means $X \vdash y$ for each $y \in Y$.

Note that part of the force of the judgement $X \vdash a$ is that X *is consistent.* We refer to the objects as *preinformation systems* and $(\emptyset, \emptyset, \emptyset)$ as the *empty* preinformation system.
The morphism sets $p\mathcal{I}n\mathcal{S}(A, B)$ are empty if $B = \emptyset$ and $A \neq \emptyset$, $\{\emptyset\}$ if $A = \emptyset$ and otherwise consist of all those $r \subset \downarrow_A \times \downarrow_B$ which satisfy

1. $\forall a \in A.\exists Y\downarrow_B.\{a\}rY$.
2. XrY & XrY' implies $Xr(Y \cup Y')$.
3. $X' \vdash_A XrY \vdash_B Y'$ implies $X'rY'$.

We refer to a morphism $r\colon A \to B$ as a *preapproximable map*; the identity on A is just $\vdash_A$ and composition is the usual composition of relations.

Remark 7.2.7 This definition is clearly very similar to that of an information system as given in [Sco82] and [WL83]. In Scott's original paper, the token sets contain a distinguished element Δ which plays the role of a least element in the corresponding domain. However, if this requirement is removed, the resulting information systems still represent Scott domains as is noted in [WL83]; the (consistent) empty set plays the role of a bottom element. Thus in [WL83] there are simply

more information systems in any equivalence class which represents a particular domain than is the case in Scott's paper [Sco82]. However, if the requirement that the empty set be consistent is removed, then the resulting structures will, as we shall see, represent Scott predomains.

However, if this step is taken, the original definition of approximable map must be altered in order that the category of preinformation systems be equivalent to that of Scott predomains. Condition (1) imposes a direct "total functionality" condition on the preapproximable maps. If the definition of *approximable* map is inspected, where say $r: A \to B$ is an approximable map, then for any $\{a\} \in A$ there may not be $Y\downarrow_B$ for which $\{a\}rY$. If this is the case, then the continuous function $|r|$ between domains $|A|$ and $|B|$ corresponding to $r: A \to B$ would map a to $\perp$. Working with Scott predomains means that there may not be a least element to absorb this inherent partiality in the definition of approximable map

Proposition 7.2.8 Definition 7.2.6 does indeed yield a category $p\mathcal{I}n\mathcal{S}$.

Proof This is essentially routine. We just look at condition (1) for the composition $A \xrightarrow{r} B \xrightarrow{s} C$ between non-empty preinformation systems. Let $a \in A$. As r and s are preapproximable maps there is $Y\downarrow_B$ such that $\{a\}rY$ and $\forall y \in Y.\exists Z_y\downarrow_C.\{a\}rY \vdash_B \{y\}sZ_y$; hence $\{a\}rYsZ_y$. But Y is finite and thus $Ys\bigcup\{Z_y \mid y \in Y\}$ implying $\{a\}sr\bigcup\{Z_y \mid y \in Y\}$. The details of conditions (2) and (3) for preapproximable maps are essentially as in [Sco82]. □

7.3 Equivalence of the Categories $p\mathcal{I}n\mathcal{S}$ and $\mathcal{S}pd$

Scott Families

Our aim now is to prove that the categories $p\mathcal{I}n\mathcal{S}$ and $\mathcal{S}pd$ are equivalent. In order to do this we introduce the auxiliary notion of Scott family of sets.

Definition 7.3.1 A *Scott family* of sets is a set of non-empty sets $\mathcal{F}$ where

1. For directed $\mathcal{S} \subset \mathcal{F}$ we have $\bigcup^\partial \mathcal{S} \in \mathcal{F}$.
2. For non-empty $\mathcal{U} \subset \mathcal{F}$ we have $\bigcap\mathcal{U} \in \mathcal{F}$ whenever $\bigcap\mathcal{U}$ is non-empty.

Remark 7.3.2 We shall often refer to Scott predomains as *domains*, Scott continuous functions as *continuous functions* and preinformation systems as *presystems.*

Proposition 7.3.3 There is a functor $|-|: p\mathcal{I}n\mathcal{S} \to \mathcal{S}pd$ given by the following prescription:

On objects, $|A|$ is the empty domain when A is the empty presystem and otherwise consists of the non-empty subsets $U \subset A$ such that

(i) $X \subset^f U$ implies $X\downarrow$.

(ii) $X \subset^f U \;\&\; X \vdash a$ implies $a \in U$.

On morphisms, $|r|\colon |A| \to |B|$ is given by

$$|r|(U) \stackrel{\text{def}}{=} \{b \in B \mid \exists X \subset^f U. X r \{b\}\}.$$

We shall refer to an element U of $|A|$ as a *point* of A.

Proof The following Lemma will prove useful:

Lemma 7.3.4 Let $(A, \downarrow, \vdash)$ be a non-empty presystem, $X\downarrow$ and $Y \subset^f A$. Then

(i) $X \vdash Y$ implies $X \cup Y\downarrow$ and $Y\downarrow$.

(ii) If $X\downarrow$ then $\overline{X} \stackrel{\text{def}}{=} \{a \mid X \vdash a\}$ is a point of A.

Proof (i) is a simple induction on the cardinality of the (finite) set Y. (ii) is a consequence of (i). □

We have to show that $|A|$ is a domain; this has only to be checked when A is non-empty. First we prove that $|A|$ is a Scott family of sets and then show every such family is a domain.
By construction $|A|$ is a set of non-empty sets. We check it is a Scott family:

1. Take $\mathcal{S} \subset |A|$ directed. Certainly $\bigcup^\partial \mathcal{S}$ is a non-empty subset of A. (i) Let $X \subset^f \bigcup^\partial \mathcal{S}$. X is finite so $X \subset^f S \in \mathcal{S}$ and thus $X\downarrow$. (ii) Let $X \vdash a$; then $a \in S \subset \bigcup^\partial \mathcal{S}$.

2. Take non-empty $\mathcal{U} \subset |A|$ for which $\bigcap \mathcal{U} \neq \emptyset$. By hypothesis there is $U \in \mathcal{U}$: (i) $X \subset^f \bigcap \mathcal{U} \subset U$ implies $X\downarrow$. (ii) If also $X \vdash a$ then $a \in U$ for any such U, i.e. $a \in \bigcap \mathcal{U}$.

Obviously $|A|$ has a poset structure when ordered by inclusion; that it is a Scott family says immediately it is a dcpo.
$|A|$ is bounded cocomplete: Take $\mathcal{U} \subset |A|$ bounded, say by $U' \in |A|$ and set $\mathcal{V} \stackrel{\text{def}}{=} \{V \in |A| \mid \mathcal{U} \subset V\}$; as $U' \in \mathcal{V}$, $\mathcal{V}$ is itself non-empty. By hypothesis there is a non-empty set $U \in \mathcal{U}$; then any element $u \in U$ will be an element of every $V \in \mathcal{V}$ and hence $\bigcap \mathcal{V}$ is non-empty. As $|A|$ is a Scott family, this means $\bigcup \mathcal{U} = \bigcap \mathcal{V} \in |A|$ and this is certainly the supremum of $\mathcal{U}$.
Finally, we show that the dcpo $|A|$ is algebraic: It will be convenient to have the

Lemma 7.3.5 The finite elements of the dcpo $|A|$ are given by the collection $\{\overline{X} \mid X\downarrow\}$.

Proof (*Suppose $U \in |A|$ is finite*): Clearly $U = \bigcup^\partial \{\overline{X} \mid X \subset^f U\}$; this makes sense, for any such X is consistent in A. As U finite, $U \subset \overline{X_0}$ for some $X_0 \subset^f U$. As U is entailment closed, we have also $\overline{X}_0 \subset U$, as desired.
(*Suppose $\overline{X} \in |A|$ for $X\downarrow$*): Let $\overline{X} \subset \bigcup^\partial \mathcal{S}$. X is of finite cardinality and so $X \subset S$ for some $S \in \mathcal{S}$. By entailment closure $\overline{X} \subset S \in |A|$ showing $\overline{X}$ is finite. □

Let $U \in |A|$. Then certainly we have $U = \bigcup^{\partial}\{\overline{X} \mid X \subset^{f} U\}$ and it is easy to see using Lemma 7.3.5 that the directed set consists precisely of the finite elements below U i.e. $|A|$ is algebraic.

Thus we have shown that $|A|$ is a domain; it remains to demonstrate that $|-|$ is well defined and functorial on morphisms.

Let $r: A \to B$ be a preapproximable map. Then $|r|$ is well defined: Take $U \in |A|$ and we show $|r|(U) \in |B|$. Let $Y \subset^{f} |r|(U)$. Then for any $y \in Y$ we have $\exists X_y \subset^{f} U.X_y r\{y\}$ and it follows from the properties enjoyed by U and r that $\bigcup\{X_y \mid y \in Y\} \vdash_A X_y r\{y\} \vdash_B \{y\}$ implying $\bigcup\{X_y \mid y \in Y\} r\{y\}$. But y is arbitrary in the finite set Y giving $\bigcup\{X_y \mid y \in Y\} r Y$ and so $Y\downarrow_B$. If also $Y \vdash_B b$ then it is immediate that $\bigcup\{X_y \mid y \in Y\} r\{b\}$ and we are done.

$|r|$ is a continuous function: To see this take $\{U_i \mid i \in I\} \subset |A|$ directed. It is easy to see that $\{|r|(U_i) \mid i \in I\}$ is directed and hence $\bigcup^{\partial}\{|r|(U_i) \mid i \in I\}$ exists in $|B|$. Certainly $\bigcup^{\partial}\{|r|(U_i) \mid i \in I\} \subset |r|(\bigcup^{\partial}\{U_i \mid i \in I\})$. For the converse take $b \in |r|(\bigcup^{\partial}\{U_i \mid i \in I\})$ and so $\exists X \subset^{f} \bigcup^{\partial}\{U_i \mid i \in I\}.X r\{b\}$. But X is finite and so $X \subset^{f} U_{i_0}$ i.e. $b \in |r|(U_{i_0}) \subset \bigcup^{\partial}\{|r|(U_i) \mid i \in I\}$.

$|-|$ is functorial on morphisms: $|\vdash| = id_{|A|}$ for any $U \in |A|$ is entailment closed. Now take $A \xrightarrow{r} B \xrightarrow{s} C$ between non-empty presystems and let $U \in |A|$; we omit to check the degenerate cases involving empty presystems. We have

$$|sr|(U) = \{c \mid \exists X \subset^{f} U.\exists Y\downarrow_B.X r Y \;\&\; Y s\{c\}\}$$

and

$$|s||r|(U) = \{c \mid \exists Y \subset^{f} \{b \mid \exists X \subset^{f} U.X r\{b\}\}.Y s\{c\}\}.$$

Take $c \in |s||r|(U)$ so that $\forall y \in Y.\exists X_y \subset^{f} U.X_y r\{y\}$ where $Y s\{c\}$. Y is finite and so we have $\bigcup\{X_y \mid y \in Y\} \vdash_A X_y r\{y\}$ leading to $\bigcup\{X_y \mid y \in Y\} r\{y\}$ and thus $\bigcup\{X_y \mid y \in Y\} r Y$. Thus $c \in |sr|(U)$. The reverse inclusion is trivial and so we may deduce from this that $|sr| = |s||r|$. This completes the proof of Proposition 7.3.3. □

Now we show that there is a functorial construction of presystems from domains. More precisely we have the

Proposition 7.3.6 There is a functor $\Xi: \mathcal{S}pd \to p\mathcal{I}n\mathcal{S}$ given by the following prescription:

On objects define ΞD to be the empty presystem if D is empty and otherwise $\Xi D \stackrel{\text{def}}{=} (D^{\circ}, \downarrow, \vdash)$ where

1. D° is the set of finite elements of D.
2. $X\downarrow$ iff $X \subset^{f} D^{\circ} \;\&\; Bd(X)$.
3. $X \vdash d$ iff $X\downarrow \;\&\; d \leq \bigvee X \;\&\; d \in D^{\circ}$.

On morphisms, $\Xi f: \Xi D \to \Xi E$ is specified by the relation $\Xi f \subset \downarrow_{\Xi D} \times \downarrow_{\Xi E}$ given by

$$X \Xi f Y \text{ iff } \bigvee Y \leq f(\bigvee X).$$

Proof We have to show that ΞD is a presystem; we just sketch the details when D is non-empty. D is algebraic so there must be at least one finite element, implying $\emptyset \neq \downarrow \subset \mathcal{P}_{fin}(D^\circ)$. We have $\vdash \subset \downarrow \times D^\circ$ by definition. The conditions (1) to (6) of Definition 7.2.6 are easy to verify. For (1) the empty set is not consistent for it is not bounded. For (6) suppose that $X \vdash Y \vdash d$; it is immediate that $d \leq \bigvee Y \leq \bigvee X$, i.e. $X \vdash d$.

Now we need to see that Ξ is well defined and functorial on continuous functions; we just sketch the former. Take $f: D \to E$ a continuous function between non-empty domains. We look at condition (1) of Definition 7.2.6. Let $d \in \Xi D$ and so by algebraicity of E there is a finite $e \in E$ with $e \leq f(d)$. Thus $\{d\}\Xi f\{e\}$; conditions (2) and (3) are equally trivial. Confirming functoriality on continuous functions is a routine calculation. □

Proof of the Equivalence

With the machinery just set up we can now prove the next theorem:

Theorem 7.3.7 The functors $|-|: p\mathcal{I}n\mathcal{S} \to \mathcal{S}pd$ and $\Xi: \mathcal{S}pd \to p\mathcal{I}n\mathcal{S}$ give rise to an equivalence of categories.

Proof Let D be a domain and A a presystem. It will be convenient to write $\Xi D \stackrel{\text{def}}{=} (D^\circ, \downarrow, \vdash)$. Of course we simply check that we have an isomorphism which is natural on components D and A; we consider D first.

There is an isomorphism natural in D, $\theta : D \cong |\Xi D| : \phi$, given by

$$\begin{aligned} \theta(d) &\stackrel{\text{def}}{=} \{e \in D^\circ \mid e \leq d\} \\ \phi(U) &\stackrel{\text{def}}{=} \bigvee^\partial U, \end{aligned}$$

on non-empty D and by the empty continuous function when $D = \emptyset$.

θ is well defined: $U \stackrel{\text{def}}{=} \{e \in D^\circ \mid e \leq d\}$ is a directed hence non-empty subset of D°. To see that $U \in |\Xi D|$ let $X \subset^f U$. Then

1. X is bounded by d so $X\downarrow$.
2. If $X \vdash e$ then $e \leq \bigvee X \leq \bigvee^\partial U = d$ and so $e \in U$.

Now for continuity: Take $S \subset D$ directed. Then $\theta(S)$ is certainly directed and that $\bigcup^\partial \theta(S) = \theta(\bigvee^\partial S)$ is easy.

ϕ is well defined: Take $U \in |\Xi D|$; we show that U is directed. U is non-empty by definition and if $\{e, e'\} \subset^f U$ we have $\{e, e'\}$ consistent in ΞD implying $Bd(\{e, e'\})$. Hence $s \stackrel{\text{def}}{=} \bigvee\{e, e'\}$ exists in D and s will clearly inherit finiteness from e and e', showing $s \in U$ on noting $\{e, e'\} \vdash s$.

ϕ is continuous: Take $\mathcal{U} \subset |\Xi D|$ directed. $\phi(\mathcal{U})$ inherits directedness from $\mathcal{U}$. Clearly $\phi(\mathcal{U}) \leq \phi(\bigcup^\partial \mathcal{U}))$ and $\mathcal{U} \leq \bigvee^\partial \phi(\mathcal{U})$; this is all we need. That the pair (ϕ, ψ)

yields an isomorphism of domains is virtually immediate; we omit the verification of naturality.

Now we write $A \stackrel{\text{def}}{=} (A, \downarrow, \vdash)$ and $\Xi|A| \stackrel{\text{def}}{=} (|A|^{\circ}, \downarrow', \vdash')$. We show that there is a natural isomorphism $r : A \cong \Xi|A| : s$ given by the empty preapproximable map when A is empty and otherwise

$$\begin{aligned} X r \{\overline{X_i} \mid i \in I\} \quad &\text{iff} \quad X_i\downarrow \;\&\; X\downarrow \;\&\; \forall i \in I.\overline{X_i} \subset \overline{X} \\ \{\overline{X_i} \mid i \in I\} s X \quad &\text{iff} \quad X_i\downarrow \;\&\; X\downarrow \;\&\; X \subset^{f} \bigcup\{\overline{X_i} \mid i \in I\}. \end{aligned}$$

This definition makes sense. To see this recall Lemma 7.3.5 and also note that the supremum exists for $\{\overline{X_i} \mid i \in I\}\downarrow'$ implying that $Bd(\{\overline{X_i} \mid i \in I\})$ in $|A|$.

r is a preapproximable map: For (1) $\{a\} r \{\overline{\{a\}}\}$. Condition (2) is easy. For condition (3) suppose that $X' \vdash X r \{\overline{X_i} \mid i \in I\} \vdash' \{\overline{X'_j} \mid j \in J\}$. The only thing not clear is that for any j we have $\overline{X'_j} \subset \overline{X'}$; but it is easy to see that $\overline{X'_j} \subset \bigcup\{\overline{X_i} \mid i \in I\} \subset \overline{X}$ and $\overline{X} \subset \overline{X'}$.

s is a preapproximable map: For (1) note that $\{\overline{X}\} s X$. Condition (2) is easy. For (3) suppose that $\{\overline{X'_j} \mid j \in J\} \vdash' \{\overline{X_i} \mid i \in I\} s X \vdash X'$. We need to see that $X' \subset^{f} \bigcup\{\overline{X'_j} \mid j \in J\}$; but it is the case that $X \subset^{f} \bigcup\{\overline{X_i} \mid i \in I\} \subset \bigcup\{\overline{X'_j} \mid j \in J\}$ and we are done by entailment closure.

