Computational Adequacy of the FIX-Logic

Roy L. Crole

Department of Computing, Imperial College,
LONDON, England.

July 13, 1995

Abstract

This paper presents computational adequacy results for the FIX logical system introduced by Crole and Pitts in LICS '90. More precisely, we take two simple PCF style languages (whose dynamic semantics follow a call-by-value and call-by-name regime) give translations of the languages into suitable judgements in the FIX-logic and prove that the translations are adequate for the static and dynamic semantics. This shows that the FIX-logic can be regarded as a programming metalogic which will uniformly interpret both call-by-value and call-by-name languages. The proofs of dynamic adequacy make use of a logical relations technique which is based on the methods of Plotkin and Tait. We also show that there is some choice in the translation of recursion; certain translations make use of an existence property of the FIX-logic to prove computational adequacy.

1 Introduction

This paper makes use of the FIX logical system (which was introduced in [3]) as a programming metalogic (see also [8]) into which programming languages can be translated and reasoned with. In this introduction we give a summary of the crucial elements of the FIX logical system and then give a brief heuristic outline of two simple programming languages which will be translated into the FIX-logic.

A Summary of the FIX Logical System

NB: A complete description of FIX can be found in [4]; we shall give only an outline of the format of FIX, and readers are urged to consult the latter paper in detail if they are not familiar with the FIX logical system.

FIX is an intuitionistic predicate logic which bears some similarity to standard intuitionistic calculus; for the latter see [5]. The primary form of judgement is a sequent-in-context of the form $\Gamma, \Lambda \vdash \Phi$. Here, Γ is a context of typed variables, Λ is a finite set of propositions and Φ is a single proposition. One should think of such a judgement as meaning that one has a derivation of the proposition Φ which involves a certain number of undischarged hypotheses each of which must appear in the set Λ. The variables occurring in the propositions in Λ and the proposition Φ are listed in Γ. The FIX-logic is specified by giving rules for inductively generating such judgements. More precisely, a FIX-theory is specified by giving a *FIX-signature* Sg of ground types, function symbols and relation symbols. From such a signature we can generate well formed types, terms and propositions, and from these we can generate theorems which are sequents-in-context of the above form.

Let us suppose we are given a FIX-signature Sg as above. The basic judgement forms that appear in a FIX-theory are

- *Terms-in-context* $\Gamma \vdash M\colon \alpha$ where a *context*, Γ, is a finite list of (object level variable, type) pairs, usually written $[x_1\colon \alpha_1, \ldots, x_n\colon \alpha_n]$ where the object level variables $x_1, \ldots, x_n$ are *distinct*, M is a raw term, and α is a type,
- *propositions-in-context* $\Gamma \vdash \Phi$ prop, where Φ is a raw proposition, and
- *sequents-in-context* $\Gamma, \Lambda \vdash \Phi$ as described above.

We write $\vdash M\colon \alpha$, $\vdash \Phi$ prop and $\vdash \Phi$ respectively if both Γ and Λ are empty.

We shall sketch the ideas behind each of these judgements. First the terms-in-context. The *types* consist of the ground types, natural numbers, unit, null, fixpoint type, (co)products, functions and computation types. The *raw terms* M are the usual ones associated with the latter types, and include terms formed (in the usual way) from the function symbols of Sg. We adopt Martin-Löf's theory of arities and expressions in order to write down the syntax of the raw terms. [1] Given a context

[1] See Appendix for details.

Γ, a *FIX-term* (in context Γ) is any raw FIX-term M satisfying $\Gamma \vdash M\colon \alpha$ for some (necessarily unique) type α. We refer to α as the *type* of the FIX-term M. Such FIX-terms are generated according to certain rules to be found in [4]. (We may also refer to a FIX-term $\Gamma \vdash M\colon \alpha$ when the latter is well-formed.)

The propositions-in-context are built up using the FIX-terms. There are precise rules (omitted here) for generating well formed propositions-in-context, which we call *FIX-propositions*; please see [4]. The FIX-terms act as individuals for the propositions. For each type α there is an equality predicate $=_\alpha$ on raw terms for which equations such as β and η equality for λ-calculus hold. In general, the FIX-propositions are generated by the usual rules for intuitionistic predicate calculus with equality *excluding* implication and existential quantification, but including certain other rules given in [4]. The rules directly relevant to this paper are

Universal Modality

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

Existential Modality

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

Now that we have the syntax for FIX, we can give rules for deducing sequents-in-context. A *FIX-theory*, Th, is specified by a FIX-signature (say Sg) 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 certain rules which appear in [4]. We refer to the general set-up as the *FIX-logic*.

We shall need the following results about FIX, which are stated without proof.

LEMMA 1.1 In the FIX-logic, if $\Gamma, \Lambda, \Phi \vdash \Psi$, then

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

PROPOSITION 1.2 Within the fix logical system, the following birule is derivable:

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

THEOREM 1.3 ["Existence Property"] If E is a closed term of type $T\alpha$, then the judgement $\vdash \Diamond(E, \Phi)$ 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.)

A Categorical Semantics for FIX

We outline the essence of a categorical semantics for a FIX-theory—for a general introduction to categorical semantics see [2]. Let $\mathcal{C}$ be a category. A *FIX-hyperdoctrine* is a $\mathcal{C}$-indexed poset $\mathbb{C}\colon \mathcal{C}^{op} \to \mathcal{POS}et$ which has enough structure to model FIX. The types are modelled by objects in $\mathcal{C}$, and FIX-terms by morphisms. Each FIX-proposition $\Gamma \vdash \Phi$ prop is modelled by an element in a posetal fibre of $\mathbb{C}$, where the fibre is over the object of $\mathcal{C}$ which is modelling the context Γ. Logical entailment is modelled in the usual way by the posetal order of the fibres. We illustrate with a picture:

Let $x\colon \alpha \vdash M\colon \beta$, $x\colon \alpha \vdash \Phi$ prop, and $x\colon \alpha, \Phi \vdash \Psi$ be a FIX-term, proposition and theorem respectively. Write $[\![-]\!]\colon \mathrm{FIX} \to \mathbb{C}$ for the interpretation (model) of FIX in $\mathbb{C}$. Write also $A \stackrel{\mathrm{def}}{=} [\![\alpha]\!]$, $B \stackrel{\mathrm{def}}{=} [\![\beta]\!]$, $m \stackrel{\mathrm{def}}{=} [\![x\colon \alpha \vdash M\colon \beta]\!]$, $p \stackrel{\mathrm{def}}{=} [\![x\colon \alpha \vdash \Phi]\!]$ and $q \stackrel{\mathrm{def}}{=} [\![x\colon \alpha \vdash \Psi]\!]$. Then:

$$x\colon \alpha, \Phi \vdash \Psi \quad \overset{[\![-]\!]}{\cdots\cdots\cdots\cdots\cdots\blacktriangleright} \quad p \leq q \quad \in \quad \mathbb{C}(A)$$

$$x\colon \alpha \vdash \Phi \quad \overset{[\![-]\!]}{\cdots\cdots\cdots\cdots\cdots\cdots\blacktriangleright} \quad p \quad \in \quad \mathbb{C}(A)$$

$$x\colon \alpha \vdash M\colon \beta \quad \overset{[\![-]\!]}{\cdots\cdots\cdots\cdots\blacktriangleright} \quad m\colon A \to B \quad \in \quad \mathcal{C}$$

A Domain Theoretic Semantics for FIX

We shall give a very brief summary of a domain-theoretic model of FIX, in order to set up notation for the remainder of this paper. Let $\omega\mathcal{CPO}$ be the category of ω-cocomplete posets (ω-cpos) and Scott-continuous functions. There is a functor $\mathcal{I}\colon \omega\mathcal{CPO}^{op} \to Poset$ which takes an ω-cpo to its poset of inclusive subsets and this is an example of a FIX-hyperdoctrine.

This gives a model of FIX in which types are interpreted by ω-cpos and propositions are interpreted as inclusive subsets. The inclusive subsets of an ω-cpo form a particular kind of poset-with-structure, and the sequents-in-context are interpreted by inequalities (inclusions) in such posets. As regards notation in this paper, we just note that $\omega\mathcal{CPO}$ is an example of a cartesian closed let-category (see [1]) for which the operation of lifting gives rise to the let-category structure. Let D, D' and D'' be three ω-cpos. We write $D \times D'$ for the binary product of D and D', $D \Rightarrow D'$ for the exponential of D and D', that is

$$D \Rightarrow D' \stackrel{\mathrm{def}}{=} \{\ f\colon D \to D' \ \mid \ f \text{ is a continuous function } \},$$

and if $f: D \times D' \to D''$ is a continuous function then then we write $\lambda(f): D \to (D' \Rightarrow D'')$ for the mate of f across the exponential adjunction, that is, $\lambda(f)$ is the "currying" of f. We shall write $((-)_\perp, \eta, \mathit{lift})$ for the let-structure, where if D is an ω-cpo we write $D_\perp \stackrel{\text{def}}{=} \{[d] \mid d \in D\} \bigcup \{\perp\}$ for the lift of D, $\eta_D: D \to D_\perp$ sends d to $[d]$, that is, η_D is the canonical inclusion of D into $D_\perp$, and given a continuous function $f: D \times D' \to D''_\perp$ then $\mathit{lift}(f): D \times D'_\perp \to D''_\perp$ where $\mathit{lift}(f)(d, \perp) \stackrel{\text{def}}{=} \perp$ and $\mathit{lift}(f)$ is otherwise f. The terminal ω-cpo will be written 1 and the natural number object $\mathbb{N}$. We shall use the semantic brackets $[\![-]\!]_{\omega\mathcal{CPO}}$ to denote the interpretation of FIX in $\omega\mathcal{CPO}$.

Two Simple Languages

We shall make use of two little programming languages, both of which are closely allied to Plotkin's PCF. Recall that 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 which was first investigated by Plotkin in [11].

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. A discussion of evaluation strategies can be found in [10].