Checking that $sr = \vdash$ is easy. We sketch the details for $rs = \vdash'$. Suppose that $\{\overline{X_i} \mid i \in I\} rs \{\overline{X'_j} \mid j \in J\}$ and so for some $X\downarrow$ and for each j we have $\overline{X'_j} \subset \overline{X} \subset \bigcup\{\overline{X_i} \mid i \in I\}$ implying that $\{\overline{X_i} \mid i \in I\} \vdash' \{\overline{X'_j} \mid j \in J\}$. Conversely suppose that $\{\overline{X_i} \mid i \in I\} \vdash' \{\overline{X'_j} \mid j \in J\}$. We see that

$$\bigcup\{X'_j \mid j \in J\} \subset^{f} \bigcup\{\overline{X'_j} \mid j \in J\} \subset \bigcup\{\overline{X_i} \mid i \in I\} \in |A|$$

implying $\bigcup\{X'_j \mid j \in J\}\downarrow$. Finally we can conclude that $\{\overline{X_i} \mid i \in I\} s \bigcup\{X'_j \mid j \in J\}$ and $\bigcup\{X'_j \mid j \in J\} r \{\overline{X'_j} \mid j \in J\}$ so we are done.

To finish, we give one case of naturality, namely given a preapproximable map $m\colon A \to A'$ we check that the following diagram commutes:

$$\begin{array}{ccc} \Xi|A| & \xrightarrow{\;s\;} & A \\ {\scriptstyle \Xi|m|}\downarrow & & \downarrow{\scriptstyle m} \\ \Xi|A'| & \xrightarrow{\;s\;} & A' \end{array}$$

Suppose that $\{\overline{X_i} \mid i \in I\} s \Xi|m| X'$. Then for some $\{\overline{X'_j} \mid j \in J\}$ we have $X' \subset^{f} \bigcup\{\overline{X_j} \mid j \in J\} \subset |m|(\bigcup\{\overline{X_i} \mid i \in I\})$. Recall that X' is non-empty and using the definition of $|m|$ we see that $\forall x' \in X'.\exists Y_{x'} \subset^{f} \bigcup\{\overline{X_i} \mid i \in I\}.Y_{x'} m \{x'\}$. As X' is a finite set we deduce $\bigcup\{Y_{x'} \mid x' \in X'\}\downarrow$ because $\bigcup\{Y_{x'} \mid x' \in X'\} \subset^{f} \bigcup\{\overline{X_i} \mid i \in I\}$ and thus $\bigcup\{Y_{x'} \mid x' \in X'\} \vdash Y_{x'} m \{x'\}$. Collecting our conclusions together we see that $\bigcup\{Y_{x'} \mid x' \in X'\} m X'$ and $\{\overline{X_i} \mid i \in I\} s \bigcup\{Y_{x'} \mid x' \in X'\}$ i.e. $\{\overline{X_i} \mid i \in I\} ms X'$.

Conversely let $\{\overline{X_i} \mid i \in I\} m s X'$. Then there is some $X\downarrow$ for which $\{\overline{X_i} \mid i \in I\} s X$ and $X m X'$ and taking $x' \in X'$ we have $X m \{x'\}$. Hence we deduce that $\overline{X'} \subset |m|(\bigcup\{\overline{X_i} \mid i \in I\}) \in |A'|$ using the definition of $|m|$ and entailment closure of points of $|A'|$; but this says that $\{\overline{X_i} \mid i \in I\} \Xi |m| \{\overline{X'}\}$. Of course $\{\overline{X'}\} s X'$, so we are done. □

7.4 The Large ωcpo of Presystems

Here we shall be a little more precise about set-theoretical conventions. Let us work with Zermelo Fraenkel (ZF) set theory and assume the existence of a universe of sets U. Then a *class* will be a (meta)-subset of the universe U. This can be made more precise by defining a class to be an equivalence class of ZF-formulae identified under universally quantified bi-implication, but we omit all formal details. We now show that the collection of all presystems forms an ω cocomplete partially ordered class under a suitable ordering.

Definition 7.4.1 Given presystems A and B we define an order relation $\leq$ on the class PS of all presystems where $(\emptyset, \emptyset, \emptyset)$ is a least element and for non-empty A and B, $A \leq B$ iff

1. $A \subset B$.
2. $X\downarrow_A$ iff $X \subset A \;\&\; X\downarrow_B$.
3. $X \vdash_A a$ iff $X \subset A \;\&\; a \in A \;\&\; X \vdash_B a$.

Lemma 7.4.2 If $A \leq B$ and the token set A equals that of B then $A = B$. □

Theorem 7.4.3 The class PS of presystems with the above ordering is an ω cocomplete partially ordered class with a least element.

Proof It is easy to see that the order $\leq$ is indeed a partial order. Suppose that we have a chain $\{A_i \mid i \in \omega\}$ of presystems. Then the supremum exists and is given by the presystem $(\bigcup\{A_i \mid i \in \omega\}, \bigcup\{\downarrow_i \mid i \in \omega\}, \bigcup\{\vdash_i \mid i \in \omega\})$. The least element is the empty presystem. □

Definition 7.4.4 The category $\omega\mathcal{CPO}$ has objects which are ω cocomplete partially ordered *classes* and morphisms which are Scott continuous *function classes*. Thus the category $\omega\mathcal{C}po$ is a full subcategory of $\omega\mathcal{CPO}$.

Proposition 7.4.5 The category $\omega\mathcal{CPO}$ is a let category with respect to lifting of domains which has (finite products), stable finite coproducts, $\perp$-exponentials, NNO and FPO.

Proof The only thing in doubt is the existence of $\perp$-exponentials; it is possible to check that $\omega\mathcal{CPO}$ is closed under the formation of $\perp$-exponentials in the chosen formulation of set theory. □

Our aim is to see that the object PS in the category $\omega\mathcal{CPO}$ can be used to play the rôle of a type universe. The precise details of how this can be done will emerge in Chapter 9. Clearly the essence of the idea comes from the fact that the categories $p\mathcal{I}n\mathcal{S}$ and $\mathcal{S}pd$ are equivalent, but as we have just seen, the class of all presystems can be seen as a (large) ωcpo. We saw in Proposition 7.2.4 that $\mathcal{S}pd$ has finite (co)products and $\perp$-exponentials. Thus so does $p\mathcal{I}n\mathcal{S}$. With a view to using PS as a type universe, we give constructions of these categorical structures directly, and show that their formation is continuous with respect to the order on PS. We will need the following observations:

For each finite ordinal $n \in \omega$ there is a large ωcpo $PS^n \stackrel{\text{def}}{=} PS \times \ldots \times PS$ enjoying the obvious pointwise order. Then it is easy to see that any morphism $f\colon PS^n \to PS$ is continuous iff it is so in each coordinate. This observation, together with the next lemma will be found useful in later work; we need the

Definition 7.4.6 Let $f\colon PS \to PS$ be a function class on the the underlying class of PS. Then f is said to be *continuous on token sets* if given a chain of presystems $\{A_n \mid n \in \omega\}$ we have $f(\bigcup\{A_n \mid n \in \omega\}) \subset \bigcup\{f(A_n) \mid n \in \omega\}$ where the containment is between token sets.

Lemma 7.4.7 Let $f\colon PS \to PS$ be a morphism. Then f is continuous iff f is monotonic and continuous on tokens.

Proof The "only if" case is immediate. Conversely let $\{A_n \mid n \in \omega\}$ be a chain of presystems. Then $\bigcup\{f(A_n) \mid n \in \omega\} \leq f(\bigcup\{A_n \mid n \in \omega\})$ and by Lemma 7.4.2 we are done. □

7.5 Categorical Constructions in $p\mathcal{I}n\mathcal{S}$

We give explicit (canonical) constructions of finite (co)products and $\perp$-exponentials in the category $p\mathcal{I}n\mathcal{S}$. These constructions are continuous in the following sense:

Lemma 7.5.1 There are morphisms in the category $\omega\mathcal{CPO}$

$$\ulcorner 0 \urcorner\colon 1 \to PS \qquad \ulcorner 1 \urcorner\colon 1 \to PS \qquad \ulcorner \perp \urcorner\colon PS \to PS$$
$$\ulcorner \times \urcorner\colon PS^2 \to PS \qquad \ulcorner + \urcorner\colon PS^2 \to PS \qquad \ulcorner \multimap \urcorner\colon PS^2 \to PS$$

which give rise to the canonical initial object, terminal object, liftings, binary (co)products and $\perp$-exponentials in the category $p\mathcal{I}n\mathcal{S}$.

Proof Throughout the proof let A and B be non-empty presystems; the effect of the morphisms which involve empty presystems will be clear, although we do make this explicit for the $\perp$-exponential construction. We omit all details concerning well definedness and continuity, except for $\ulcorner \multimap \urcorner$.

Put $\ulcorner 0 \urcorner(*) \stackrel{\text{def}}{=} (\emptyset, \emptyset, \emptyset)$ and $\ulcorner 1 \urcorner(*) \stackrel{\text{def}}{=} (\{*\}, \{\{*\}\}, \{*\} \vdash *)$; it is clear these definitions work.

Now we prescribe $\ulcorner \perp \urcorner\colon PS \to PS$; this will take the empty presystem to the one point presystem and take $A = (A, \downarrow, \vdash)$ to $(A_\perp, \downarrow_\perp, \vdash_\perp)$ where

- $A_\perp \stackrel{\text{def}}{=} A \cup \{\perp\}$ where $\perp \notin A$.
- $X\downarrow_\perp$ iff $X \setminus \{\perp\}\downarrow$ *or* $X = \{\perp\}$.
- $X \vdash_\perp a$ iff $X \setminus \{\perp\} \vdash a$ *or* $a = \perp$.

Given presystems $(A, \downarrow_A, \vdash_A)$ and $(B, \downarrow_B, \vdash_B)$ define $A^\ulcorner \times {}^\urcorner B$ to be $(P, \downarrow, \vdash)$ where

- $P \stackrel{\text{def}}{=} A \times B$.
- $Z\downarrow$ iff $\pi_1 Z\downarrow_A$ & $\pi_2 Z\downarrow_B$.
- $Z \vdash p$ iff $\pi_1 Z \vdash_A \pi_1 p$ & $\pi_2 Z \vdash_B \pi_2 p$ where π_1 and π_2 are the projections of the product $A \times B$ in *Set*.

We define $A + B$ to be $(C, \downarrow, \vdash)$ where

- $C \stackrel{\text{def}}{=} A + B$ where we shall put $A + B \stackrel{\text{def}}{=} \{1\} \times A \cup \{2\} \times B$.
- $Z\downarrow$ iff $\exists X\downarrow_A . Z = \{1\} \times X$ *or* $\exists Y\downarrow_B . Z = \{2\} \times Y$.
- $Z \vdash c$ iff $\exists a \in A . \exists X\downarrow_A . Z = \{1\} \times X$ & $c = (1, a)$ & $X \vdash_A a$ *or* $\exists b \in B . \exists Y\downarrow_B . Z = \{2\} \times Y$ & $c = (2, b)$ & $Y \vdash_B b$.

The morphism $^\ulcorner\!\!\rightarrow\!\!{}^\urcorner : PS^2 \to PS$ is defined by

$$\begin{array}{lcl} ((\emptyset,\emptyset,\emptyset),(\emptyset,\emptyset,\emptyset)) & \mapsto & (\{*\},\{\{*\}\},\{*\} \vdash *) \\ (A,(\emptyset,\emptyset,\emptyset)) & \mapsto & (\{*\},\{\{*\}\},\{*\} \vdash *) \\ ((\emptyset,\emptyset,\emptyset),A) & \mapsto & (\{*\},\{\{*\}\},\{*\} \vdash *) \\ (A,B) & \mapsto & (F,\downarrow,\vdash) \end{array}$$

where

- $F \stackrel{\text{def}}{=} \{X \blacktriangleright b \mid X\downarrow_A \ \&\ b \in B\} \cup \{*\}$.
- $Z\downarrow$ iff $Z \subset^f \mathcal{P}_{fin}(F)$ *and* it is either the case that $Z = \{*\}$ *or* we have $Z \setminus \{*\} \neq \emptyset$ & $\forall I \subset^f n . \bigcup\{X_i \mid i \in I\}\downarrow_A$ implies $\{b_i \mid i \in I\}\downarrow_B$ where $n = \{0, \ldots, n-1\}$ and $Z \setminus \{*\} = \{X_i \blacktriangleright b_i \mid i \in n\}$.
- $Z \vdash f$ iff $Z\downarrow$ *and* either $f = *$ or $f \neq *$ & $Z \setminus \{*\} \neq \emptyset$ & $\bigcup\{b_i \mid X \vdash_A X_i\} \vdash_B b$.

First we see that F really is a presystem. As A and B are non-empty the basic criteria of Definition 7.2.6 are satisfied. Further:

1. $\emptyset \notin \downarrow$ by definition.
2. If $Z' \subset^f Z\downarrow$ then $Z'\downarrow$ is immediate.
3. Let $f \in F$. If $f = *$ then $\{*\}\downarrow$ by definition and if $f = X \blacktriangleright b$ just note that $X\downarrow_A$ and $\{b\}\downarrow_B$.

4. Let $Z \vdash f$. If $f = *$ then $Z \cup \{*\}\downarrow$ is immediate. Otherwise let $f = X \blacktriangleright b$ and write $\{X_i \blacktriangleright b_i \mid i \in n\}$ for the non-empty set $Z \setminus \{*\}$. Let $I \subset^f n$. Suppose that $\bigcup\{X_i \mid i \in I\} \cup X\downarrow_A$. Using Lemma 7.3.4 we may deduce that $\bigcup\{X_i \mid i \in I\} \cup \bigcup\{X_j \mid X \vdash_A X_j\}\downarrow_A$ and as $Z\downarrow$ this means that $\bigcup\{b_i \mid i \in I\} \cup \bigcup\{b_j \mid X \vdash_A X_j\}\downarrow_B$. Finally note that $\bigcup\{b_j \mid X \vdash_A X_j\} \vdash_B b$ from which we can deduce that $\bigcup\{b_i \mid i \in I\} \cup \{b\}\downarrow_B$.

5. Let $Z\downarrow$ and $X \blacktriangleright b \in Z$. Write $Z \setminus \{*\} = \{X_i \blacktriangleright b_i \mid i \in n\}$ where $X_0 = X$ and $b_0 = b$. Then it is trivial that $\bigcup\{b_i \mid X \vdash_A X_i\} \vdash_B b$.

6. Suppose that $Z' \vdash Z \vdash f$. The result is trivial if $f = *$ and so let $f = X \blacktriangleright b$. Checking the definitions we may safely put $Z \setminus \{*\} = \{X_i \blacktriangleright b_i \mid i \in n\}$ and $Z' \setminus \{*\} = \{X'_j \blacktriangleright b'_j \mid j \in n'\}$ and deduce that $\bigcup\{b_i \mid X \vdash_A X_i\} \vdash_B b$ and $\bigcup\{b'_j \mid X_i \vdash_A X'_j\} \vdash_B b_i$. (Note that we have $X \vdash X_i$ for at least one i). Obviously $X \vdash_A \bigcup\{X'_j \mid X \vdash_A X'_j\}$ and so by Lemma 7.3.4 we have $\{X'_j \mid X \vdash_A X'_j\}\downarrow_A$. Hence $\bigcup\{b'_j \mid X \vdash X'_j\}\downarrow_B$ and from $\bigcup\{b'_j \mid X_i \vdash_A X'_j\} \subset \bigcup\{b'_j \mid X \vdash_A X'_j\}$ we can deduce that $\bigcup\{b'_j \mid X \vdash_A X'_j\} \vdash_B b_i$. Thus $\bigcup\{b'_j \mid X \vdash_A X'_j\} \vdash_B b$ as required.

It remains to show that $\ulcorner\!\multimap\!\urcorner : PS^2 \to PS$ is a morphism in $\omega\mathcal{CPO}$. We have only to see continuity in each coordinate, and appealing to Lemma 7.4.7 we may simply verify that $\ulcorner\!\multimap\!\urcorner$ is continuous on token sets in each coordinate. We verify this for the first coordinate of $\ulcorner\!\multimap\!\urcorner$.
Take $A \leq A'$ and B in PS. We show $F \stackrel{\text{def}}{=} A\ulcorner\!\multimap\!\urcorner B \leq F' \stackrel{\text{def}}{=} A'\ulcorner\!\multimap\!\urcorner B$.

1. $\downarrow_A \subset \downarrow_{A'}$ so $F \subset F'$.

2. Note that $Z\downarrow$ implies $Z \subset F$. We need $Z\downarrow$ iff $Z \subset F$ and $Z\downarrow'$. This is clear unless $Z \setminus \{*\}$ is non-empty, say $Z \setminus \{*\} = \{X_i \blacktriangleright b_i \mid i \in n\}$. Using the note we see $\bigcup\{X_i \mid i \in I\}\downarrow_A$ iff $\{X_i \mid i \in I\}\downarrow_{A'}$ and so $Z\downarrow$ iff $Z\downarrow'$.

3. We need to show that $Z \vdash f$ iff $Z \subset F$ & $f \in F$ & $Z \vdash' f$. This is only non trivial if f is not $*$ and $\emptyset \neq Z \setminus \{*\} = \{X_i \blacktriangleright b_i \mid i \in n\}$ with $f = X \blacktriangleright b$. Note that $Z \vdash f$ implies $Z \subset F$ and as $X \vdash_A X_i$ iff $X \vdash_{A'} X_i$ we have $Z \vdash f$ iff $Z \vdash' f$.

Take a directed set $\{A_i \mid i \in I\}$ in PS. We need to prove $(\bigvee^\partial\{A_i \mid i \in I\})\ulcorner\!\multimap\!\urcorner B \subset \bigvee^\partial\{F_i \mid i \in I\}$. Take $f \in (\bigvee^\partial\{A_i \mid i \in I\})\ulcorner\!\multimap\!\urcorner B$. Certainly we are okay if f is $*$; let $f = X \blacktriangleright b$. Then $X\downarrow_{\bigvee^\partial\{A_i \mid i \in I\}}$, so $X \subset^f A_i$ for some i and hence $X\downarrow_{A_i}$. Therefore $X \blacktriangleright b \in A_i\ulcorner\!\multimap\!\urcorner B \subset \bigvee^\partial\{F_i \mid i \in I\}$. □

Definition 7.5.2 Suppose that A and B are Scott predomains. A continuous function $m: A \to B$ is called an *embedding* if there is a (necessarily unique) continuous *partial* function $r: B \to A$ for which $rm = id_A$ and $mr \leq id_B$. r is referred to as a *partial-projection*. The class of Scott predomains together with embeddings (and specified partial-projections) forms a category which will be denoted by $\mathcal{Spd}^{ep}$. We shall write $r(a)\Downarrow$ to mean that r is defined at $a \in A$.