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 $\sigma \ ::= \ \mathtt{bool} \mid \mathtt{nat} \mid \sigma \Rightarrow \sigma$. We write *Type* for the set of all types. 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
	$\mid$	$SS(\mathtt{m})$	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
	$\mid$	$\mathtt{R}_{\tau,\sigma}(f.x.\mathtt{m},\mathtt{n})$	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 standard, and omitted except for recursive functions:

Recursive Functions

$$\frac{\Gamma, f{:}\,\sigma \Rightarrow \sigma', x{:}\,\sigma \vdash \mathtt{m}{:}\,\sigma' \quad \Gamma \vdash \mathtt{n}{:}\,\sigma}{\Gamma \vdash \mathtt{R}_{\sigma,\sigma'}(f.x.\mathtt{m},\mathtt{n}){:}\,\sigma'}$$

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 $\mathtt{nat}$ and $\mathtt{bool}$ will be referred to as *ground* types.

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, and shall often abbreviate $\vdash \mathtt{m}{:}\,\sigma$ to just $\mathtt{m}{:}\,\sigma$. It will be convenient to write

$$Exp_\sigma \stackrel{\text{def}}{=} \{ \quad \mathtt{m} \quad \mid \quad \vdash \mathtt{m}{:}\,\sigma \quad \}.$$

The *canonical* QL expressions comprise those closed expressions which occur in the grammar:

$$\mathtt{c} \ ::= \ \mathtt{tt} \mid \mathtt{ff} \mid \mathtt{k}_n \mid \lambda x{:}\,\sigma.\mathtt{m},$$

and we write $Can_\sigma \subseteq Exp_\sigma$ for the set of canonical expressions assigned the type σ. We now give the syntax of QL a call by value dynamic semantics via an *evaluation relation* (see [7] and [12]) which will take the form $\mathtt{m} \Downarrow \mathtt{c}$, 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

$$\frac{\mathtt{c} \text{ canonical}}{\mathtt{c} \Downarrow \mathtt{c}}$$

Conditionals

$$\frac{\mathtt{b} \Downarrow \mathtt{tt} \quad \mathtt{m} \Downarrow \mathtt{c}}{\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n}) \Downarrow \mathtt{c}} \qquad \frac{\mathtt{b} \Downarrow \mathtt{ff} \quad \mathtt{n} \Downarrow \mathtt{c}}{\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n}) \Downarrow \mathtt{c}}$$

Arithmetic

$$\frac{\mathtt{m} \Downarrow \mathtt{k}_n}{SS(\mathtt{m}) \Downarrow \mathtt{k}_{n+1}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{k}_{n+1}}{\mathtt{P}(\mathtt{m}) \Downarrow \mathtt{k}_n} \qquad \frac{\mathtt{m} \Downarrow \mathtt{k}_0}{\mathtt{P}(\mathtt{m}) \Downarrow \mathtt{k}_0}$$

$$\frac{\mathtt{m} \Downarrow \mathtt{k}_0}{\mathtt{Z}(\mathtt{m}) \Downarrow \mathtt{tt}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{k}_{n+1}}{\mathtt{Z}(\mathtt{m}) \Downarrow \mathtt{ff}}$$

Functions

$$\frac{\mathtt{m} \Downarrow \lambda x{:}\sigma.\mathtt{m}' \quad \mathtt{n} \Downarrow \mathtt{c}' \quad \mathtt{m}'[\mathtt{c}'/x] \Downarrow \mathtt{c}}{\mathtt{mn} \Downarrow \mathtt{c}}$$

Recursive Functions

$$\frac{\mathtt{n} \Downarrow \mathtt{c}' \quad \mathtt{m}[\lambda x{:}\sigma.\mathtt{R}_{\sigma,\sigma'}(f.x.\mathtt{m},x)/f,\mathtt{c}'/x] \Downarrow \mathtt{c}}{\mathtt{R}_{\sigma,\sigma'}(f.x.\mathtt{m},\mathtt{n}) \Downarrow \mathtt{c}}$$

It is easy to see that the dynamic semantics is deterministic and if $\mathtt{m} \Downarrow \mathtt{c}$ then $\mathtt{m}$ and $\mathtt{c}$ have the same type. More precisely, the relation $\Downarrow$ defines a partial function between QL expressions in that if $\mathtt{m} \Downarrow \mathtt{c}$ and $\mathtt{m} \Downarrow \mathtt{c}'$ then $\mathtt{c}$ and $\mathtt{c}'$ are indeed α-equivalent.

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, $x_i\colon \sigma_i \vdash \mathtt{m}\colon \sigma$, 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 $x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n \vdash \mathtt{m}\colon \sigma$ is translated to

$$x_1\colon [\![\sigma_1]\!]^v, \ldots, x_n\colon [\![\sigma_n]\!]^v \vdash \vec{u}.[\![\mathtt{m}]\!]^v(\vec{x})\colon T[\![\sigma]\!]^v,$$

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 $fv([\![\mathtt{m}]\!]^v) = \{u_1, \ldots, u_n\}$.[2] Given a closed QL expression (in context) $\vdash \mathtt{m}\colon \sigma$, this is of course translated to a judgement $\vdash [\![\mathtt{m}]\!]^v\colon T[\![\sigma]\!]^v$. 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 $\Rightarrow$ TERM and which we shall denote by Pred and Zero. The (representatives for these) expressions are (using η equality in the meta-λ-calculus) given by

$$\begin{aligned}\mathsf{Pred}(n) &\stackrel{\text{def}}{=} \mathsf{Snd}((x.\langle \mathsf{Suc}(\mathsf{Fst}(x)), \mathsf{Fst}(x)\rangle)^n(\langle \mathsf{O}, \mathsf{O}\rangle))\\ \mathsf{Zero}(n) &\stackrel{\text{def}}{=} (x.\mathsf{Inr}_{unit}(\langle\rangle))^n(\mathsf{Inl}_{unit}(\langle\rangle))\end{aligned}$$

where the reader is referred to [4] for the definition of the syntax above.