Lemma 7.5.3 There is a functor $|-|: PS \to \mathcal{S}pd^{ep}$. Moreover, the effect of $|-|$ on objects commutes up to isomorphism (in $\mathcal{S}pd$) with the functors $\ulcorner g \urcorner$ and g where g runs over 0, 1, $\perp$, $\times$, $+$ and $\multimap$.

Proof $|-|$ is defined on objects by taking points. Given $A \leq A'$, $U \in |A|$ and $U' \in |A'|$ we define an embedding partial-projection pair (m, r) where

$$m : |A| \underset{\longleftarrow}{\overset{\subset}{\longrightarrow}} |A'| : r$$

by setting

$$\begin{aligned} m(U) &\stackrel{\text{def}}{=} \{a' \in A' \mid \exists X \subset^f U. X \vdash_{A'} a'\}, \\ r(U') &\stackrel{\text{def}}{=} \begin{cases} A \cap U' & \text{if this is non-empty} \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

We omit the routine details verifying that (m, r) is a well defined embedding partial-projection pair. We give the definitions of the isomorphisms and check details for $\multimap$. We begin by noting that all is clear for 0 and 1.

Define $\vartheta : |A|_\perp \cong |\ulcorner \perp \urcorner(A)| : \varphi$ by

$$\vartheta(U) \stackrel{\text{def}}{=} \begin{cases} U \cup \{\perp\} & \text{if } U \neq \perp \\ \{\perp\} & \text{otherwise} \end{cases} \qquad \varphi(V) \stackrel{\text{def}}{=} \begin{cases} V \setminus \{\perp\} & \text{if } V \neq \{\perp\} \\ \perp & \text{otherwise} \end{cases}$$

Define $\vartheta : |A| \times |B| \cong |A^\ulcorner \times \urcorner B| : \varphi$ by $\vartheta(U, V) \stackrel{\text{def}}{=} U \times V$ and $\varphi(W) \stackrel{\text{def}}{=} (\pi_1 W, \pi_2 W)$.

Define $\vartheta : |A| + |B| \cong |A^\ulcorner + \urcorner B| : \varphi$ by

$$\begin{aligned} \vartheta(C) &\stackrel{\text{def}}{=} \begin{cases} \{1\} \times U & \text{if } C = (1, U) \\ \{2\} \times V & \text{if } C = (2, V) \end{cases} \\ \varphi(W) &\stackrel{\text{def}}{=} \begin{cases} (1, \bigcup\{X_Z \mid Z \subset^f W\}) \text{ where } Z = \{1\} \times X_Z \\ \qquad\qquad\qquad\qquad or \\ (2, \bigcup\{Y_Z \mid Z \subset^f W\}) \text{ where } Z = \{2\} \times Y_Z \end{cases} \end{aligned}$$

where the definition of X_Z and Y_Z and well definedness of φ can be seen from inspection of the construction of $A^\ulcorner + \urcorner B$ in the proof of Lemma 7.5.1.

Define $\vartheta : (|A| \to |B|_\perp) \cong |A^\ulcorner \multimap \urcorner B| : \varphi$ by

$$\begin{aligned} \vartheta(g) &\stackrel{\text{def}}{=} \{X \blacktriangleright b \mid b \in g(\overline{X})\} \cup \{*\} \\ \varphi(V)(U) &\stackrel{\text{def}}{=} \begin{cases} \{b \mid \exists X \subset^f U. X \blacktriangleright b \in V\} & \text{if this is non-empty} \\ \text{else } \perp \end{cases} \end{aligned}$$

(Recall that $|A| \multimap |B| \stackrel{\text{def}}{=} |A| \to |B|_\perp$; see Proposition 7.2.4). ϑ is well defined: Take $Z \subset^f \vartheta(g)$. If $Z = \{*\}$ then $Z\!\downarrow$. Suppose not; take $Z \setminus \{*\} \stackrel{\text{def}}{=} \{X_i \blacktriangleright b_i \mid i \in n\}$, $I \subset^f n$ and let $\bigcup\{X_i \mid i \in I\}\!\downarrow_A$. From the definition of ϑ and the monotonicity of g we see that

$$\{b_i \mid i \in I\} \subset^f \bigcup\{g(\overline{X_i}) \mid i \in I\} \subset g(\overline{\bigcup\{X_i \mid i \in I\}}) \in |B|_\perp.$$

Thus $Z\downarrow$ as required. Now suppose also that $Z \vdash f$. If $f = *$ then $f \in \vartheta(g)$. Suppose not. We have $\bigcup\{b_i \mid X \vdash X_i\} \vdash_B b$ where $f = X \blacktriangleright b$ and so $X \vdash X_i$ for at least one i: let I index such X_i. Note that $X \vdash \bigcup\{X_i \mid i \in I\}$ so by Lemma 7.3.4 $\bigcup\{X_i \mid i \in I\}\downarrow_A$. Of course $\overline{\bigcup\{X_i \mid i \in I\}} \subset \overline{X}$ and arguing as above we conclude $\{b_i \mid i \in I\} \subset^f g(\overline{X}) \in |B|_\perp$ implying that $b \in g(\overline{X})$. Hence $f \in \vartheta(g)$.

φ is well defined: Take $V \in |A^\Gamma \multimap \neg B|$, $U \in |A|$ and show $\varphi(V)(U) \in |B|_\perp$. For the non trivial case take $Y \subset^f \varphi(V)(U)$ and let $y \in Y$. Then $\exists X_y \subset^f U. X_y \blacktriangleright y \in V$. Noting that $\bigcup\{X_y \mid y \in Y\} \subset^f U$ and $\{X_y \blacktriangleright y \mid y \in Y\} \subset^f V$ along with the definition of $\downarrow$ we have $\bigcup\{y \mid y \in Y\} = Y\downarrow_B$. Now suppose that $Y \vdash_B b$. Noting that $Y = \bigcup\{y \mid \bigcup\{X_y \mid y \in Y\} \vdash_A X_y\} \vdash_B b$ we have $\{X_y \blacktriangleright y \mid y \in Y\} \vdash \bigcup\{X_y \mid y \in Y\} \blacktriangleright b$ which implies that $\bigcup\{X_y \mid y \in Y\} \blacktriangleright b \in V$ and thus $b \in \varphi(V)(U)$. Finally we have to see that φ is a continuous function; we omit the simple details.

(ϑ, φ) is an isomorphism: Suppose that $\varphi(\vartheta(g))(U)$ is not $\perp$. Then we have

$$\begin{aligned}
\varphi(\vartheta(g))(U) &= \{b \mid \exists X \subset^f U. X \blacktriangleright b \in \{X \blacktriangleright b \mid b \in g(\overline{X})\} \cup \{*\}\} \\
&= \{b \mid \exists X \subset^f U. b \in g(\overline{X})\} \\
&= \bigcup^\vartheta \{g(\overline{X}) \mid X \subset^f U\} \\
&= g(U)
\end{aligned}$$

where the final step follows by continuity of g. For the converse, note that

$$\vartheta(\varphi(V)) = \{X \blacktriangleright b \mid \exists X' \subset^f \overline{X}. X' \blacktriangleright b \in V\} \cup \{*\}.$$

Certainly $V \subset \vartheta\varphi(V)$. Now take $X \blacktriangleright b \in \vartheta\varphi(V)$. We can find some X' for which $X \vdash_A X'$ and hence $\bigcup\{b \mid X \vdash_A X'\} \vdash_B b$ implies $\{X' \blacktriangleright b\} \vdash X \blacktriangleright b$. Thus $X \blacktriangleright b \in V$, completing the proof. □

Remark 7.5.4 Note that the construction of the product $A^\Gamma \times \neg B$ in $p\mathcal{I}n\mathcal{S}$ is quite different from the usual construction of products of information systems; for a discussion of products of information systems see [Sco82]. Note also that we have not given the full details of the categorical constructions, for example the preapproximable maps which are the projections of $|A^\Gamma \times \neg B|$. The details are easy to fill in.

7.6 The Small ωcpo of Presystems

We can avoid the set-theoretical complications of the last section by restricting the cardinality of the sets which underly our constructions.

Definition 7.6.1 A Scott predomain is *countably based* if its set of finite elements is countable. Such Scott predomains give rise to a full sub-category $\mathcal{S}pd_\omega$ of $\mathcal{S}pd$. A presystem is *countably based* if its token set is countable. Such presystems give rise to a full sub-category $p\mathcal{I}n\mathcal{S}_\omega$ of $p\mathcal{I}n\mathcal{S}$.

With this definition, it is not too difficult to verify the following results, which will be put to use in Chapter 9.

Theorem 7.6.2 The functors $|-|: p\mathcal{I}n\mathcal{S} \to \mathcal{S}pd$ and $\Xi: \mathcal{S}pd \to p\mathcal{I}n\mathcal{S}$ defined in Section 7.3 restrict to an equivalence $\mathcal{S}pd_\omega \simeq p\mathcal{I}n\mathcal{S}_\omega$. □

Lemma 7.6.3 The set $PS_{\mathbb{N}}$ of all presystems whose token sets are subsets of $\mathbb{N}$ is an ωcpo with least element.

Proof Suprema in $PS_{\mathbb{N}}$ are given by unions and the least element is the empty presystem. The details are easy to verify. □

Lemma 7.6.4 There are morphisms in the category $\omega\mathcal{C}po$

$$\begin{array}{lll} \ulcorner 0 \urcorner : 1 \to PS_{\mathbb{N}} & \ulcorner 1 \urcorner : 1 \to PS_{\mathbb{N}} & \ulcorner \bot \urcorner : PS_{\mathbb{N}} \to PS_{\mathbb{N}} \\ \ulcorner \times \urcorner : PS_{\mathbb{N}}^2 \to PS_{\mathbb{N}} & \ulcorner + \urcorner : PS_{\mathbb{N}}^2 \to PS_{\mathbb{N}} & \ulcorner \rightharpoonup \urcorner : PS_{\mathbb{N}}^2 \to PS_{\mathbb{N}} \end{array}$$

which give rise to the canonical initial object, terminal object, liftings, binary (co)products and $\bot$-exponentials in the category $p\mathcal{I}n\mathcal{S}_\omega$. □

Lemma 7.6.5 There is a functor $|-|: PS_{\mathbb{N}} \to \mathcal{S}pd_\omega^{ep}$. Moreover, the effect of $|-|$ on objects commutes up to isomorphism (in $\mathcal{S}pd_\omega$) with the functors $\ulcorner g \urcorner$ and g where g runs over 0, 1, $\bot$, $\times$, $+$ and $\rightharpoonup$. □

7.7 Some Miscellaneous Results

While it is not central to out main concerns, we note a pleasing relationship between Scott families of sets and presystems, namely that there is a one to one correspondence between them.

Definition 7.7.1 Given a Scott family of sets $\mathcal{F}$ define $A_{\mathcal{F}} = (A_{\mathcal{F}}, \downarrow, \vdash)$ to be the empty presystem if $\mathcal{F}$ is empty and otherwise put

1. $A_{\mathcal{F}} \stackrel{\text{def}}{=} \bigcup \mathcal{F}$.
2. $X\downarrow$ iff $X \neq \emptyset$ & $\exists U \in \mathcal{F}. X \subset^f U$.
3. $X \vdash a$ iff $X\downarrow$ & $a \in \bigcup \mathcal{F}$ & $(\forall U \in \mathcal{F}. X \subset U$ implies $a \in U)$.

Proposition 7.7.2 For a Scott family $\mathcal{F}$, $A_{\mathcal{F}}$ is a preinformation system. □

Proposition 7.7.3 Let $\mathcal{F}$ be a Scott family. Then $|A_{\mathcal{F}}| = \mathcal{F}$.

Proof We sketch the details. Showing $\mathcal{F} \subset |A_{\mathcal{F}}|$ is easy. To prove the converse take $U \in |A_{\mathcal{F}}|$ (we assume that $\mathcal{F}$ is non-empty). The proof goes by approximating U via appropriately chosen sets; in particular that $U = \bigcup^\partial \{\overline{X} \mid X \subset^f U\} \in \mathcal{F}$. To do this, set $\mathcal{V} \stackrel{\text{def}}{=} \{V \in \mathcal{F} \mid X \subset^f V\}$. It is clear that $\bigcap \mathcal{V}$ is non empty and so $\overline{X} = \bigcap \mathcal{V} \in \mathcal{F}$; hence $\{\overline{X} \mid X \subset^f U\} \subset \mathcal{F}$. Note that this latter set is directed ($\{\overline{X}, \overline{Y}\} \subset \overline{X \cup Y}$); showing U is its supremum is easy. □

Proposition 7.7.4 Let $A = (A, \downarrow, \vdash)$ be a presystem. Then $A = A_{|A|}$.

Proof The empty case is trivial so let A be non-empty. We need to show the following facts:

1. $A_{|A|} = \bigcup |A|$.
2. $X \downarrow_{A_{|A|}}$ iff $X \neq \emptyset$ & $\exists U \in |A|.X \subset^f U$.
3. $X \vdash_{A_{|A|}}$ iff $X \downarrow_{A_{|A|}}$ & $a \in \bigcup |A|$ & $(\forall U \in |A|.X \subset^f U$ implies $a \in U)$.

The easy details are omitted. □

Chapter 8

The $\mathrm{FIX}^{\mu}_{=}$ Logical System

Our goal is to define a logic in which we are able to solve recursive domain equations with the aid of the fixpoint type. The approach we adopt is to set up a logic in which there is a universal type [Car86]. The elements of this type act as codes for the external or observable types. Thus a recursive type can be realised by considering the corresponding fixpoint of the universal type. In order to make things precise, we shall define a dependently typed equational logic called $\mathrm{FIX}^{\mu}_{=}$. This is essentially the same as $\mathrm{FIX}_{=}$ but has T-exponentials and a universal type.

8.1 The Dependently Typed Equational Logic $\mathrm{FIX}^{\mu}_{=}$

Signatures for $\mathrm{FIX}^{\mu}_{=}$

A $\mathrm{FIX}^{\mu}_{=}$ signature Sg is specified by

- A collection of *basic type valued function symbols* which are tagged with an arity t: $\mathrm{TERM}^n \to \mathrm{TYPE}$.
- A collection of *distinguished type valued function symbols* denoted by *unit*, *null*, *nat*, *fix*, *dom*, *El*.
- A collection of *distinguished type valued type constructor symbols* $\times, +, \rightharpoonup, T$.
- A collection of *basic term valued function symbols* which are tagged with an arity f: $\mathrm{TERM}^n \to \mathrm{TERM}$.
- A collection of *distinguished term valued function symbols* which consists of the distinguished function symbols from a $\mathrm{FIX}_{=}$ signature augmented with $\ulcorner null \urcorner$, $\ulcorner unit \urcorner$, $\ulcorner \times \urcorner$, $\ulcorner + \urcorner$, $\ulcorner \rightharpoonup \urcorner$, $\ulcorner T \urcorner$, lp, Jp, lc, Jc, lf, Jf, Ret.

We now define an abstract syntax signature $\Sigma = (Gar, Con)$ where we shall set Gar=$\{\mathrm{TYPE}, \mathrm{TERM}\}$ and Con consists of the function symbols, type constructor symbols and a countable number of object level variables of arity TERM. The distinguished symbols have the following arities:

- *unit*, *null*, *nat*, *fix*, *dom* : TYPE.
- *El* : $\mathrm{TERM} \to \mathrm{TYPE}$.

- $\times, +, \rightharpoonup : \text{TYPE}^2 \to \text{TYPE}$.
- $T : \text{TYPE} \to \text{TYPE}$.
- $\ulcorner null \urcorner, \ulcorner unit \urcorner : \text{TERM}$.
- $\ulcorner T \urcorner$, lp, Jp, lc, Jc, lf, Jf, Ret : $\text{TERM} \to \text{TERM}$.
- $\ulcorner \times \urcorner, \ulcorner + \urcorner, \ulcorner \rightharpoonup \urcorner : \text{TERM}^2 \to \text{TERM}$.

The *raw* $\text{FIX}^{\mu}_{=}$ *types* are closed expressions of the abstract syntax generated from Σ which have arity TYPE and the *raw* $\text{FIX}^{\mu}_{=}$ *terms* closed expressions with arity TERM.

Judgements in $\text{FIX}^{\mu}_{=}$

The logic $\text{FIX}^{\mu}_{=}$ is a dependently typed equational logic. The forms of judgement which we use involve contexts Γ, raw terms M, raw types α and finite lists of raw terms χ. A *context*

$$\Gamma = [x_1 : \alpha_1, \ldots, x_n : \alpha_n]$$

is a finite list of (variable,type) pairs where the variables are distinct and $\text{OV}(\alpha_i) \subset \{x_1, \ldots, x_{i-1}\}$, with $\text{OV}(\alpha)$ the finite set of *object level* variables in α. We use the notation LEN(L) to denote the length of a list L. The judgements that we consider are

1. Γ ctxt.
2. $\Gamma \vdash \alpha$ type where $\text{OV}(\alpha) \subset \text{OV}(\Gamma)$.
3. $\Gamma \vdash M : \alpha$ where $\text{OV}(M) \cup \text{OV}(\alpha) \subset \text{OV}(\Gamma)$.
4. $\chi : \Gamma \to \Gamma'$ where $\text{OV}(\chi) \subset \text{OV}(\Gamma)$.
5. $\Gamma = \Gamma'$ where $\text{LEN}(\Gamma) = \text{LEN}(\Gamma')$.
6. $\Gamma \vdash \alpha = \alpha'$ where $\text{OV}(\alpha) \cup \text{OV}(\alpha') \subset \text{OV}(\Gamma)$.
7. $\Gamma \vdash M = M' : \alpha$ where $\text{OV}(M) \cup \text{OV}(M') \cup \text{OV}(\alpha) \subset \text{OV}(\Gamma)$.
8. $\chi = \chi' : \Gamma \to \Gamma'$ where $\text{OV}(\chi) \cup \text{OV}(\chi') \subset \text{OV}(\Gamma)$ and $\text{LEN}(\chi) = \text{LEN}(\chi') = \text{LEN}(\Gamma)$.

Equational theories for $\text{FIX}^{\mu}_{=}$

A $\text{FIX}^{\mu}_{=}$ theory Th over a signature Sg is specified by the following data:

- For each basic type valued function symbol an *introductory axiom* of the form $\Gamma_t \vdash t(\vec{x})$ type.