Note that the judgements $n\colon nat \vdash \mathsf{Pred}(n)\colon nat$ and $n\colon nat \vdash \mathsf{Zero}(n)\colon unit + unit$ 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 definitions

$$\mathsf{Fix}_\alpha(f) \stackrel{\text{def}}{=} \mathsf{It}_\alpha(e.\mathsf{Let}\,(e, x.fx), \sigma(\omega))$$

and

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

for which it is immediate that $f\colon T\alpha \Rightarrow T\alpha \vdash \mathsf{Fix}_\alpha(f)\colon T\alpha$ and

$$f\colon (\alpha \Rightarrow T\beta) \Rightarrow \alpha \Rightarrow T\beta \vdash \mathsf{Y}_{\alpha,\beta}(f)\colon \alpha \Rightarrow T\beta$$

are FIX-terms-in-context.

The translation of QL into FIX is given below:

[2] See Appendix.

- $[\![\mathtt{nat}]\!]^v \stackrel{\text{def}}{=} nat$
- $[\![\mathtt{bool}]\!]^v \stackrel{\text{def}}{=} unit + unit$
- $[\![\sigma \Rightarrow \tau]\!]^v \stackrel{\text{def}}{=} [\![\sigma]\!]^v \Rightarrow T[\![\tau]\!]^v$
- $[\![x]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(u)$ where u is a meta variable.
- $[\![\mathtt{tt}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inl}_{unit}(\langle\rangle))$
- $[\![\mathtt{ff}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inr}_{unit}(\langle\rangle))$
- $[\![\mathtt{k}_n]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Suc}^n(\mathsf{O}))$
- $[\![\mathtt{C}_\sigma(\mathtt{b},\mathtt{m},\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{b}]\!]^v, x.\{y.[\![\mathtt{m}]\!]^v, y.[\![\mathtt{n}]\!]^v\}(x))$
- $[\![SS(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Suc}(x)))$
- $[\![\mathtt{P}(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Pred}(x)))$
- $[\![\mathtt{Z}(\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.\mathsf{Val}(\mathsf{Zero}(x)))$
- $[\![\mathtt{mn}]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{m}]\!]^v, f.\mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.fx))$
- $[\![\lambda x\colon\sigma.\mathtt{m}]\!]^v \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{[\![\sigma]\!]^v}(x.[\![\mathtt{m}]\!]^v))$
- $[\![\mathtt{R}_{\sigma,\sigma'}(f.x.\mathtt{m},\mathtt{n})]\!]^v \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{n}]\!]^v, y.\mathsf{Y}_{[\![\sigma]\!]^v,[\![\sigma']\!]^v}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v))))y)$

Interpretation of the Dynamic Semantics

Clearly the minimal requirement of an interpretation of the dynamics semantics of QL is soundness, namely that if $\mathtt{m} \Downarrow \mathtt{c}$ 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 $\mathtt{m} \Downarrow \mathtt{c}'$ 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 $\mathtt{m} \Downarrow \mathtt{c}$ to the judgement $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$.

4 Adequacy Results for QL

Static Adequacy for QL

PROPOSITION 4.1 ["QL Static Adequacy"] The interpretation of the static semantics of QL in FIX is adequate, in the sense that $x_i\colon \sigma_i \vdash \mathtt{m}\colon \sigma$ 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* $\mathtt{R}(f.x.\mathtt{m},\mathtt{n})$): From the definition of $[\![\mathtt{R}(f.x.\mathtt{m},\mathtt{n})]\!]^v$, the FIX-logic rules and the induction hypothesis, we have

$$x_i\colon \sigma_i, f\colon \sigma' \Rightarrow \sigma, x\colon \sigma' \vdash \mathtt{m}\colon \sigma \quad \text{and} \quad x_i\colon \sigma_i \vdash \mathtt{n}\colon \sigma',$$

from which $x_i\colon \sigma_i \vdash \mathtt{R}(f.x.\mathtt{m},\mathtt{n})\colon \sigma$ is immediate. □

Dynamic Adequacy of QL

We shall prove a theorem based on Plotkin's methods given in [13]. We write $D(-)$ for the composition $[\![-]\!]_{\omega\mathcal{CPO}} \circ \vec{u}.[\![-]\!]^v(\vec{x})\colon \mathrm{QL} \to \mathrm{FIX} \to \omega\mathcal{CPO}$ where $[\![-]\!]_{\omega\mathcal{CPO}}$ 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; more precisely, we shall define a family of type-indexed relations

$$(\lhd_\sigma \subseteq D(\sigma) \times \mathit{Can}_\sigma \mid \sigma \in \mathit{Type}).$$

In the definition which follows, each $\trianglelefteq_\sigma \subseteq D(\sigma)_\perp \times \mathit{Exp}_\sigma$ is a relation defined in terms of $\lhd_\sigma$ by asking that $e \trianglelefteq_\sigma \mathtt{m}$ iff

$$e = [d] \quad \text{implies} \quad \exists \mathtt{c} \in \mathit{Can}_\sigma \quad (\mathtt{m} \Downarrow \mathtt{c} \quad \text{and} \quad d \lhd_\sigma \mathtt{c}).$$

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 $n \in \mathbb{N}$.
- $f \lhd_{\sigma\Rightarrow\tau} \lambda x\colon \sigma.\mathtt{m}$ iff

$$\forall d \in D(\sigma). \quad \forall \mathtt{c} \in \mathit{Can}_\sigma. \quad (d \lhd_\sigma \mathtt{c} \quad \text{implies} \quad f(d) \trianglelefteq_\tau (\lambda x\colon \sigma.\mathtt{m})\mathtt{c}).$$

With this, we have the following lemmas:

LEMMA 4.2 Suppose that $\{d_i \mid i \in \omega\}$ is an ω-chain in $D(\sigma)$ and that $d_i \lhd_\sigma \mathtt{c}$ for each $i \in \omega$. Then $\bigvee_{i\in\omega} d_i \lhd_\sigma \mathtt{c}$.

PROOF We induct on the structure of σ. In the case that σ is `nat` and `bool` the result is immediate for there are only constant ω-chains in the cpo's $\mathbb{N}$ and $1+1$.

Now suppose that σ is $\sigma \Rightarrow \tau$, $\{f_i \mid i \in \omega\}$ is a chain in $D(\sigma) \Rightarrow D(\tau)_\perp$ and that $f_i \lhd_{\sigma\Rightarrow\tau} \mathtt{c}$ for each i and some canonical $\mathtt{c} \stackrel{\text{def}}{=} \lambda x{:}\sigma.\mathtt{m} \in Can_{\sigma\Rightarrow\tau}$. So for any $d \in D(\sigma)$ and canonical $\mathtt{c} \in Can_\sigma$ for which $d \lhd_\sigma \mathtt{c}$ we have $f_i d \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m})\mathtt{c}$. We need to show that $(\bigvee_{i\in\omega} f_i)d = \bigvee_{i\in\omega} f_i d \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m})\mathtt{c}$, that is if $\bigvee_{i\in\omega} f_i d$ is not $\perp$ (say $[d'] \in D(\tau)_\perp$) then there is some $\mathtt{c}' \in Can_\tau$ for which $(\lambda x{:}\sigma.\mathtt{m})\mathtt{c} \Downarrow \mathtt{c}'$ and $d' \lhd_\tau \mathtt{c}'$. So suppose that $\bigvee_{i\in\omega} f_i d = [d'] \in D(\tau)_\perp$ implying

$$K \stackrel{\text{def}}{=} \{i \in \omega \mid f_i d = [d'_i] \in D(\tau)_\perp\}$$

is non-empty and that $\bigvee_{k\in K} f_k d = [d']$. Now for any $k \in K$, we have $f_k d = [d'_k] \in D(\tau)_\perp$ implying that there is some (unique, for $\Downarrow$ defines a partial function) $\mathtt{c}' \in Can_\tau$ for which $(\lambda x{:}\sigma.\mathtt{m})\mathtt{c} \Downarrow \mathtt{c}'$ and $d'_k \lhd_\tau \mathtt{c}'$. But from the induction hypothesis, we have $\bigvee_{k\in K} d'_k = d' \lhd_\tau \mathtt{c}'$ which completes the proof. □

LEMMA 4.3 Suppose that $\{e_i \mid i \in \omega\}$ is an ω-chain in $D(\sigma)_\perp$ and that $e_i \trianglelefteq_\sigma \mathtt{m}$ for each i. Then $\bigvee_{i\in\omega} e_i \trianglelefteq_\sigma \mathtt{m}$.

PROOF Suppose that $\bigvee_{i\in\omega} e_i = [d] \in D(\sigma)_\perp$. Then

$$K \stackrel{\text{def}}{=} \{i \in \omega \mid e_i = [d_i]\}$$

is non-empty. Hence $\mathtt{m} \Downarrow \mathtt{c}$ for some unique $\mathtt{c}$ and $d_k \lhd_\sigma \mathtt{c}$ for each $k \in K$ implying that $\bigvee_{k\in K} d_k = d \lhd_\sigma \mathtt{c}$ by appeal to Lemma 4.2. □

We can now prove

THEOREM 4.4 Let $x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n \vdash \mathtt{m}{:}\sigma$ be a QL term in context and suppose that for $i = 1, \ldots, n$ we have $d_i \in D(\sigma_i)$, $\vdash c_i{:}\sigma_i$ and $d_i \lhd_{\sigma_i} \mathtt{c}_i$. Then the continuous function $D(\Gamma \vdash \mathtt{m}){:}\, D(\sigma_1) \times \ldots \times D(\sigma_n) \to D(\sigma)_\perp$ 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 three cases