- For each basic term valued function symbol an *introductory axiom* of the form $\Gamma_f \vdash f(\vec{x}) : \alpha_f$.
- A collection of judgements of the form $\Gamma \vdash M = M' : \alpha$ called the *term equality axioms.*
- A collection of judgements of the form $\Gamma \vdash \alpha = \alpha'$ called the *type equality axioms.*

The *theorems* of Th are exactly those judgements which are provable from the following rules:

Equational Logic: Contexts

$$\frac{}{[]\ \text{ctxt}} \qquad \frac{}{[] = []} \qquad \frac{\Gamma \vdash \alpha\ \text{type}}{\Gamma, x:\alpha\ \text{ctxt}} \qquad \frac{\Gamma = \Gamma' \quad \Gamma \vdash \alpha = \beta(\vec{x})}{\Gamma, x:\alpha = \Gamma', y:\beta(\vec{y})}$$

Equational Logic: Types

$$\frac{\Gamma \vdash \alpha\ \text{type}}{\Gamma \vdash \alpha = \alpha} \qquad \frac{\Gamma \vdash \alpha = \alpha'}{\Gamma \vdash \alpha' = \alpha}$$

$$\frac{\Gamma \vdash \alpha = \alpha' \quad \Gamma \vdash \alpha' = \alpha''}{\Gamma \vdash \alpha = \alpha''} \qquad \frac{\chi = \chi' : \Gamma \to \Gamma' \quad \Gamma' \vdash \alpha(\vec{y}) = \alpha'(\vec{y})}{\Gamma \vdash \alpha(\chi) = \alpha'(\chi')}$$

Equational Logic: Terms

$$\frac{\Gamma, x:\alpha, \Gamma'\ \text{ctxt}}{\Gamma, x:\alpha, \Gamma' \vdash x:\alpha} \qquad \frac{\Gamma \vdash M:\alpha \quad \Gamma \vdash \alpha = \alpha'}{\Gamma \vdash M:\alpha'} \qquad \frac{\Gamma \vdash M = M':\alpha \quad \Gamma \vdash \alpha = \alpha'}{\Gamma \vdash M = M':\alpha'}$$

$$\frac{\Gamma \vdash M:\alpha}{\Gamma \vdash M = M:\alpha} \qquad \frac{\Gamma \vdash M = M':\alpha}{\Gamma \vdash M' = M:\alpha} \qquad \frac{\Gamma \vdash M = M':\alpha \quad \Gamma \vdash M' = M'':\alpha}{\Gamma \vdash M = M'':\alpha}$$

$$\frac{\chi = \chi' : \Gamma \to \Gamma' \quad \Gamma' \vdash M(\vec{y}) = M'(\vec{y}) : \alpha(\vec{y})}{\Gamma \vdash M(\chi) = M'(\chi') : \alpha(\chi)}$$

Equational logic: Context Morphisms

$$\frac{\Gamma\ \text{ctxt}}{[] : \Gamma \to []} \qquad \frac{\chi : \Gamma \to \Gamma' \quad \Gamma' \vdash \beta(\vec{y})\ \text{type} \quad \Gamma \vdash M : \beta(\chi)}{\chi, M : \Gamma \to \Gamma', y:\beta(\vec{y})}$$

$$\frac{\Gamma\ \text{ctxt}}{[] = [] : \Gamma \to []} \qquad \frac{\chi = \chi' : \Gamma \to \Gamma' \quad \Gamma' \vdash \beta(\vec{y})\ \text{type} \quad \Gamma \vdash M = M' : \beta(\chi)}{\chi, M = \chi', M' : \Gamma \to \Gamma', y:\beta(\vec{y})}$$

Equational Logic: Axioms 1

$$\frac{\chi : \Gamma \to \Gamma_t}{\Gamma \vdash t(\chi)\ \text{type}} \qquad \frac{\chi : \Gamma \to \Gamma_f \quad \Gamma_f \vdash \alpha_f(\vec{x})\ \text{type}}{\Gamma \vdash f(\chi) : \alpha_f(\chi)}$$

provided t has introductory axiom $\Gamma \vdash t(\vec{x})$ type and that f has introductory axiom $\Gamma_f \vdash f(\vec{x}) : \alpha_f$ type.

Equational Logic: Axioms 2

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash M' : \alpha}{\Gamma \vdash M = M' : \alpha} \qquad \frac{\Gamma \vdash \alpha \text{ type} \quad \Gamma \vdash \alpha' \text{ type}}{\Gamma \vdash \alpha = \alpha'}$$

provided that $\Gamma \vdash M = M' : \alpha$ is term equality axiom and that $\Gamma \vdash \alpha = \alpha'$ is a type equality axiom.

Elementary External Types

$$\frac{[] : \Gamma \to []}{\Gamma \vdash t \text{ type}}$$

where t is one of the types *unit*, *null*, *nat*, *fix*, *dom*.

External Binary Product

$$\frac{\Gamma \vdash \alpha \text{ type} \quad \Gamma \vdash \beta \text{ type}}{\Gamma \vdash \alpha \times \beta \text{ type}}$$

External Binary Coproduct

$$\frac{\Gamma \vdash \alpha \text{ type} \quad \Gamma \vdash \beta \text{ type}}{\Gamma \vdash \alpha + \beta \text{ type}}$$

External T-Exponential

$$\frac{\Gamma \vdash \alpha \text{ type} \quad \Gamma \vdash \beta \text{ type}}{\Gamma \vdash \alpha \multimap \beta \text{ type}}$$

External Computation

$$\frac{\Gamma \vdash \alpha \text{ type}}{\Gamma \vdash T\alpha \text{ type}}$$

External Decoding

$$\frac{\Gamma \vdash D : dom}{\Gamma \vdash El(D) \text{ type}}$$

Terms in Context

The rules for term formation in $\text{FIX}_{=}$ are part of $\text{FIX}^{\mu}_{=}$ with the rule for Function Terms (see Page 25) replaced by the rule for T-exponentials.

T-Exponential Terms

$$\frac{\Gamma, x{:}\alpha \vdash F(x){:}T\beta}{\Gamma \vdash \lambda_\alpha(F){:}\alpha \multimap \beta} \qquad \frac{\Gamma \vdash M{:}\alpha \multimap \beta \quad \Gamma \vdash N{:}\beta}{\Gamma \vdash MN{:}T\beta}$$

Internal Elementary Types

$$\Gamma \vdash \ulcorner unit \urcorner{:}\, dom \qquad \Gamma \vdash \ulcorner nul \Gamma \urcorner{:}\, dom$$

Internal Binary Product

$$\frac{\Gamma \vdash D{:}\, dom \quad \Gamma \vdash D'{:}\, dom}{\Gamma \vdash D^{\ulcorner} \times^{\urcorner} D'{:}\, dom}$$

Internal Binary Coproduct

$$\frac{\Gamma \vdash D{:}\, dom \quad \Gamma \vdash D'{:}\, dom}{\Gamma \vdash D^{\ulcorner} +^{\urcorner} D'{:}\, dom}$$

Internal T-Exponential

$$\frac{\Gamma \vdash D{:}\, dom \quad \Gamma \vdash D'{:}\, dom}{\Gamma \vdash D^{\ulcorner} \multimap^{\urcorner} D'{:}\, dom}$$

Internal Computation

$$\frac{\Gamma \vdash D{:}\, dom}{\Gamma \vdash \ulcorner T \urcorner D{:}\, dom}$$

Product Externalisation Terms

$$\frac{\Gamma \vdash M{:}\, El(D^{\ulcorner} \times^{\urcorner} D')}{\Gamma \vdash \mathsf{Ip}(M){:}\, El(D) \times El(D')} \qquad \frac{\Gamma \vdash P{:}\, El(D) \times El(D')}{\Gamma \vdash \mathsf{Jp}(P){:}\, El(D^{\ulcorner} \times^{\urcorner} D')}$$

Coproduct Externalisation Terms

$$\frac{\Gamma \vdash M{:}\, El(D^{\ulcorner} +^{\urcorner} D')}{\Gamma \vdash \mathsf{Ic}(M){:}\, El(D) + El(D')} \qquad \frac{\Gamma \vdash C{:}\, El(D) + El(D')}{\Gamma \vdash \mathsf{Jc}(C){:}\, El(D^{\ulcorner} +^{\urcorner} D')}$$

T-Exponential Externalisation Terms

$$\frac{\Gamma \vdash M{:}\, El(D^{\ulcorner} \multimap^{\urcorner} D')}{\Gamma \vdash \mathsf{If}(M){:}\, El(D) \multimap El(D')} \qquad \frac{\Gamma \vdash F{:}\, El(D) \multimap El(D')}{\Gamma \vdash \mathsf{Jf}(F){:}\, El(D^{\ulcorner} \multimap^{\urcorner} D')}$$

Universal Type Retraction Terms

$$\frac{\Gamma \vdash E : T dom}{\Gamma \vdash \mathsf{Ret}(E) : dom}$$

Computation Externalisation

$$\frac{\Gamma \vdash D : dom}{\Gamma \vdash El(\ulcorner T \urcorner D) = TEl(D)}$$

Equations in Context

The rules for equation formation in $\mathrm{FIX}_{=}$ are part of $\mathrm{FIX}^{\mu}_{=}$ with the rule for Function Equations (see Page 27) replaced by the rule for T-Exponential Equations.

T-Exponential Equations

$$\frac{\Gamma \vdash M : \alpha \rightharpoonup \beta}{\Gamma \vdash \lambda_{\alpha}(x.Mx) = M : \alpha \rightharpoonup \beta} \qquad \frac{\Gamma, x : \alpha \vdash F(x) : T\beta \quad \Gamma \vdash M : \alpha}{\Gamma \vdash \lambda_{\alpha}(F)\ M = F(M) : T\beta}$$

Product Externalisation Equations

$$\frac{\Gamma \vdash M : El(D^{\ulcorner} \times {}^{\urcorner} D')}{\Gamma \vdash \mathsf{Jp}(\mathsf{Ip}(M)) = M : El(D^{\ulcorner} \times {}^{\urcorner} D')} \qquad \frac{\Gamma \vdash P : El(D) \times El(D')}{\Gamma \vdash \mathsf{Ip}(\mathsf{Jp}(P)) = P : El(D) \times El(D')}$$

Coproduct Externalisation Equations

$$\frac{\Gamma \vdash M : El(D^{\ulcorner} + {}^{\urcorner} D')}{\Gamma \vdash \mathsf{Jc}(\mathsf{Ic}(M)) = M : El(D^{\ulcorner} + {}^{\urcorner} D')} \qquad \frac{\Gamma \vdash C : El(D) + El(D')}{\Gamma \vdash \mathsf{Ic}(\mathsf{Jc}(C)) = C : El(D) + El(D')}$$

T-Exponential Externalisation Equations

$$\frac{\Gamma \vdash M : El(D^{\ulcorner} \rightharpoonup {}^{\urcorner} D')}{\Gamma \vdash \mathsf{Jf}(\mathsf{If}(M)) = M : El(D^{\ulcorner} \rightharpoonup {}^{\urcorner} D')} \qquad \frac{\Gamma \vdash F : El(D) \rightharpoonup El(D')}{\Gamma \vdash \mathsf{If}(\mathsf{Jf}(F)) = F : El(D) \rightharpoonup El(D')}$$

Universal Type Retraction Equations

$$\frac{\Gamma \vdash D : dom}{\Gamma \vdash \mathsf{Ret}(\mathsf{Val}(D)) = D : dom}$$

Substitution and Weakening

There are particularly useful rules concerning substitution and weakening which may be derived from the rules for deducing theorems of $\mathrm{FIX}^{\mu}_{=}$ theories. Indeed, we have the following lemma:

Lemma 8.1.1 The following rules are derivable, where we write J for any one of the expressions e type, $e = e'$, $e : e'$ or $e = e' : e''$.

$$\frac{\chi : \Gamma \to \Gamma' \quad \Gamma' \vdash J(\vec{y})}{\Gamma \vdash J(\chi)} \qquad \frac{\Gamma \vdash \alpha \text{ type} \quad \Gamma, \Gamma' \vdash J}{\Gamma, x : \alpha, \Gamma' \vdash J}$$

$$\frac{\Gamma \vdash M = M' : \alpha \quad \Gamma, x : \alpha, \Gamma'(x) \vdash N(x) = N'(x) : \beta(x)}{\Gamma, \Gamma'(M) \vdash N(M) = N'(M') : \beta(M)}$$

$$\frac{\Gamma \vdash M = M' : \alpha \quad \Gamma, x : \alpha, \Gamma'(x) \vdash \beta(x) = \beta'(x)}{\Gamma, \Gamma'(M) \vdash \beta(M) = \beta'(M')}$$

Proof Proceeds by an induction on the derivation of judgements. □

8.2 Recursive Types via Fixpoint Objects

We have seen how to interpret types of languages as objects in categories. It is well known that the type of natural numbers can be represented as the recursive type $\mu X.X + unit$. If we are interpreting formal typing statements in, say, a category $\mathcal{C}$, then a sensible denotation of such a recursive type would be a solution to the equation $X \cong X + 1$. More generally, recursive types can be thought of as fixpoints of assignments $A \mapsto F(A)$ on objects A. The type is denoted by an object A_0 such that $F(A_0) \cong A_0$ and the operation F will usually be a functor satisfying properties which ensure that A_0 exists [SP82].

The basic categorical notion of a type of types in a category $\mathcal{C}$ is that there is a category object $\mathbb{U}$ in $\mathcal{C}$ which is externally equivalent to $\mathcal{C}$ indexed over itself, i.e. there is an equivalence $\mathcal{C}(-, \mathbb{U}) \simeq \mathcal{C}/-$. With this, an endofunctor F on $\mathcal{C}$ will give rise to an internal functor $\tilde{F} : \mathbb{U} \to \mathbb{U}$. Thus with the above interpretation of recursive types as a fixed point of such endofunctors F we may equivalently solve for a fixed point of $\tilde{F}$. With this motivation, we prove the following proposition:

Proposition 8.2.1 There is an expression of the abstract syntax, Fix, for which given a recursive typing judgement of the form $\Gamma, x : dom \vdash D(x) : dom$ one may derive the judgements

$$\Gamma \vdash \mathsf{Fix}(D) : dom \qquad \text{and} \qquad \Gamma \vdash D(\mathsf{Fix}(D)) = D : dom.$$

Proof Set $\mathsf{Fix} \stackrel{\text{def}}{=} d.\mathsf{It}_{dom}(y.d(\mathsf{Ret}(y)), \sigma(\omega))$. Then the claim is immediate from the $\mathrm{FIX}^{\mu}_{=}$ rules. □

Chapter 9

Categorical Semantics of the $\mathrm{FIX}^{\mu}_{=}$ Logic

9.1 Categories for Modelling Dependent Type Theories

We review a categorical structure which can be used to model dependent type theories. Some of the earliest work in this area was undertaken by Cartmell [Car86] with additional work by Taylor [Tay86]. Here we shall give a presentation of "categories with attributes" which is based on on Pitts' account in [Pit90a]. Further useful information can be found in [Str89], [HP89], [CGW89] and [Ben85].

Categories with Attributes

Definition 9.1.1 A *category with attributes* is specified by a category $\mathcal{C}$ with terminal object (called the *base* category) which is equipped with the following structure:

- For each object X in $\mathcal{C}$, a collection of *fibrations over* X, $\mathrm{Fib}(X)$. We write $\pi_F : X \bullet F \to X$ for the *projection* from the *total object* to the *base object* of F.
- For each morphism $f : Y \to X$ in $\mathcal{C}$ and fibration F over X, there is a fibration f^*F over Y called the *pullback* of F along f for which there is a pullback square in $\mathcal{C}$ of the form

$$\begin{array}{ccc} Y \bullet f^*F & \xrightarrow{f \bullet F} & X \bullet F \\ {\scriptstyle \pi_{f^*F}}\downarrow & & \downarrow{\scriptstyle \pi_F} \\ Y & \xrightarrow{\ f\ } & X \end{array}$$

such that we have

$$id_X^* F = F \qquad id_X \bullet F = id_{X \bullet F}$$
$$g^*(f^*F) = (fg)^*F \qquad (f \bullet F) \circ (g \bullet f^*F) = (fg) \bullet F$$

where $Z \xrightarrow{g} Y \xrightarrow{f} X$.

Notation for Categories with Attributes

It will be convenient to adopt some notational conventions which will be useful when presenting the categorical semantics of dependent type theories.

Given a fibration $F \in \mathrm{Fib}(X)$ we shall write $a \in_X F$ to indicate that a is a section of π_F. Note that given $f: Y \to X$ in $\mathcal{C}$, there is a section $f^*a \in_Y f^*F$ arising from the universal property of pullbacks.

Fibration lists L and their *associated total objects* $\overline{L}$ are defined inductively. The empty list $[\,]$ is a fibration list with $\overline{[\,]} \stackrel{\text{def}}{=} 1$. If L is a fibration list and $F \in \mathrm{Fib}(\overline{L})$, then L, F is a fibration list with $\overline{L, F} \stackrel{\text{def}}{=} \overline{L} \bullet F$.

Section lists l and their *associated morphisms* $\overline{l}$ are defined inductively. The empty list $[\,]$ is a section list $[\,]: L \to [\,]$ (where L is a fibration list) with $\overline{[\,]} \stackrel{\text{def}}{=} ! : \overline{L} \to 1$. If $l: L \to L'$ is a section list, $F' \in \mathrm{Fib}(\overline{L'})$ and $a' \in_{\overline{L}} \overline{l}^* F'$, then $l, a': L \to L', F'$ is a section list with

$$\overline{l, a'} \quad \stackrel{\text{def}}{=} \quad \overline{L} \xrightarrow{a'} \overline{L} \bullet \overline{l}^* F' \xrightarrow{\overline{l} \bullet F'} \overline{L'} \bullet F'.$$

Given a fibration list L, L' the *associated projection morphism* $\pi_{L'}: \overline{L, L'} \to \overline{L}$ is defined by induction on the length of L'. We put

$$\begin{aligned} \pi_{[\,]} &\stackrel{\text{def}}{=} \overline{L} \xrightarrow{id} \overline{L}, \\ \pi_{L', F'} &\stackrel{\text{def}}{=} \overline{L, L', F'} = \overline{L, L'} \bullet F' \xrightarrow{\pi_{F'}} \overline{L, L'} \xrightarrow{\pi_{L'}} \overline{L}. \end{aligned}$$

Finally, we complete our notational conventions with the definition of generic sections. Given a fibration list L, F, L' the *generic section* $\delta(L, F, L') \in_{\overline{L,F,L'}} \pi^*_{F,L'} F$ is defined by appealing to the universal property of the pullback square

$$\begin{array}{ccc} \overline{L, F, L'} \bullet \pi^*_{F,L'} F & \xrightarrow{\pi_{F,L'} \bullet F} & \overline{L} \bullet F \\ \Big\downarrow{\scriptstyle \pi_{\pi^*_{F,L'} F}} \quad \Big\uparrow{\scriptstyle \delta(L,F,L')} & & \Big\downarrow{\scriptstyle \pi_F} \\ \overline{L, F, L'} & \xrightarrow{\pi_{F,L'}} & \overline{L} \end{array}$$

together with the observation that $\pi_{F,L'} = \pi_F \pi_{L'}$.