(*Case* $\mathtt{m}$ *is* $\lambda x{:}\sigma.\mathtt{m}$): Suppose that the conditions of the lemma are satisfied. Then we need to show that $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m})(\vec{d}) \trianglelefteq_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$. Using the definition of $D(-)$ we can show that $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m}) = \eta \circ \lambda(D(\Gamma, x{:}\sigma \vdash \mathtt{m}))$ where

$$\eta{:}\, D(\sigma) \Rightarrow D(\tau)_\perp \to (D(\sigma) \Rightarrow D(\tau)_\perp)_\perp.$$

Hence $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m})(\vec{d}) = [\lambda(D(\Gamma, x{:}\sigma \vdash \mathtt{m}))(\vec{d})]$. By definition of the $\trianglelefteq$ relation we show that $\lambda(D(\Gamma, x{:}\sigma \vdash \mathtt{m}))(\vec{d}) \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$; thus if $d \lhd_\sigma \mathtt{c}$ it remains

to show $\lambda(D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m}))(\vec{d})(d) \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}])\mathtt{c}$. 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 $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}, \mathtt{c}/x] \Downarrow \mathtt{c}'$ for some $\mathtt{c}'$ provided that $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m})(\vec{d})(d)$ is not $\bot$. But then $(\lambda x{:}\sigma.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}])\mathtt{c} \Downarrow \mathtt{c}'$ 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, $\Gamma \vdash \mathtt{m}{:}\sigma \Rightarrow \tau$ 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 $[f] \trianglelefteq_{\sigma\Rightarrow\tau} \mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$ and $[d] \trianglelefteq_\sigma \mathtt{n}[\vec{\mathtt{c}}/\vec{x}]$. Hence $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}] \Downarrow \lambda x{:}\sigma.\mathtt{m}'$ and $\mathtt{n}[\vec{\mathtt{c}}/\vec{x}] \Downarrow \mathtt{c}$. 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 $(\lambda x{:}\sigma.\mathtt{m}')\mathtt{c} \Downarrow \mathtt{c}'$. Thus we may deduce $\mathtt{m}'[\mathtt{c}/x] \Downarrow \mathtt{c}'$ and conclude $\mathtt{m}[\vec{\mathtt{c}}/\vec{x}]\mathtt{n}[\vec{\mathtt{c}}/\vec{x}] \Downarrow \mathtt{c}'$.

(*Case* $\mathtt{m}$ *is* $\mathtt{R}_{\sigma,\tau}(f.x.\mathtt{m}, \mathtt{n})$): We have to show that

$$D(\Gamma \vdash \mathtt{R}(f.x.\mathtt{m}, \mathtt{n}))(\vec{d}) \trianglelefteq_\tau \mathtt{R}(f.x.\mathtt{m}[\vec{\mathtt{c}}/\vec{x}], \mathtt{n}[\vec{\mathtt{c}}/\vec{x}])$$

where $\Gamma, y{:}\sigma \Rightarrow \tau, x{:}\sigma \vdash \mathtt{m}{:}\tau$ and $\Gamma \vdash \mathtt{n}{:}\sigma$. Let us write $\tilde{\mathtt{m}} \stackrel{\text{def}}{=} \mathtt{m}[\vec{\mathtt{c}}/\vec{x}]$ and similarly for $\tilde{\mathtt{n}}$. Suppose that $D(\Gamma \vdash \mathtt{R}(f.x.\mathtt{m}, \mathtt{n}))(\vec{d}) = [\overline{d}] \in D(\tau)_\bot$; call the supposition $(*)$. Then it remains to show that there is $\overline{\mathtt{c}} \in \mathit{Can}_\tau$ for which $\mathtt{R}(f.x.\tilde{\mathtt{m}}, \tilde{\mathtt{n}}) \Downarrow \overline{\mathtt{c}}$ and $\overline{d} \lhd_\tau \overline{\mathtt{c}}$. Unravelling the definitions, there are (using an obvious notation) continuous functions $D(\Gamma \vdash \mathtt{n}){:}\,\Pi D(\vec{\sigma}) \to D(\sigma)_\bot$ and

$$D(\Gamma, y{:}\sigma \Rightarrow \tau, x{:}\sigma \vdash \mathtt{m}) : \Pi D(\vec{\sigma}) \times (D(\sigma) \Rightarrow D(\tau)_\bot) \times D(\sigma) \to D(\tau)_\bot$$

so that

$$\lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m}))) : \Pi D(\vec{\sigma}) \to (D(\sigma) \Rightarrow D(\tau)_\bot) \Rightarrow (D(\sigma) \Rightarrow D(\tau)_\bot)$$

in which given a morphism $h{:}\,D \times D' \to D''$ in $\omega\mathcal{CPO}$ we have $\lambda(h){:}\,D \to (D' \Rightarrow D'')$ from the cartesian closed structure. Hence we have a continuous function

$$D(\Gamma \vdash \mathtt{R}(f.x.\mathtt{m}, \mathtt{n})) \stackrel{\text{def}}{=} ev_\bot \circ \langle \chi, D(\Gamma \vdash \mathtt{n})\rangle : \Pi D(\vec{\sigma}) \to D(\tau)_\bot$$

where $\chi{:}\,\Pi D(\vec{\sigma}) \to (D(\sigma) \Rightarrow D(\tau)_\bot)$ with

$$\chi(\vec{d}) \stackrel{\text{def}}{=} \bigvee_{i\in\omega} \lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot})$$

where $\underline{\bot}{:}\,D(\sigma) \to D(\tau)_\bot$ is the bottom map, and $ev_\bot \stackrel{\text{def}}{=} \mathit{lift}(ev)$ is the lift of the evaluation morphism $ev{:}\,(D(\sigma) \Rightarrow D(\tau)_\bot) \times D(\sigma) \to D(\tau)_\bot$ in $\omega\mathcal{CPO}$.

By induction $D(\Gamma \vdash \mathtt{n})(\vec{d}) \trianglelefteq_\sigma \tilde{\mathtt{n}}$. By supposition $(*)$ we have

$$ev_\bot \circ \langle \chi, D(\Gamma \vdash \mathtt{n})\rangle(\vec{d}) = [\overline{d}]$$

implying that $D(\Gamma \vdash \mathtt{n})(\vec{d})$ is not bottom, say $D(\Gamma \vdash \mathtt{n})(\vec{d}) = [d'] \in D(\sigma)_\bot$. Hence there is $\mathtt{c}' \in \mathit{Can}_\sigma$ for which $\tilde{\mathtt{n}} \Downarrow \mathtt{c}'$ and $d' \lhd_\sigma \mathtt{c}'$.

We claim that $\chi(\vec{d}) \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)$. We prove this by showing that

$$\lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot}) \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)$$

and then appealing to Lemma 4.2. We induct on i. In the case that $i = 0$ all is clear. Now we assume the relation holds for $0, 1, \ldots, i$, we choose arbitrary $d \lhd_\sigma \mathtt{c}$, and we show that

$$D(\Gamma, y, x \vdash \mathtt{m})(\vec{d}, \lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot}), d) \trianglelefteq_\tau (\lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x))\mathtt{c}.$$

By the induction hypothesis on i, we have

$$\lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot}) \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x),$$

and hence from the structural induction hypothesis

$$D(\Gamma, y, x \vdash \mathtt{m})(\vec{d}, \lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot}), d) \quad \trianglelefteq_\tau \quad \tilde{\mathtt{m}}[\lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)/y, \mathtt{c}/x].$$

So suppose that $D(\Gamma, y, x \vdash \mathtt{m})(\vec{d}, \lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))(\vec{d})^i(\underline{\bot}), d)$ is not bottom, say $[\widehat{d}] \in D(\tau)_\bot$. Then there is $\widehat{\mathtt{c}}$ for which

$$\tilde{\mathtt{m}}[\lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)/y, \mathtt{c}/x] \Downarrow \widehat{\mathtt{c}}$$

from which we deduce $(\lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x))\mathtt{c} \Downarrow \widehat{\mathtt{c}}$ and $\widehat{d} \lhd_\tau \widehat{\mathtt{c}}$. This completes the induction on i.

Using once again the structural induction hypothesis for $\mathtt{m}$ and our recent deductions, we have

$$D(\Gamma, y, x \vdash \mathtt{m})(\vec{d}, \chi(\vec{d}), d') \trianglelefteq_\tau \mathtt{m}[\vec{\mathtt{c}}/\vec{x}, \lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)/y, \mathtt{c}'/x]$$

and using the supposition $(*)$ together with the observation

$$\begin{array}{rcl} D(\Gamma, y, x \vdash \mathtt{m})(\vec{d}, \chi(\vec{d}), d') & = & \lambda(\lambda(D(\Gamma, y, x \vdash \mathtt{m})))\chi(\vec{d})(d') \\ & = & \chi(\vec{d})(d') \\ & = & ev_\bot \circ \langle \chi(\vec{d}), [d'] \rangle \\ & = & [\overline{d}] \in D(\tau)_\bot \end{array}$$

we have the existence of some $\overline{\mathtt{c}}$ for which $\tilde{\mathtt{m}}[\lambda x{:}\,\sigma.\mathtt{R}(f.x.\tilde{\mathtt{m}}, x)/y, \mathtt{c}'/x] \Downarrow \overline{\mathtt{c}}$. Using also $\tilde{\mathtt{n}} \Downarrow \mathtt{c}'$ we can deduce that $\mathtt{R}(f.x.\tilde{\mathtt{m}}, \tilde{\mathtt{n}}) \Downarrow \overline{\mathtt{c}}$, and of course $\overline{d} \lhd_\tau \overline{\mathtt{c}}$, so we are done. $\Box$

We shall also need the following lemma:

Lemma 4.5 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 The proof is a trivial structural induction on $\mathtt{m}$. We illustrate with one example.