9.2 FIX Categories with Attributes

We have seen that the logical systems $\mathrm{FIX}_=$ and FIX correspond in a precise way to FIX categories and FIX hyperdoctrines respectively. We shall now define the categorical structure which corresponds to $\mathrm{FIX}^{\mu}_{=}$; such structures will be called FIX categories with attributes. Useful background information can be found in [Pit87] and [Joh77].

Definition 9.2.1 Let $\mathcal{C}$ be a category-with-attributes where for each object X of $\mathcal{C}$, $\mathrm{Fib}(X)$ is regarded as a category with objects the fibrations over X and morphisms

given by $\mathrm{Fib}(X)(F,F') \stackrel{\mathrm{def}}{=} \mathcal{C}/X(\pi_F,\pi_{F'})$. Then $\mathcal{C}$ is a *FIX category-with-attributes* if it satisfies the following conditions:

- Each $\mathrm{Fib}(X)$ is a let-category with (finite products), stable finite coproducts and T-exponentials. For a fibration F over X and a morphism $f: Y \to X$ we have $Tf^*(F) = f^*(TF)$, and the pullback functions preserve the categorical structure of the categories $\mathrm{Fib}(X)$.
- There are distinguished fibrations $\hat{1}$, $\hat{0}$, $\hat{N}$ and $\hat{\Omega}$ over 1 such that the fibrations obtained by pulling back along the unique morphism $!: X \to 1$ (for any object X) are the canonical terminal object, initial object, natural numbers object and fixpoint object of the category $\mathrm{Fib}(X)$.
- There is a specified distinguished fibration $\hat{U}$ over the terminal object for which $\hat{U}$ is a retract of $T\hat{U}$ in $\mathrm{Fib}(1)$. There is a specified distinguished fibration τ over $1\bullet\hat{U}$ which gives rise, via specified *internal type constructor* morphisms

$$\begin{array}{ll} \ulcorner 0 \urcorner : 1 \to 1\bullet\hat{U} & \ulcorner 1 \urcorner : 1 \to 1\bullet\hat{U} \\ \ulcorner T \urcorner : 1\bullet\hat{U} \to 1\bullet\hat{U} & \ulcorner \times \urcorner : 1\bullet\hat{U} \times 1\bullet\hat{U} \to 1\bullet\hat{U} \\ \ulcorner + \urcorner : 1\bullet\hat{U} \times 1\bullet\hat{U} \to 1\bullet\hat{U} & \ulcorner \rightharpoonup \urcorner : 1\bullet\hat{U} \times 1\bullet\hat{U} \to 1\bullet\hat{U} \end{array}$$

  to certain specified canonical *type decoding* isomorphisms, where we write $pr_i : 1\bullet\hat{U} \times 1\bullet\hat{U} \to 1\bullet\hat{U}$ for projection:

  1. $\ulcorner 1 \urcorner^* \tau \cong \hat{1}$ in $\mathrm{Fib}(1)$.
  2. $\ulcorner 0 \urcorner^* \tau \cong \hat{0}$ in $\mathrm{Fib}(1)$.
  3. $\ulcorner T \urcorner^* \tau \cong T\tau$ in $\mathrm{Fib}(1\bullet\hat{U})$.
  4. $\ulcorner \times \urcorner^* \tau \cong pr_1^*\tau \times pr_2^*\tau$ in $\mathrm{Fib}(1\bullet\hat{U} \times 1\bullet\hat{U})$.
  5. $\ulcorner + \urcorner^* \tau \cong pr_1^*\tau + pr_2^*\tau$ in $\mathrm{Fib}(1\bullet\hat{U} \times 1\bullet\hat{U})$.
  6. $\ulcorner \rightharpoonup \urcorner^* \tau \cong pr_1^*\tau \rightharpoonup pr_2^*\tau$ in$\mathrm{Fib}(1\bullet\hat{U} \times 1\bullet\hat{U})$.

The FIX Category with Attributes $\omega\mathcal{C}po$

The results from Chapter 7 will enable us to give a concrete example of a FIX category with attributes. It will be convenient to have the following lemmas:

Lemma 9.2.2 Given a continuous functor $A: X \to \omega\mathcal{C}po^{ep}$ where X is an ωcpo and $\omega\mathcal{C}po^{ep}$ is the category of embedding partial-projection pairs over $\omega\mathcal{C}po$, applying the (covariant) Grothendieck construction to A yields a morphism $\pi_A: \mathbb{G}(A) \to X$ in $\omega\mathcal{C}po$.

Proof By definition,

$$\mathbb{G}(A) \stackrel{\mathrm{def}}{=} \Sigma_{x \in X} A_x = \{(x,U) \mid U \in A_x \;\&\; x \in X\}$$

with a partial order structure given by $(x,U) \leq (x',U')$ iff $x \leq x'$ and $i_{x,x'}(U) \subset U'$ where $i_{x,x'}: A_x \to A_{x'}$ is the embedding determined by the functor A. We omit the details which check that $\mathbb{G}(A)$ is an ωcpo with the above order in which suprema of chains is given by

$$\bigvee\{(x_n,U_n) \mid n \in \omega\} \stackrel{\text{def}}{=} (\bigvee\{x_n \mid n \in \omega\}, \bigvee\{i_{x_n,\bigvee\{x_n \mid n\in\omega\}}(U_n) \mid n \in \omega\}).$$

It is trivial that π_A is continuous. □

Lemma 9.2.3 Let $\mathcal{I}$ be a filtered category, $\mathcal{C}$ a cocomplete category and $D: \mathcal{I}\times\mathcal{I} \to \mathcal{C}$ a functor. Then we have

$$\varinjlim_{I\in\mathcal{I}} \varinjlim_{J\in\mathcal{I}} D(I,J) \cong \varinjlim_{J\in\mathcal{I}} \varinjlim_{I\in\mathcal{I}} D(I,J) \cong \varinjlim_{I\in\mathcal{I}} D(I,I).$$

Proof Routine application of the definition of colimit and filtered category. □

Lemma 9.2.4 Let $A: X \to \omega\mathcal{C}po^{ep}$ be a continuous functor. Let $\{x_n \mid n \in \omega\}$ be a chain in X and write x for the supremum of this chain. Then we have $id_{A_x} = \bigvee\{i_{x_n,x}r_{x_n,x} \mid n \in \omega\}$.

Proof This result follows from unravelling the construction of filtered colimits in the category $\omega\mathcal{C}po^{ep}$. Details of a similar result for the category of embedding projection pairs over the full subcategory of $\omega\mathcal{C}po$ of pointed ωcpo's can be found in [SP82]; the proof for embedding *partial*-projection pairs over $\omega\mathcal{C}po$ is virtually identical. □

Lemma 9.2.5 Let A and B be as in Lemma 9.2.2. Then $f \in \omega\mathcal{C}po/X(\pi_A,\pi_B)$ corresponds to an X-indexed family $(f_x: A_x \to B_x \mid x \in X)$ of continuous functions for which given $x \leq x'$ and a chain $\{x_n \mid n \in \omega\} \subset X$ we have

$$j_{x,x'}f_x \leq f_{x'}i_{x,x'} \tag{9.1}$$

$$f_x = \bigvee\{j_{x_n,x}f_{x_n}r_{x_n,x} \mid n \in \omega\} \text{ where } x \stackrel{\text{def}}{=} \bigvee\{x_n \mid n \in \omega\} \tag{9.2}$$

where we write i and j for embeddings, r for partial-projection. Thus for example, $i_{x,x'}: A_x \to A_{x'}$.

Proof Note that the underlying set-theoretic function of $f \in \omega\mathcal{C}po/X(\pi_A,\pi_B)$ corresponds to a family $(f_x \mid x \in X)$ of set-theoretic functions which have the required form. It is easy to see that f is monotone just in case each f_x is monotone and 9.1 holds. Now take an arbitrary directed set in $\mathbb{G}(A)$ of the form $\{(x_n,U_n) \mid n \in \omega\}$ and assume the continuity (and monotonicity) of f. This leads to the requirement that

$$\bigvee\{j_{x_n,x}f_{x_n}(U_n) \mid n \in \omega\} = f_x(\bigvee\{i_{x_n,x}(U_n) \mid n \in \omega\}) \tag{9.3}$$

where $x \stackrel{\text{def}}{=} \bigvee\{x_n \mid n \in \omega\}$ and the set on the left hand side of equation 9.3 is a chain due to equation 9.1 (itself a consequence of the monotonicity of f). By taking an instance of equation 9.3 in which $x_n = x_{n+1}$ for each n, we see that each f_x is continuous. To see that 9.3 implies 9.2 take a chain $\{x_n \mid n \in \omega\} \subset X$. It is the case that $\{(x_n, r_{x_n,x}(U)) \mid n \in \omega \;\&\; r_{x_n,x}(U)\Downarrow\}$ is a chain in $\mathbb{G}(A)$ where $U \in A_x$. We can apply an instance of equation 9.3 to this directed set and deduce:

$$\begin{aligned} & \bigvee\{j_{x_n,x} f_{x_n}(r_{x_n,x}(U)) \mid n \in \omega \;\&\; r_{x_n,x}(U)\Downarrow\} \\ = \;& f_x(\bigvee\{i_{x_n,x}(r_{x_n,x}(U)) \mid n \in \omega \;\&\; r_{x_n,x}(U)\Downarrow\}) \\ \text{by Lemma 9.2.4} \;= \;& f_x(U). \end{aligned}$$

Conversely, suppose we are given a family $(f_x \mid x \in X)$ of continuous functions which satisfy 9.2. It remains to prove that the corresponding morphism f is continuous, that is 9.3 holds. Indeed we have

$$\begin{aligned} f_x(\bigvee\{i_{x_n,x}(U_n) \mid n \in \omega\}) &= \bigvee\{j_{x_m,x} f_{x_m} r_{x_m,x}(\bigvee\{i_{x_n,x}(U_n) \mid n \in \omega\}) \mid m \in \omega\} \\ &= \bigvee\{\bigvee\{j_{x_m,x} f_{x_m} r_{x_m,x} i_{x_n,x}(U_n) \mid n \in \omega\} \mid m \in \omega\} \\ \text{by Lemma 9.2.3} \;&= \bigvee\{j_{x_n,x} f_{x_n} r_{x_n,x} i_{x_n,x}(U_n) \mid n \in \omega\} \\ &= \bigvee\{j_{x_n,x} f_{x_n}(U_n) \mid n \in \omega\}. \end{aligned}$$

□

We can now prove the main result of this section, namely

Proposition 9.2.6 The category $\omega\mathcal{C}po$ is a FIX category with attributes.

Proof The base category is of course $\omega\mathcal{C}po$ and this certainly has a terminal object. The collection of fibrations over a Scott predomain X is given by the collection of continuous functors of the form $A\colon X \to \omega\mathcal{C}po^{ep}$. Recall Lemma 9.2.2: Given a fibration A over X the total object and projection are given by $\pi_A\colon \mathbb{G}(A) \to X$. If $f\colon Y \to X$ is a morphism in $\omega\mathcal{C}po$ we set $f^*A \stackrel{\text{def}}{=} Af$. It is simple to check that

$$\mathbb{G}(f^*A) \stackrel{\text{def}}{=} \Sigma_{y \in Y} Af_y \cong Y \times_X \mathbb{G}(A)$$

and hence that we get a pullback square in $\omega\mathcal{C}po$ of the required form, where $f\bullet A\colon \mathbb{G}(f^*A) \to \mathbb{G}(A)$ is given by $f\bullet A(y, V) \stackrel{\text{def}}{=} (f(y), V)$. The functoriality conditions are satisfied, and we have shown that $\omega\mathcal{C}po$ is a category with attributes.

Now we need to see that each $\mathrm{Fib}(X)$ is a let category with (finite products), stable finite coproducts and T-exponentials. This structure arises pointwise. For example, given fibrations A and B, their product object $A \times B$ is defined by setting

$$A \times B(x \leq x') \stackrel{\text{def}}{=} i_{x,x'} \times j_{x,x'}\colon A_x \times B_x \leftrightarrows A_{x'} \times B_{x'}\colon r_{x,x'} \times s_{x,x'}$$

and (for example) the projection morphism on A is specified by the family of projections $(\pi_x\colon A_x \times B_x \to A_x \mid x \in X)$, which are easily checked to satisfy Lemma 9.2.5.

We sketch the details of the let category structure, which arises by pointwise application of the properties of the let category $\omega\mathcal{C}po$ (with the lifting monad): The operation on objects is given by $A \mapsto \perp A$, where $\perp A$ is defined by

$$\perp A(x \leq x') \stackrel{\text{def}}{=} (i_{x,x'})_\perp : (A_x)_\perp \leftrightarrows (A_{x'})_\perp : (r_{x,x'})_\perp .$$

Morphisms $\eta_A : A \to \perp A$ are specified by the continuous family $((\eta_A)_x : A_x \to (A_x)_\perp \mid x \in X)$ with $(\eta_A)_x(U) \stackrel{\text{def}}{=} [U]$. Given $f : A \times B \to C_\perp$ we define $\mathit{lift}(f) : A \times B_\perp \to C_\perp$ by specifying the family $(\mathit{lift}(f_x) : A_x \times (B_x)_\perp \to (C_x)_\perp \mid x \in X)$. We omit the remaining details, but note that the structure preserving conditions are immediate.

Now we need to define the fibrations $\hat{1}$, $\hat{0}$, $\hat{N}$ and $\hat{\Omega}$. Let us write $\{*\}$ for the terminal object of $\omega\mathcal{C}po$. Then set $\hat{1}(*) \stackrel{\text{def}}{=} 1$, $\hat{0}(*) \stackrel{\text{def}}{=} 0$, $\hat{N}(*) \stackrel{\text{def}}{=} \mathbb{N}$ and $\hat{\Omega}(*) \stackrel{\text{def}}{=} \Omega$ where $\mathbb{N}$ and Ω are the NNO and FPO of $\omega\mathcal{C}po$. We consider the details for the FPO in $\mathrm{Fib}(X)$. Thus we need to see the existence of $\omega : \hat{1}! \to \perp\hat{\Omega}!$ and $\sigma : \perp\hat{\Omega}! \to \hat{\Omega}!$ which give rise to a FPO $\hat{\Omega}!$ in $\mathrm{Fib}(X)$. Using Lemma 9.2.5 we need to define a family $(\omega_x : \hat{1}!_x \to \perp\hat{\Omega}!_x \mid x \in X)$ that is $(\omega_x : 1 \to \Omega_\perp \mid x \in X)$ and also $(\sigma_x : \perp\hat{\Omega}!_x \to \hat{\Omega}!_x \mid x \in X)$ that is $(\sigma_x : \Omega_\perp \to \Omega \mid x \in X)$. We take these to be the constant families given by the FPO (Ω, σ, ω) of $\omega\mathcal{C}po$. We omit to check the equaliser diagram requirement; however we shall give all the necessary details for the universal property of the FPO. Thus given a morphism $f : \perp A \to A$ in $\mathrm{Fib}(X)$ we need a unique $h : \hat{\Omega}! \to A$ for which $\mathit{flet}(\eta h) = h\sigma$. Using Lemma 9.2.5 this amounts to defining a unique continuous family $(h_x \mid x \in X)$ which satisfies equations 9.1 and 9.2 and for which the following diagram commutes:

$$\begin{array}{ccc} \Omega_\perp & \xrightarrow{\sigma} & \Omega \\ {\scriptstyle (h_x)_\perp}\downarrow & (*) & \downarrow{\scriptstyle h_x} \\ (A_x)_\perp & \xrightarrow{f_x} & A_x \end{array}$$

The existence of a candidate for the family $(h_x \mid x \in X)$ is immediate from the existence of a FPO Ω in $\omega\mathcal{C}po$. We need to check that the h_x satisfy the conditions 9.1 and 9.2.

(*Satisfaction of 9.1*): Take $x \leq x'$ and consider the diagram

$$\begin{array}{ccc} \hat{\Omega}!_x & \xrightarrow{id} & \hat{\Omega}!_{x'} \\ {\scriptstyle h_x}\downarrow & & \downarrow{\scriptstyle h_{x'}} \\ A_x & \xrightarrow{j_{x,x'}} & A_{x'} \end{array}$$

together with $i_{x,x'} : (A_x)_\perp \to (A_{x'})_\perp$. Condition 9.1 demands that $j_{x,x'} h_x \leq h_{x'}\, id$. Indeed,

$$j_{x,x'} h_x \;=\; j_{x,x'} h_x \sigma\sigma^{-1}$$

$$
\begin{aligned}
&= j_{x,x'} f_x (h_x)_\perp \sigma^{-1} \\
&\leq f_{x'} i_{x,x'} (h_x)_\perp \sigma^{-1} \\
&= f_{x'} (j_{x,x'})_\perp (h_x)_\perp \sigma^{-1} \\
&= f_{x'} (h_{x'})_\perp \sigma^{-1} \\
&= h_{x'}.
\end{aligned}
$$

(*Satisfaction of 9.2*): Take a chain $\{x_n \mid n \in \omega\}$ in X and set $x = \bigvee \{x_n \mid n \in \omega\}$. Put $j_{x_n,x}: A_{x_n} \to A_x$, $j'_{x_n,x}: (A_{x_n})_\perp \to (A_x)_\perp$ and $r_{x_n,x}: (A_x)_\perp \to (A_{x_n})_\perp$. Then

$$
\begin{aligned}
(\bigvee \{j_{x_n,x} h_{x_n} \mid n \in \omega\})\sigma &= \bigvee \{j_{x_n,x} h_{x_n} \sigma \mid n \in \omega\} \\
&= \bigvee \{j_{x_n,x} f_{x_n} (h_{x_n})_\perp \mid n \in \omega\} \\
&= \bigvee \{j_{x_n,x} f_{x_n} r_{x_n,x} j'_{x_n,x} (h_{x_n})_\perp \mid n \in \omega\} \\
\text{by Lemma 9.2.3} \quad &= \bigvee \{j_{x_n,x} f_{x_n} r_{x_n,x} \mid n \in \omega\} \bigvee \{j'_{x_n,x} (h_{x_n})_\perp \mid n \in \omega\} \\
&= f_x \bigvee \{(j_{x_n,x})_\perp (h_{x_n})_\perp \mid n \in \omega\} \\
&= f_x \bigvee \{(j_{x_n,x} h_{x_n})_\perp \mid n \in \omega\} \\
&= f_x (\bigvee \{j_{x_n,x} h_{x_n} \mid n \in \omega\})_\perp.
\end{aligned}
$$

Appealing to the universal property of the FPO in $\omega\mathcal{C}po$, it must be the case that $h_x = \bigvee \{j_{x_n,x} h_{x_n}\, id \mid n \in \omega\}$ which is what we wanted to prove.

We define the fibration $\hat{U}: 1 \to \omega\mathcal{C}po^{ep}$ by setting $\hat{U}(*) \stackrel{\text{def}}{=} PS_{\mathbb{N}}$. The retract condition in Fib(1) is immediate from Lemma 7.6.3. Of course $1 \bullet \hat{U} = 1 \times PS_{\mathbb{N}}$, but we shall take this to be just $PS_{\mathbb{N}}$ in the remainder of this proof. The continuous functor $\tau: PS_{\mathbb{N}} \to \omega\mathcal{C}po^{ep}$ is given by using Lemma 7.6.5 and regarding $\mathcal{S}pd_\omega^{ep}$ as a full subcategory of $\omega\mathcal{C}po^{ep}$. It remains to verify the existence of certain canonical isomorphisms; we only verify $\ulcorner \multimap \urcorner^* \tau \cong pr_1^* \tau \to (pr_2^* \tau)_\perp$ in $\mathrm{Fib}(PS_{\mathbb{N}} \times PS_{\mathbb{N}})$ where $pr_i: PS_{\mathbb{N}} \times PS_{\mathbb{N}} \to PS_{\mathbb{N}}$. Unravelling the definitions, we have to see that we have a diagram of the form

$$
\begin{array}{ccc}
\Sigma_{(A,B) \in PS_{\mathbb{N}}} |A^{\ulcorner \multimap \urcorner} B| & \cong & \Sigma_{(A,B) \in PS_{\mathbb{N}}} |A| \to |B|_\perp \\
& {\scriptstyle \pi_{\ulcorner \multimap \urcorner^* \tau}} \searrow \quad \swarrow {\scriptstyle \pi_{pr_1^* \tau \to (pr_2^* \tau)_\perp}} & \\
& PS_{\mathbb{N}}^2 &
\end{array}
$$

To do this we define a family of isomorphisms in $\omega\mathcal{C}po$ which satisfy Lemma 9.2.5:

$$
(\varphi_{(A,B)} : |A^{\ulcorner \multimap \urcorner} B| \cong |A| \to |B|_\perp : \vartheta_{(A,B)} \mid (A,B) \in PS_{\mathbb{N}}^2)
$$

and to do this we use Lemma 7.6.5 (Lemma 7.5.3). We shall only check that the morphism $\varphi_{(A,B)}$ satisfies equation 9.2. Take a chain $\{(A_n, B_n) \mid n \in \omega\}$ in $PS_{\mathbb{N}}^2$ and set (A,B) to be the supremum. Unravelling the details, if we write

$$
\begin{array}{ll}
r_n: |A^{\ulcorner \multimap \urcorner} B| \to |A_n{}^{\ulcorner \multimap \urcorner} B_n| & \varphi_n: |A_n{}^{\ulcorner \multimap \urcorner} B_n| \to |A_n| \to |B_n|_\perp \\
s_n: |A| \to |A_n| & (i_n)_\perp: |B_n|_\perp \to |B|_\perp
\end{array}
$$

for the obvious morphisms, and set $F_n \stackrel{\text{def}}{=} A_n \ulcorner\!\rightharpoonup\!\urcorner B_n$ then explicitly we have

$$r_n(V) \stackrel{\text{def}}{=} \begin{cases} F_n \cap V \text{ if this is non-empty} \\ \text{undefined otherwise} \end{cases}$$

$$\varphi_n(V)(U) \stackrel{\text{def}}{=} \begin{cases} \{b \mid \exists X \subset^f U. X \blacktriangleright b \in V\} \text{ if this is non-empty} \\ \text{else } \bot \end{cases}$$

$$(i_n)_\bot \circ f \circ s_n(U) \stackrel{\text{def}}{=} \begin{cases} \text{undefined if } A_n \cap U \text{ is non-empty, else} \\ \bot \text{ if } f(A_n \cap U) \text{ is } \bot \text{ else} \\ \{b \mid \exists Y \subset^f f(A_n \cap U). Y \vdash_B b\} \end{cases}$$

and we need to prove that $\varphi = \bigvee\{i_n \varphi_n r_n \mid n \in \omega\}$. In order to show this we introduce some notation. Put

$$\begin{aligned} F &\stackrel{\text{def}}{=} A \ulcorner\!\rightharpoonup\!\urcorner B \\ e_n &\stackrel{\text{def}}{=} (i_n)_\bot \circ \varphi_n(r_n(V)) \circ s_n(U) \\ S &\stackrel{\text{def}}{=} \varphi(V)(U) \\ T &\stackrel{\text{def}}{=} \bigvee\{e_n \mid e_n \Downarrow\}. \end{aligned}$$

Then we need to show that $S = T$ where

$$e_n = \begin{cases} \text{undefined if } F_n \cap V = \emptyset \text{ or } A_n \cap U = \emptyset \text{ else} \\ \bot \text{ if } \{b \mid \exists X \subset^f A_n \cap U. X \blacktriangleright b \in F_n \cap V\} = \emptyset \text{ else} \\ W_n \stackrel{\text{def}}{=} \{b \mid \exists Y \subset^f \{b \mid \exists X \subset^f A_n \cap U. X \blacktriangleright b \in F_n \cap V\}. Y \vdash_B b\} \end{cases}$$

We begin by noting that $e_n \Downarrow$ for at least one n. Suppose that $S = \bot$. Then whenever $e_n \Downarrow$ we must have $e_n = \bot$. Therefore $T = \bigvee\{\bot \mid e_n \Downarrow\} = \bot$. Now suppose that $T = \bot$. Then whenever $e_n \Downarrow$ there can be no $X \subset^f A_n \cap U$ and $b \in B_n$ for which $X \blacktriangleright b \in F_n \cap V$. Suppose for a contradiction that S is not $\bot$, which is to say there is $X \subset^f U$ and $b \in B$ for which $X \blacktriangleright b \in V$. Then we must have $X \blacktriangleright b \in F_{n_0}$ for some n_0 which implies $X \subset^f A_{n_0}$ and $b \in B_{n_0}$. These data imply that $e_{n_0} \Downarrow$, which is contradictory. Hence $S = \bot$.

It follows that S is not bottom just in case T is not bottom. Thus it remains to show if S is not bottom then $S = \bigcup\{W_n \mid n \in \omega\}$. Let $b \in S$. Then there is some $X \subset^f U$ for which $X \blacktriangleright b \in V$ and so $X \blacktriangleright b \in F_{n_0} \cap V$ for some n_0. This implies $X \downarrow_{A_{n_0}}$ and so $X \subset^f A_{n_0} \cap U$. Trivially $\{b\} \vdash_B b$ and so $b \in W_{n_0}$. Conversely, suppose that $b \in W_{n_0}$ for some n_0. Then there are X, Y for which $Y \vdash_B b$, $X \subset^f A_{n_0} \cap U$, and $X \blacktriangleright y \in F_{n_0} \cap V$ for each $y \in Y$. Hence $b \in B_{n_0}$ and $\bigcup\{y \mid X \vdash_{A_{n_0}} X\} \vdash_{B_{n_0}} b$ which implies $\{X \blacktriangleright y \mid y \in Y\} \vdash_{F_{n_0}} X \blacktriangleright b$ and hence $X \blacktriangleright b \in V$. Therefore $b \in S$.

We leave all remaining details of the type coding isomorphisms to the reader; with this, the proof is complete. □

9.3 Categorical Semantics of $\mathrm{FIX}^{\mu}_{=}$

Structures for $\mathrm{FIX}^{\mu}_{=}$ Signatures

A *structure* $\mathbf{M}$ for a $\mathrm{FIX}^{\mu}_{=}$ signature Sg in FIX category with attributes is specified by:

- For each basic type valued function symbol t, a fibration list L_t and a fibration $F_t \in \mathrm{Fib}(\overline{L_t})$.
- For each basic term valued function symbol f, a fibration list L_f, a fibration $F_f \in \mathrm{Fib}(\overline{L_f})$ and a section $a_f \in_{\overline{L_f}} F_f$.

It is assumed that the length of the fibration lists match the arities of the function symbols.

Interpretation of $\mathrm{FIX}^{\mu}_{=}$ Expressions

Let us suppose that we have a structure $\mathbf{M}$ in a FIX category with attributes $\mathcal{C}$ for a $\mathrm{FIX}^{\mu}_{=}$ signature Sg. We define relations between the forms of judgement given on Page 134 and appropriate structure in $\mathcal{C}$. These relations are of the following kinds where it is assumed that the $\mathrm{FIX}^{\mu}_{=}$ judgements are well formed:

1. $[\![\Gamma \text{ ctxt}]\!] \blacktriangleright L$ where L is a fibration list with $\mathrm{LEN}(L) = \mathrm{LEN}(\Gamma)$.
2. $[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F$ where $\mathrm{LEN}(L) = \mathrm{LEN}(\Gamma)$ and $F \in \mathrm{Fib}(\overline{L})$.
3. $[\![\Gamma \vdash M{:}\alpha]\!] \blacktriangleright L \vdash a : F$ where $\mathrm{LEN}(L) = \mathrm{LEN}(\Gamma)$, $F \in \mathrm{Fib}(\overline{L})$ and $a \in_{\overline{L}} F$.
4. $[\![\chi{:}\Gamma \to \Gamma']\!] \blacktriangleright l{:}L \to L'$ where $\mathrm{LEN}(L) = \mathrm{LEN}(\Gamma)$ and $l{:}L \to L'$ is a section list with $\mathrm{LEN}(l) = \mathrm{LEN}(\chi)$.

The relations are defined inductively by the following rules:

- **Contexts**

$$\frac{}{[\![[\,] \text{ ctxt}]\!] \blacktriangleright [\,]} \qquad \frac{[\![\Gamma \text{ ctxt}]\!] \blacktriangleright L \quad [\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F}{[\![\Gamma, x{:}\alpha \text{ ctxt}]\!] \blacktriangleright L, F}$$

- **Types**

$$\frac{[\![\chi{:}\Gamma \to \Gamma_t]\!] \blacktriangleright l{:}L \to L_t}{[\![\Gamma \vdash t(\chi) \text{ type}]\!] \blacktriangleright L \vdash \overline{l}^* F_t}$$

- **Terms**

$$\frac{\begin{cases} [\![\Gamma, x{:}\alpha, \Gamma' \vdash \alpha \text{ type}]\!] \blacktriangleright L, F, L' \vdash \pi^*_{F,L'} F \\ [\![\Gamma \text{ ctxt}]\!] \blacktriangleright L \quad [\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F \quad [\![\Gamma, x{:}\alpha, \Gamma' \text{ ctxt}]\!] \blacktriangleright L, F, L' \end{cases}}{[\![\Gamma, x{:}\alpha, \Gamma' \vdash x{:}\alpha]\!] \blacktriangleright L, F, L' \vdash \delta(L, F, L'){:}\pi^*_{F,L'} F}$$

$$\frac{[\![\chi{:}\Gamma \to \Gamma_f]\!] \blacktriangleright l{:}L \to L_f \quad [\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash \overline{l}^* F_f}{[\![\Gamma \vdash f(\chi){:}\alpha]\!] \blacktriangleright L \vdash \overline{l}^* a_f{:}\overline{l}^* F_f}$$

- **Term Lists**

$$\frac{[\![\Gamma \text{ ctxt}]\!] \blacktriangleright L}{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}$$

$$\frac{\left\{\begin{array}{l} [\![\Gamma \vdash M\colon \alpha'(\chi)]\!] \blacktriangleright L \vdash a\colon \vec{l}^{*} F' \\ [\![\chi\colon \Gamma \to \Gamma']\!] \blacktriangleright l\colon L \to L' \quad [\![\Gamma' \vdash \alpha'(\vec{y}) \text{ type}]\!] \blacktriangleright L' \vdash F' \end{array}\right.}{[\![\chi, M\colon \Gamma \to \Gamma', y\colon \alpha'(\vec{y})]\!] \blacktriangleright l, a\colon L \to L', F'}$$

- **Elementary External Types**

$$\frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \mathit{unit} \text{ type}]\!] \blacktriangleright L \vdash !^{*}\hat{1}} \qquad \frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \mathit{null} \text{ type}]\!] \blacktriangleright L \vdash !^{*}\hat{0}} \qquad \frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \mathit{nat} \text{ type}]\!] \blacktriangleright L \vdash !^{*}\hat{N}}$$

$$\frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \mathit{fix} \text{ type}]\!] \blacktriangleright L \vdash !^{*}\hat{\Omega}} \qquad \frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \mathit{dom} \text{ type}]\!] \blacktriangleright L \vdash !^{*}\hat{U}}$$

where $!\colon \overline{L} \to 1$.

- **External Binary Products**

$$\frac{[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F \quad [\![\Gamma \vdash \beta \text{ type}]\!] \blacktriangleright L \vdash F'}{[\![\Gamma \vdash \alpha \times \beta \text{ type}]\!] \blacktriangleright L \vdash F \times F'}$$

- **External Binary Coproducts**

$$\frac{[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F \quad [\![\Gamma \vdash \beta \text{ type}]\!] \blacktriangleright L \vdash F'}{[\![\Gamma \vdash \alpha + \beta \text{ type}]\!] \blacktriangleright L \vdash F + F'}$$

- **External T-Exponentials**

$$\frac{[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F \quad [\![\Gamma \vdash \beta \text{ type}]\!] \blacktriangleright L \vdash F'}{[\![\Gamma \vdash \alpha \multimap \beta \text{ type}]\!] \blacktriangleright L \vdash F \multimap F'}$$

- **External Computations**

$$\frac{[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F}{[\![\Gamma \vdash T\alpha \text{ type}]\!] \blacktriangleright L \vdash TF}$$

- **External Decoding**

$$\frac{[\![\Gamma \vdash D\colon \mathit{dom}]\!] \blacktriangleright L \vdash \langle id, d\rangle\colon !^{*}(\hat{U})}{[\![\Gamma \vdash El(D) \text{ type}]\!] \blacktriangleright L \vdash d^{*}\tau}$$

where $!\colon \overline{L} \to 1$ and $d\colon \overline{L} \to 1 \bullet \hat{U}$.

- **Unit and Null Terms**

$$\frac{[\![\Gamma \text{ ctxt}]\!] \blacktriangleright L}{[\![\Gamma \vdash \langle\rangle\colon \mathit{unit}]\!] \blacktriangleright L \vdash \langle id, \iota!\rangle\colon \hat{1}} \qquad \frac{[\![\Gamma \vdash M\colon]\!] \blacktriangleright L \vdash a\colon \hat{0}}{[\![\Gamma \vdash \{\}_{\alpha}(M)\colon \alpha]\!] \blacktriangleright L \vdash !\kappa a\colon F}$$

where $!\colon \overline{L} \to 1$, $\iota\colon 1 \cong 1\bullet\hat{1}$, $\kappa\colon 1\bullet\hat{0} \cong 0$ and $!\colon 0 \to \overline{L}\bullet F$.

- **Binary Product Terms**

$$\frac{[\![\Gamma \vdash M\colon \alpha]\!] \blacktriangleright L \vdash a\colon F \quad [\![\Gamma \vdash N\colon \beta]\!] \blacktriangleright L \vdash a'\colon F'}{[\![\Gamma \vdash \langle M, N\rangle\colon \alpha \times \beta]\!] \blacktriangleright L \vdash \langle a, a'\rangle\colon F \times F'}$$

$$\frac{[\![\Gamma \vdash P\colon \alpha \times \beta]\!] \blacktriangleright L \vdash a\colon F \times F'}{[\![\Gamma \vdash \mathsf{Fst}(P)\colon \alpha]\!] \blacktriangleright L \vdash \pi_1 a\colon F} \qquad \frac{[\![\Gamma \vdash P\colon \alpha \times \beta]\!] \blacktriangleright L \vdash a\colon F \times F'}{[\![\Gamma \vdash \mathsf{Snd}(P)\colon \beta]\!] \blacktriangleright L \vdash \pi_2 a'\colon F'}$$

where π_1 and π_2 are the projections arising from the product $F \times F'$ in the category $\mathrm{Fib}(\overline{L})$.

- **Binary Coproduct Terms**

$$\frac{[\![\Gamma \vdash M\colon \alpha]\!] \blacktriangleright L \vdash a\colon F}{[\![\Gamma \vdash \mathsf{Inl}_\beta(M)\colon \alpha + \beta]\!] \blacktriangleright L \vdash i a\colon F + F'} \qquad \frac{[\![\Gamma \vdash N\colon \beta]\!] \blacktriangleright L \vdash a'\colon F'}{[\![\Gamma \vdash \mathsf{Inl}_\alpha(N)\colon \alpha + \beta]\!] \blacktriangleright L \vdash j a'\colon F + F'}$$

$$\frac{\left\{\begin{array}{l} [\![\Gamma, x\colon \alpha \vdash F(x)\colon \gamma]\!] \blacktriangleright L, F \vdash a\colon \pi_F^* F'' \\ [\![\Gamma, y\colon \beta \vdash G(y)\colon \gamma]\!] \blacktriangleright L, F' \vdash a'\colon \pi_{F'}^* F'' \\ [\![\Gamma \vdash C\colon \alpha + \beta]\!] \blacktriangleright L \vdash a''\colon F + F' \end{array}\right.}{[\![\Gamma \vdash \{F, G\}(C)\colon \gamma]\!] \blacktriangleright L \vdash [\pi_F\bullet F'' \circ a, \pi_{F'}\bullet F'' \circ a']a''\colon F''}$$

where we note that $a\colon \overline{L}\bullet F \to (\overline{L}\bullet F)\bullet(\pi_F^* F'')$, $a'\colon \overline{L}\bullet F' \to (\overline{L}\bullet F')\bullet(\pi_{F'}^* F'')$ and $[-,+]$ denotes the abstraction of unique mediating morphisms arising from the coproduct $F + F'$ in $\mathrm{Fib}(\overline{L})$.

- **T-Exponential Terms**

$$\frac{[\![\Gamma \vdash F\colon \alpha \multimap \beta]\!] \blacktriangleright L \vdash a\colon F \multimap F' \quad [\![\Gamma \vdash M\colon \alpha]\!] \blacktriangleright L \vdash a'\colon F}{[\![\Gamma \vdash FM\colon T\beta]\!] \blacktriangleright ap\langle a, a'\rangle\colon TF'}$$

$$\frac{[\![\Gamma, x\colon \alpha \vdash F(x)\colon T\beta]\!] \blacktriangleright L, F \vdash a\colon T\pi_F^* F'}{[\![\Gamma \vdash \lambda_\alpha(F)\colon \alpha \multimap \beta]\!] \blacktriangleright L \vdash cur(\pi_F\bullet TF' \circ a \circ pr)\iota\colon F \multimap F'}$$

where we recall that $\pi_F^* TF' = T\pi_F^* F'$ by definition and we have $\iota\colon \overline{L} \cong \overline{L}\bullet!^*\hat{1}$ and $pr\colon \overline{L}\bullet!^*\hat{1} \times \overline{L}\bullet F \to \overline{L}\bullet F$.

- **Computation Terms**

$$\frac{[\![\Gamma \vdash M\colon \alpha]\!] \blacktriangleright L \vdash a\colon F}{[\![\Gamma \vdash \mathsf{Val}(M)\colon T\alpha]\!] \blacktriangleright L \vdash \eta a\colon TF}$$

$$\frac{[\![\Gamma \vdash E\colon T\alpha]\!] \blacktriangleright L \vdash a\colon TF \quad [\![\Gamma, x\colon \alpha \vdash F(x)\colon T\beta]\!] \blacktriangleright L, F \vdash a'\colon T\pi_F^* F'}{[\![\Gamma \vdash \mathsf{Let}\,(E, F)\colon T\beta]\!] \blacktriangleright L \vdash let(\pi_F\bullet TF' \circ a') \circ a\colon TF'}$$

- **Natural Number Terms**

$$\frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash 0\colon nat]\!] \blacktriangleright L \vdash !^*0\colon !^*\hat{N}} \qquad \frac{[\![\Gamma \vdash N\colon nat]\!] \blacktriangleright L \vdash \langle id, n\rangle\colon !^*\hat{N}}{[\![\Gamma \vdash \mathsf{Suc}(N)\colon nat]\!] \blacktriangleright L \vdash \langle id, sn\rangle\colon !^*\hat{N}}$$

$$\frac{\left\{\begin{array}{l} [\![\Gamma \vdash M\colon \alpha]\!] \blacktriangleright L \vdash a\colon F \\ [\![\Gamma, x\colon \alpha \vdash F(x)\colon \alpha]\!] \blacktriangleright L, F \vdash a'\colon \pi_F^* F \quad [\![\Gamma \vdash N\colon nat]\!] \blacktriangleright L \vdash \langle id, n\rangle\colon !^*\hat{N} \end{array}\right.}{[\![\Gamma \vdash F^N(M)\colon \alpha]\!] \blacktriangleright L \vdash g\langle id, n\rangle\colon F}$$

where $!\colon \overline{L} \to 1$, $0\colon 1 \to 1 \bullet \hat{N}$, $s\colon 1 \bullet \hat{N} \to 1 \bullet \hat{N}$ and g is the unique morphism arising from the universal property of the NNO $!^* \hat{N} \in \mathrm{Fib}(\overline{L})$:

$$\begin{array}{ccccc} \overline{L} \bullet !^* \hat{1} & \xrightarrow{id \times 0} & \overline{L} \bullet !^* \hat{N} & \xrightarrow{id \times s} & \overline{L} \bullet !^* \hat{N} \\ \downarrow{\scriptstyle pr_{\overline{L}}} & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle g} \\ \overline{L} & \xrightarrow{a} & \overline{L} \bullet F & \xrightarrow{a'} & \overline{L} \bullet F \end{array}$$

- **Fixpoint Terms**

$$\frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \omega\colon Tfix]\!] \blacktriangleright L \vdash !^* \omega\colon !^* T \hat{\Omega}} \qquad \frac{[\![\Gamma \vdash E\colon Tfix]\!] \blacktriangleright L \vdash \langle id, e \rangle\colon T !^* \hat{\Omega}}{[\![\Gamma \vdash \sigma(E)\colon fix]\!] \blacktriangleright L \vdash \langle id, \sigma e \rangle\colon !^* \hat{\Omega}}$$

$$\frac{[\![\Gamma \vdash N\colon fix]\!] \blacktriangleright L \vdash \langle id, n \rangle\colon !^* \hat{\Omega} \quad [\![\Gamma, x\colon T\alpha \vdash F(x)\colon \alpha]\!] \blacktriangleright L, TF \vdash a\colon \pi_{TF}^* F}{[\![\Gamma \vdash \mathsf{It}_\alpha(F, N)\colon \alpha]\!] \blacktriangleright L \vdash g\langle id, n \rangle\colon F}$$

where $!\colon \overline{L} \to 1$ and g is the unique morphism arising from the universal property of the FPO $!^* \hat{\Omega} \in \mathrm{Fib}(\overline{L})$:

$$\begin{array}{ccc} \overline{L} \bullet T !^* \hat{\Omega} & \xrightarrow{id \times \sigma} & \overline{L} \bullet !^* \hat{\Omega} \\ \downarrow{\scriptstyle let(\eta g)} & & \downarrow{\scriptstyle g} \\ \overline{L} \bullet TF & \xrightarrow[a]{} & \overline{L} \bullet F \end{array}$$

where $\eta g\colon \overline{L} \bullet !^* \hat{\Omega} \to \overline{L} \bullet F \to \overline{L} \bullet TF$.

- **Internal Elementary Types**

$$\frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \ulcorner unit \urcorner\colon dom]\!] \blacktriangleright L \vdash !^* \ulcorner 0 \urcorner\colon !^* \hat{U}} \qquad \frac{[\![[\,]\colon \Gamma \to [\,]]\!] \blacktriangleright [\,]\colon L \to [\,]}{[\![\Gamma \vdash \ulcorner null \urcorner\colon dom]\!] \blacktriangleright L \vdash !^* \ulcorner 1 \urcorner\colon !^* \hat{U}}$$

where $!\colon \overline{L} \to 1$.

- **Internal Binary Product**

$$\frac{[\![\Gamma \vdash D\colon dom]\!] \blacktriangleright L \vdash \langle id, d \rangle\colon !^* \hat{U} \quad [\![\Gamma \vdash D'\colon dom]\!] \blacktriangleright L \vdash \langle id, d' \rangle\colon !^* \hat{U}}{[\![\Gamma \vdash D \ulcorner \times \urcorner D'\colon dom]\!] \blacktriangleright L \vdash \langle id, \ulcorner \times \urcorner \langle d, d' \rangle \rangle\colon !^* \hat{U}}$$

- **Internal Binary Coproduct**

$$\frac{[\![\Gamma \vdash D\colon dom]\!] \blacktriangleright L \vdash \langle id, d \rangle\colon !^* \hat{U} \quad [\![\Gamma \vdash D'\colon dom]\!] \blacktriangleright L \vdash \langle id, d' \rangle\colon !^* \hat{U}}{[\![\Gamma \vdash D \ulcorner + \urcorner D'\colon dom]\!] \blacktriangleright L \vdash \langle id, \ulcorner + \urcorner \langle d, d' \rangle \rangle\colon !^* \hat{U}}$$

- **Internal T-Exponential**

$$\frac{[\![\Gamma \vdash D\colon dom]\!] \blacktriangleright L \vdash \langle id, d\rangle\colon !^*\hat{U} \quad [\![\Gamma \vdash D'\colon dom]\!] \blacktriangleright L \vdash \langle id, d'\rangle\colon !^*\hat{U}}{[\![\Gamma \vdash D^\ulcorner\!\multimap\!^\urcorner D'\colon dom]\!] \blacktriangleright L \vdash \langle id, {}^\ulcorner\!\multimap\!^\urcorner\langle d, d'\rangle\rangle\colon !^*\hat{U}}$$

- **Internal Computation**

$$\frac{[\![\Gamma \vdash D\colon dom]\!] \blacktriangleright L \vdash \langle id, d\rangle\colon !^*\hat{U}}{[\![\Gamma \vdash {}^\ulcorner T^\urcorner D\colon dom]\!] \blacktriangleright L \vdash \langle id, {}^\ulcorner T^\urcorner d\rangle\colon !^*\hat{U}}$$

- **Product Externalisation**

$$\frac{[\![\Gamma \vdash M\colon El(D^\ulcorner\!\times\!^\urcorner D')]\!] \blacktriangleright L \vdash a\colon ({}^\ulcorner\!\times\!^\urcorner\langle d, d'\rangle)^*\tau}{[\![\Gamma \vdash \mathsf{Ip}(M)\colon El(D) \times El(D')]\!] \blacktriangleright L \vdash \rho a\colon d^*\tau \times d'^*\tau}$$
$$\frac{[\![\Gamma \vdash P\colon El(D) \times El(D')]\!] \blacktriangleright L \vdash a\colon d^*\tau \times d'^*\tau}{[\![\Gamma \vdash \mathsf{Jp}(P)\colon El(D^\ulcorner\!\times\!^\urcorner D')]\!] \blacktriangleright L \vdash \rho^{-1}a\colon ({}^\ulcorner\!\times\!^\urcorner\langle d, d'\rangle)^*\tau}$$

where $\rho\colon \overline{L}\bullet({}^\ulcorner\!\times\!^\urcorner\langle d, d'\rangle)^*\tau \cong \overline{L}\bullet(d^*\tau \times d'^*\tau)$.

- **Coproduct Externalisation**

$$\frac{[\![\Gamma \vdash M\colon El(D^\ulcorner\!+\!^\urcorner D')]\!] \blacktriangleright L \vdash a\colon ({}^\ulcorner\!+\!^\urcorner\langle d, d'\rangle)^*\tau}{[\![\Gamma \vdash \mathsf{Ic}(M)\colon El(D) + El(D')]\!] \blacktriangleright L \vdash \rho a\colon d^*\tau + d'^*\tau}$$
$$\frac{[\![\Gamma \vdash C\colon El(D) + El(D')]\!] \blacktriangleright L \vdash a\colon d^*\tau + d'^*\tau}{[\![\Gamma \vdash \mathsf{Jc}(C)\colon El(D^\ulcorner\!+\!^\urcorner D')]\!] \blacktriangleright L \vdash \rho^{-1}a\colon ({}^\ulcorner\!+\!^\urcorner\langle d, d'\rangle)^*\tau}$$

where $\rho\colon \overline{L}\bullet({}^\ulcorner\!+\!^\urcorner\langle d, d'\rangle)^*\tau \cong \overline{L}\bullet(d^*\tau + d'^*\tau)$.

- **T-Exponential Externalisation**

$$\frac{[\![\Gamma \vdash M\colon El(D^\ulcorner\!\multimap\!^\urcorner D')]\!] \blacktriangleright L \vdash a\colon ({}^\ulcorner\!\multimap\!^\urcorner\langle d, d'\rangle)^*\tau}{[\![\Gamma \vdash \mathsf{If}(M)\colon El(D) \multimap El(D')]\!] \blacktriangleright L \vdash \rho a\colon d^*\tau \multimap d'^*\tau}$$
$$\frac{[\![\Gamma \vdash F\colon El(D) \multimap El(D')]\!] \blacktriangleright L \vdash a\colon d^*\tau \multimap d'^*\tau}{[\![\Gamma \vdash \mathsf{Jf}(F)\colon El(D^\ulcorner\!\multimap\!^\urcorner D')]\!] \blacktriangleright L \vdash \rho^{-1}a\colon ({}^\ulcorner\!\multimap\!^\urcorner\langle d, d'\rangle)^*\tau}$$

where $\rho\colon \overline{L}\bullet({}^\ulcorner\!\multimap\!^\urcorner\langle d, d'\rangle)^*\tau \cong \overline{L}\bullet(d^*\tau \multimap d'^*\tau)$.

- **Universal Type Retraction**

$$\frac{[\![\Gamma \vdash E\colon T\,dom]\!] \blacktriangleright L \vdash \langle id, e\rangle\colon T!^*\hat{U}}{[\![\Gamma \vdash \mathsf{Ret}(E)\colon dom]\!] \blacktriangleright L \vdash \langle id, ret\,e\rangle\colon !^*\hat{U}}$$

where $ret\colon T\hat{U} \to \hat{U}$ is the retraction morphism in Fib(1).

By inspecting the above relations we can see that they give rise to partial functions in the following way. Given a judgement J and "semantic sequents" S and S', if

$[\![J]\!] \blacktriangleright S$ and $[\![J]\!] \blacktriangleright S'$ then $S = S'$ in $\mathcal{C}$. Thus if we are given a structure Sg in $\mathcal{C}$ the assignments

$$\begin{aligned} \Gamma &\mapsto [\![\Gamma \text{ ctxt}]\!] \\ \Gamma, \alpha &\mapsto [\![\Gamma \vdash \alpha \text{ type}]\!] \\ \Gamma, M, \alpha &\mapsto [\![\Gamma \vdash M : \alpha]\!] \\ \Gamma, \chi, \Gamma' &\mapsto [\![\chi : \Gamma \to \Gamma']\!] \end{aligned}$$

give rise to partial functions. If one of these partial functions is defined at an argument J then we write $[\![J]\!]\Downarrow$.

Models of $\text{FIX}^{\mu}_{=}$ Theories

The notion of satisfaction of judgements arising from $\text{FIX}^{\mu}_{=}$ signatures is complicated by the type dependency. We give the definition of judgement satisfaction next, where it should be noted that in each instance of "J is satisfied iff S" the categorical structure S is unique.

1. Γ ctxt is satisfied iff $[\![\Gamma \text{ ctxt}]\!]\Downarrow$.
2. $\Gamma \vdash \alpha$ type is satisfied iff $[\![\Gamma \vdash \alpha \text{ type}]\!]\Downarrow$.
3. $\Gamma \vdash M : \alpha$ is satisfied iff $[\![\Gamma \vdash M : \alpha]\!]\Downarrow$.
4. $\chi : \Gamma \to \Gamma'$ is satisfied iff $[\![\Gamma \text{ ctxt}]\!]\Downarrow$ and $[\![\Gamma' \text{ ctxt}]\!]\Downarrow$.
5. $\Gamma = \Gamma'$ is satisfied iff $[\![\Gamma \text{ ctxt}]\!] \blacktriangleright L$ and $[\![\Gamma' \text{ ctxt}]\!] \blacktriangleright L$.
6. $\Gamma \vdash \alpha = \alpha'$ is satisfied iff $[\![\Gamma \vdash \alpha \text{ type}]\!] \blacktriangleright L \vdash F$ and $[\![\Gamma \vdash \alpha' \text{ type}]\!] \blacktriangleright L \vdash F$.
7. $\Gamma \vdash M = M' : \alpha$ is satisfied iff $[\![\Gamma \vdash M : \alpha]\!] \blacktriangleright L \vdash a : F$ and $[\![\Gamma \vdash M' : \alpha]\!] \blacktriangleright L \vdash a : F$.
8. $\chi = \chi' : \Gamma \to \Gamma'$ is satisfied iff $[\![\chi : \Gamma \to \Gamma']\!] \blacktriangleright l : L \to L'$ and $[\![\chi' : \Gamma \to \Gamma']\!] \blacktriangleright l : L \to L'$.

Given a $\text{FIX}^{\mu}_{=}$ theory Th and a structure $\mathbf{M}$, then we say that $\mathbf{M}$ is a *model* of Th if it satisfies the rules for introducing the Th axioms; (these rules can be found on Page 135).

The Substitution Lemma

The following lemma describes how the substitution of types and terms in the syntax of a $\text{FIX}^{\mu}_{=}$ theory is modelled by the categorical structure of a FIX category with attributes.

Lemma 9.3.1 Suppose that $[\![\chi : \Gamma \to \Gamma']\!] \blacktriangleright l : L \to L'$. Then it is the case that

- $[\![\Gamma' \vdash \alpha'(\vec{y}) \text{ type}]\!] \blacktriangleright L' \vdash F'$ implies $[\![\Gamma \vdash \alpha'(\chi) \text{ type}]\!] \blacktriangleright L \vdash \overline{l}^{*} F'$.

- $[\![\Gamma' \vdash M'(\vec{y}) : \alpha'(\vec{y})]\!] \blacktriangleright L' \vdash a' : F'$ implies $[\![\Gamma \vdash M'(\chi) : \alpha'(\chi)]\!] \blacktriangleright L \vdash \overline{l}^* a' : \overline{l}^* F'$.
- $[\![\chi' : \Gamma' \to \Gamma'']\!] \blacktriangleright l' : L' \to L''$ implies $[\![\chi' \circ \chi : \Gamma \to \Gamma'']\!] \blacktriangleright l' \circ l : L \to L''$.

Proof The proof proceeds by induction on the derivation of the various judgements. □

The Soundness Theorem

Theorem 9.3.2 ["FIX$^{\mu}_{=}$ Soundness"] Suppose we are given a FIX$^{\mu}_{=}$ theory Th over a FIX$^{\mu}_{=}$ signature Sg. The collection of judgements of the theory Th which are satisfied by a structure in a FIX category with attributes $\mathcal{C}$ is closed under the rules (see Page 135) for derivation of judgements in Th. Consequently a model $\mathbf{M}$ of Th satisfies all the judgements which are theorems of Th.

Proof Once again, the proof proceeds by an induction on the derivation of the various judgement forms; the previous Lemma will be used throughout the proof. □

Chapter 10
Prospects for Further Research

10.1 Loose Ends and Future Tasks

We give a concise review of what has been achieved; in particular we highlight loose ends and indicate possible further lines of research. The following comments coincide roughly with the order of presentation of material in the thesis.

Modular Approaches to Program Semantics

Each of the logics $\mathrm{FIX}_{=}$, FIX and $\mathrm{FIX}^{\mu}_{=}$ builds upon the computational let calculus. The fundamental notion underlying the let calculus is the separation of computations from values. The extensions we have considered provide expressive logics which allow us to reason about one particular notion of computation. The development of metalogics which combine different kinds of computation is clearly an important issue; for related work in this area see [Mog90b] and [Mog90a]. Most of the work to date concerns the combining of various monads (representing different forms of computation, such as those presented at the end of Chapter 1) at an equational level rather than at the level of predicates. The investigation of monadic predicate logics where one is able to vary the underlying monad is yet to be undertaken.

Domain Theoretic Properties of $\mathrm{FIX}_{=}$

In a $\mathrm{FIX}_{=}$ theory we always have fixpoints of terms at the higher order type $(\alpha{\to}T\beta){\to}\alpha{\to}T\beta$. All concrete models presented in this thesis are domain theoretic, thus by definition objects and morphisms have an associated order. When we consider fixpoints arising from the properties of FPO's in these categories, it is always the *least* such which is delivered. As we saw in Chapter 3, FIX categories (and hence $\mathrm{FIX}_{=}$ theories) have properties reminiscent of concrete categories of domains and axiomatic domain theory. The precise relationship between $\mathrm{FIX}_{=}$ and axiomatic approaches to domains needs to be established. One line of investigation is to consider what formal orders can be imposed on $\mathrm{FIX}_{=}$ and their connection to formal fixpoints.

An example of a formal order is the following; we give the merest sketch of details. We shall need the notion of canonical and non canonical terms. These arise from the introduction and elimination rules in the $\mathrm{FIX}_{=}$ logic. More precisely, the (raw)

canonical terms are given by the grammar

$$C ::= \langle\rangle \mid \langle M, N\rangle \mid \mathsf{Inl}_\alpha(M) \mid \mathsf{Inr}_\alpha(M) \mid \lambda_\alpha(F) \mid \mathsf{Val}(M) \mid \mathsf{O} \mid \mathsf{Suc}(N) \mid \omega \mid \sigma(E)$$

and the *non canonical* terms by

$$NC ::= \mathsf{Fst}(P) \mid \mathsf{Snd}(P) \mid \{F, G\}(C) \mid FM \mid \mathsf{Let}\,(E, F) \mid F^N(M) \mid \mathsf{It}(F, N)$$

We can define an operational reduction scheme where $C \Longrightarrow C$, $\mathsf{Fst}(\langle M, N\rangle) \Longrightarrow M$ and so on. Write $M[CL/\vec{x}]$ for the substitution of closed terms CL for the object level variables in M and ΣM for the ordered list of subterms of M (e.g. M is a subterm of $\langle M, N\rangle$). Then define the *simulation ordering* by the following fixpoint: Say that $M \leq N$ iff $\forall CL$, $M[CL/\vec{x}] \Longrightarrow M'$ implies $N[CL/\vec{y}] \Longrightarrow N'$ and $\Sigma M' \leq \Sigma N'$. Some work along these lines has been carried out by Smith for the simply typed λ calculus augmented with surjective pairing and natural numbers. In [Smi89] Smith shows that the formal fixpoint obtained from iterating the term $\perp \stackrel{\text{def}}{=} (\lambda x.x(x))(\lambda x.x(x))$ coincides with that arising from the usual fixpoint combinator Y, where the coincidence is defined up to the equivalence generated by a simulation ordering rather like the one sketched above. However, the proof techniques are a little unwieldy. It might be possible to obtain similar results for the $\mathrm{FIX}_=$ logic via a logical relations argument. Using the above ordering on terms of $\mathrm{FIX}_=$, a partial order could be imposed on the collection of global elements of a type α. The definition of the category $\mathcal{L}r$ could be changed so that the relation $\lhd$ of an object $(D, \lhd, \alpha)$ satisfies

$$d \leq d' \;\&\; d \lhd M \;\&\; d' \lhd M' \supset M \leq M'.$$

With such a relation it should be possible to see that the selection of a least fixpoint by a FPO in the first coordinate will force a proof of the same fact for the third coordinate.

Categorical Semantics of FIX

The semantics of FIX is in a rather unsatisfactory state. The definition of a FIX hyperdoctrine is complex and one would prefer more of its properties to be deducible from others. Originally a semantics which mimics the domain theoretic model of FIX was pursued, modelling FIX propositions via a distinguished class of subobjects in a suitable category (c.f. the hyperdoctrine model with fibres composed of *inclusive subsets* of ωcpo's). In order to set up a categorical logic correspondence one has to manufacture a FIX category together with a distinguished class of subobjects from the FIX logic (this is essentially the Grothendieck construction applied to the initial FIX hyperdoctrine). However, the category arising from such a construction is not cartesian closed; and it is not clear how to alter the FIX logical system in a consistent way to ensure cartesian closure. If this could be achieved one would hope that some of the conditions ensuring FIX soundness would come for free: consider the fibrewise induction conditions of a FIX hyperdoctrine (imposed) and the result which shows that Peano's axioms hold in a topos (toposes model predicates by subobjects).

The Existence and Disjunction Properties

The results about existence and disjunction in the FIX logic work only for closed terms. It is possible to envisage relativised versions of these theorems, for example one could investigate for which propositions $\Phi(x)$ a judgement

$$\Gamma, x{:}\alpha, \Phi(x) \vdash \Diamond(E(x), y.\Psi(x,y))$$

in the FIX logic entails that there is a term in context $\Gamma, x{:}\alpha \vdash M(x)$ for which $\Gamma, \alpha \vdash E(x) = \mathsf{Val}(M(x))$ and $\Gamma, x{:}\alpha, \Phi(x) \vdash \Psi(x, M(x))$.

Computational Adequacy Results for PCF

The results of Chapter 6 concerning computationally adequate translations of PCF into the FIX logic were proved using a technique due to Plotkin [Plo85]. It would be nice to see such results proven by a gluing argument. A skeleton of ideas for such a proof might be as follows: Define an operational semantics on the terms of the FIX logic with judgements of the form $\Gamma \vdash E \twoheadrightarrow M{:}\alpha$ where $\Gamma \vdash E{:}T\alpha$. This would be defined inductively by rules such as

$$\frac{\Gamma \vdash E \twoheadrightarrow M{:}\alpha \quad \Gamma, x{:}\alpha \vdash F(x){:}T\beta \quad \Gamma \vdash F(M) \twoheadrightarrow M'{:}\beta}{\Gamma \vdash \mathsf{Let}\,(E,F) \twoheadrightarrow M'{:}\beta}$$

The operational semantics of FIX would be set up to ensure that $\mathtt{m} \Longrightarrow \mathtt{c}$ iff $[\![\mathtt{m}]\!]^n \twoheadrightarrow \lceil \mathtt{c} \rceil$; moreover if $\Gamma \vdash [\![\mathtt{m}]\!]^n \twoheadrightarrow V$, then $V \equiv \lceil \mathtt{c}' \rceil$ for some unique c'. The idea is then to use the equality $\vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{c}]\!]^n$ to deduce that $[\![\mathtt{m}]\!]^n \twoheadrightarrow \lceil \mathtt{c}' \rceil$ and hence $\mathtt{m} \Longrightarrow \mathtt{c}'$, by way of a logical relations gluing argument. For example, recall the category $\mathcal{L}r$ of Chapter 5. If the definition of the relation $\lhd$ of an object $(D, \lhd, \alpha)$ had a clause of the form

$$e \lhd_{T\alpha} E \text{ iff } e = [d] \supset \exists M{:}\alpha.E \twoheadrightarrow M \qquad (*)$$

we could use this (together with the formal adequacy of the FIX logic which shows that if $\vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{c}]\!]^n = \mathsf{Val}(\lceil \mathtt{c} \rceil)$ then the ωcpo interpretation of $[\![\mathtt{m}]\!]^n$ is not bottom) to show that $[\![\mathtt{m}]\!]^n \twoheadrightarrow \lceil \mathtt{c}' \rceil$. The major obstacle here is that the terms E and M above are *equivalence classes up to FIX logic equality.* This problem could, perhaps, be surmounted by working at a 2-categorical level. Thus the initial FIX category $\mathcal{F}$ would be replaced by an initial "FIX 2-category". The latter would have types as objects, pure FIX terms as morphisms and 2-cells given by suitable reductions in the FIX logic, for example $\mathsf{Let}\,(\mathsf{Val}(M), F) \mapsto F(M)$. The notion of a FIX 2-category morphism would have to be formulated carefully, together with a proof that a category similar to $\mathcal{L}r$ but defined using the clause $(*)$ is indeed a FIX 2-category.

Semantics of PCF

The full abstraction problem for PCF has been investigated by a number of researchers [Plo77], [Sto88]. The FIX logic semantics given to PCF in Chapter 6 may

throw some light on the intricacies of full abstraction and possibly simplify known results; this is just speculation at the time of writing.

Adequacy Results for Languages with Recursive Types

We have not presented any applications of the $\mathrm{FIX}^{\mu}_{=}$ logic. A first step would be to state and prove adequacy results for a PCF style programming language with recursive types. For example, if $x_i \vdash \sigma$ were a type in context of the programming language, it would be translated to a judgement of the form $x_i\colon dom \vdash [\![\sigma]\!]\colon dom$. In particular, a recursive type $x_i \vdash \mu x.\sigma$ would be translated as $x_i\colon dom \vdash \mathsf{Fix}(x.[\![\sigma]\!])\colon dom$.

Synthetic Domain Theory

Little is known about the exact links between the work of this thesis and similar ideas from synthetic domain theory. It is the case that complete Σ-spaces form a (constructive) model of the $\mathrm{FIX}_{=}$ logic; for material relevant to synthetic domain theory see [Hyl82] and [Pho90]. One could perform a routine inter-translation of the systems to gain further incites into how they relate.

10.2 Final Conclusions

We have presented three logical systems which can be used to interpret programming languages. These logics can be used to give meaning to both call by name and call by value languages in a uniform way. Each logic has a clean categorical semantics together with a domain theoretic model. We have seen that one of these logics can be used to give computationally adequate interpretations of small programming languages. In essence, immediate future work consists of trying to simplify some of the categorical semantics, giving more extensive examples illustrating the use of the logical systems as program logics and in particular extending the applications to languages with recursive types.

Bibliography

[Bee85] M. Beeson. *Foundations of Constructive Mathematics.* Springer Verlag, 1985.

[Ben85] J. Benabou. Fibred category theory and the foundations of naive category theory. *Journal of Symbolic Logic*, 50(1):10–33, March 1985.

[Car86] L. Cardelli. A polymorphic lambda calculus with type:type. Technical Report 10, Systems Research Center, 130 Lytton Avenue, Palo Alto, CA, 1986.

[Car86] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 86.

[CGW89] T. Coquand, C. Gunter, and G. Winskel. Domain theoretic models of polymorphism. *Information and Computation*, 81:123–167, 1989.

[CP90a] R.L. Crole and A.M. Pitts. New foundations for fixpoint computations. In *5th Annual Symposium on Logic in Computer Science*, pages 489–497. I.E.E.E. Computer Society Press, 1990.

[CP90b] R.L. Crole and A.M. Pitts. New foundations for fixpoint computations: FIX hyperdoctrines and the FIX logic. Technical Report 204, University of Cambridge, Computer Laboratory, August 1990. Revised version accepted for the LICS 1990 Special Edition of Information and Computation.

[Cro90] R.L. Crole. Categories, equational logic and typed lambda calculi. Notes for a Graduate Lecture Course, September 1990.

[Dum77] M. Dummett. *Elements of Intuitionism.* O.U.P., 1977.

[Gir89] J.-Y. Girard. *Proofs and Types.* Cambridge Tracts in Theoretical Computer Science. C.U.P., 1989. Translated and with appendices by P. Taylor and Y. Lafont.

[HP89] J.M.E. Hyland and A.M. Pitts. The theory of constructions: Categorical semantics and topos-theoretic models. In *Categories in Computer Science and Logic*, volume 92 of *Contemp. Math.*, pages 137–199, 1989.

[Hyl82] J.M.E. Hyland. The effective topos. In A.S. Troelstra and D. van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, Studies in Logic and the Foundation of Mathematics. North Holland, 1982.

[Joh77] P.T. Johnstone. *Topos Theory.* Academic Press, 1977.

[JP78] P.T. Johnstone and R. Paré, editors. *Indexed Categories and their Applications*, volume 661 of *Lecture Notes In Mathematics.* Springer Verlag, 1978.

[Jun88] A. Jung. Cartesian closed categories of algebraic cpo's. Technical Report 1110, Technische Hochschule Darmstadt, January 1988.

[Kah88] G. Kahn. Natural semantics. In K. Fuchi and M. Nivat, editors, *Programming of Future Generation Computers*, pages 237–258. Elsevier Science Publishers B.V. North Holland, 1988.

[Kel82] M. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge University Press, 1982.

[Laf88] Y. Lafont. *Logiques, Catégories et Machines.* PhD thesis, Univ. Paris VII, 1988.

[Lan64] P.J. Landin. The mechanical evaluation of expressions. *Computer Journal*, 6:308–320, 1964.

[Law69] F.W. Lawvere. Adjointness in foundations. *Dialectica*, 23(3/4):281–296, 1969.

[LS80] J. Lambek and P.J. Scott. Intuitionist type theory and the free topos. *Journal of Pure and Applied Algebra*, 19:215–257, 1980.

[LS81] D.J. Lehmann and M.B. Smyth. Algebraic specification of data types: A synthetic approach. *Mathematical Systems Theory*, 14:97–139, 1981.

[LS86] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Mathematics. C.U.P., 1986.

[Mac71] S. Maclane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer Verlag, 1971.

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

[Mog89a] E. Moggi. Computational lambda calculus and monads. In *Fourth annual symposium on Logic In Computer Science*, pages 14–23. I.E.E.E. Computer Society Press, 1989.

[Mog89b] E. Moggi. Notions of computation and monads. Preprint, L.F.C.S., University of Edinburgh, U.K., November 1989.

[Mog90a] E. Moggi. A category theoretic account of program modules. Draft paper for the CLICS Project, September 1990.

[Mog90b] E. Moggi. Modular approach to denotational semantics, September 1990. Working Draft.

[MR77] M. Makkai and G.E. Reyes. *First Order Categorical Logic.* Lecture Notes In Mathematics. Springer Verlag, 1977.

[NPS90] B. Nordstrom, K. Petersson, and J.M. Smith. *Programming in Martin Löf's Type Theory*, volume 7 of *Monographs on Computer Science.* O.U.P., 1990.

[Pau87] L.C. Paulson. *Logic and Computation.* Cambridge Tracts in Theoretical Computer Science. C.U.P., 1987.

[Pho90] W.K.-S. Phoa. *Domain Theory in Realizability Toposes.* PhD thesis, University of Cambridge, 1990.

[Pit87] A.M. Pitts. Polymorphism is set theoretic, constructively. In *Summer Conference on Category Theory and Computer Science.* University of Edinburgh, Scotland, U.K., September 1987.

[Pit89] A.M. Pitts. Notes on categorical logic. Graduate Lecture Course, Cambridge University Computer Laboratory, 1989.

[Pit90a] A.M. Pitts. Categorical logic. Draft Chapter for the Handbook in Categorical Logic, 1990.

[Pit90b] A.M. Pitts. Evaluation logic. Technical Report 198, University of Cambridge, Computer Laboratory, August 1990.

[Plo75] G.D. Plotkin. Call by name, call by value and the λ calculus. *Theoretical Computer Science*, 1:125–129, 1975.

[Plo77] G.D. Plotkin. L.C.F considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Plo80] G.D. Plotkin. A course on denotational semantics. Lecture notes from the University of Edinburgh, Scotland, U.K., 1980.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI–FN 19, Department of Computer Science, University of Aarhus, Denmark, 1981.

[Plo85] G.D. Plotkin. Denotational semantics with partial functions. Unpublished lecture notes from CSLI summer school, 1985.

[Sco69a] D.S. Scott. Models of the lambda calculus. Unpublished manuscript, 1969.

[Sco69b] D.S. Scott. A type theoretic alternative to CUCH, ISWIM, OWHY. Unpublished manuscript, University of Oxford, 1969.

[Sco70a] D.S. Scott. The lattice of flow diagrams. Technical Report 3, Oxford University, Programming Research Group, 1970.

[Sco70b] D.S. Scott. Towards a mathematical theory of computation. In *4th Annual Princeton Conference on Information Sciences and Systems*, 1970.

[Sco71] D.S. Scott. Continuous lattices. Technical Report 7, Oxford University, Programming Research Group, 1971.

[Sco82] D.S. Scott. Domains for denotational semantics. In *ICALP 1982*, volume 140 of *Lecture Notes In Computer Science*, pages 577–613. Springer Verlag, 1982.

[See83] R.A.G. Seely. Hyperdoctrines, natural deduction and the beck condition. *Zeitschr. f. math. Logik und Grundlagen d. Math*, 29:505–542, 1983.

[Smi89] S.F. Smith. From operational to denotational semantics. Technical Report 89–12, The Johns Hopkins University, 1989.

[SP82] M.B. Smyth and G.D. Plotkin. The category theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11(4):761–783, 1982.

[SS71] D.S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. Technical Report 6, Oxford University, Programming Research Group, 1971.

[Sto88] A. Stoughton. *Fully Abstract Models of Programming Languages.* Research Notes in Theoretical Computer Science. Pitman/John Wiley, 1988.

[Str74] C. Strachey. The varieties of programming languages. Technical Report 10, Oxford University, Programming Research Group, 1974.

[Str89] T. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions.* PhD thesis, Universitat Passau, 1989. Reference MIP - 8913.

[SW74] C. Strachey and C. Wadsworth. Continuations–a mathematical theory for handling full jumps. Technical Report 11, Oxford University, Programming Research Group, 1974.

[Tay86] P. Taylor. *Recursive Domains, Indexed Category Theory and Polymorphism.* PhD thesis, University of Cambridge, 1986.

[WL83] G. Winskel and K.G. Larsen. Using information systems to solve recursive domain equations effectively. Technical Report 51, University of Cambridge, Computer Laboratory, 1983.