(*Case* $\mathtt{m}$ *is* $\mathtt{R}(f.x.\mathtt{m},\mathtt{n})$):

$$\begin{array}{rcl} \vdash [\![\mathtt{R}(f.x.\mathtt{m},\mathtt{n})\ \ [\mathtt{c}/y]]\!]^v & = & \mathsf{Let}\,([\![\mathtt{n}[\mathtt{c}/y]]\!]^v, u.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}[\mathtt{c}/y]]\!]^v))))u) \\ \text{which by induction is} & = & \mathsf{Let}\,([\![\mathtt{n}]\!]^v, u.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v))))u)\ \ [\lceil\mathtt{c}\rceil/y] \\ & = & [\![\mathtt{R}(f.x.\mathtt{m},\mathtt{n})]\!]^v[\lceil\mathtt{c}\rceil/y]. \end{array}$$

□

THEOREM 4.6 ["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 $\mathtt{m} \Downarrow \mathtt{c}$ 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 $\mathtt{m} \Downarrow \mathtt{c}'$.

PROOF The proof in the forwards direction proceeds by induction on the derivation of $\mathtt{m} \Downarrow \mathtt{c}$; we give details for the cases of application and recursive function terms.

(*Case* $\mathtt{mn} \Downarrow \mathtt{c}$): Using minimality of $\Downarrow$ and the induction hypothesis, we obtain

$$\begin{array}{rcl} \vdash [\![\mathtt{m}]\!]^v & = & \mathsf{Val}(\lambda(x.[\![\mathtt{m}']\!]^v)) \\ \vdash [\![\mathtt{n}]\!]^v & = & \mathsf{Val}(\lceil\mathtt{c}'\rceil) \\ \vdash [\![\mathtt{m}'[\mathtt{c}'/x]]\!]^v & = & \mathsf{Val}(\lceil\mathtt{c}\rceil). \end{array}$$

Thus we have

$$\begin{array}{rcl} \vdash [\![\mathtt{mn}]\!]^v & = & \mathsf{Let}\,([\![\mathtt{m}]\!]^v, f.\mathsf{Let}\,([\![\mathtt{n}]\!]^v, x.fx)) \\ & = & \lambda(x.[\![\mathtt{m}']\!]^v)\lceil\mathtt{c}'\rceil \\ & = & [\![\mathtt{m}']\!]^v[\lceil\mathtt{c}'\rceil/x] \\ \text{which by Lemma 4.5} & = & [\![\mathtt{c}]\!]^v, \end{array}$$

as required.

(*Case* $\mathtt{R}(f.x.\mathtt{m},\mathtt{n}) \Downarrow \mathtt{c}$): Using minimality of $\Downarrow$ and the induction hypothesis, we obtain

$$\begin{array}{rcl} \vdash [\![\mathtt{n}]\!]^v & = & \mathsf{Val}(\lceil\mathtt{c}'\rceil) \\ \vdash [\![\mathtt{m}[\lambda x{:}\,\sigma.\mathtt{R}(f.x.\mathtt{m},x)/f,\mathtt{c}'/x]]\!]^v & = & \mathsf{Val}(\lceil\mathtt{c}\rceil). \end{array}$$

Let us put $M \stackrel{\text{def}}{=} \lambda(f.\lambda(x.[\![\mathtt{m}]\!]^v))$ and note that

$$\begin{array}{rcl} \vdash [\![\lambda x{:}\,\sigma.\mathtt{R}(f.x.\mathtt{m},x)]\!]^v & = & \mathsf{Val}(\lambda(x.\mathsf{Y}(M)x)) \\ & = & \mathsf{Val}(\mathsf{Y}(M)). \end{array}$$

Thus we have

$$\begin{array}{rcl}
\vdash [\![\mathtt{R}(f.x.\mathtt{m},\mathtt{n})]\!]^v & = & \mathsf{Let}\,([\![\mathtt{n}]\!]^v, y.\mathsf{Y}(\lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v)))y)) \\
& = & \lambda(f.(\lambda(x.[\![\mathtt{m}]\!]^v)))\ \ \mathsf{Y}(M)\ \ \ulcorner \mathtt{c}' \urcorner \\
& = & \lambda(x.[\![\mathtt{m}]\!]^v)[\mathsf{Y}(M)/f]\ \ \ulcorner \mathtt{c}' \urcorner \\
& = & \lambda(x.[\![\mathtt{m}]\!]^v[\mathsf{Y}(M)/f])\ \ \ulcorner \mathtt{c}' \urcorner \\
& = & [\![\mathtt{m}]\!]^v[\mathsf{Y}(M)/f][\ulcorner \mathtt{c}' \urcorner/x] \\
& = & [\![\mathtt{m}]\!]^v[\ulcorner \lambda x{:}\,\sigma.\mathtt{R}(f.x.\mathtt{m},x)\urcorner/f, \ulcorner \mathtt{c}' \urcorner/x] \\
\text{which by Lemma 4.5} & = & [\![\mathtt{c}]\!]^v,
\end{array}$$

and so we are done.

For the converse direction, suppose that $\vdash [\![\mathtt{m}]\!]^v = [\![\mathtt{c}]\!]^v$. We have $[\![\mathtt{m}]\!]^v = \mathsf{Val}(\ulcorner \mathtt{c} \urcorner)$ and hence it is the case that $D(\vdash \mathtt{m})(*)$ is not $\bot$, say $[d]$. Appeal to Theorem 4.4 to deduce that $[d] \trianglelefteq_\sigma \mathtt{m}$ and hence there is some canonical $\mathtt{c}'$ for which $\mathtt{m} \Downarrow \mathtt{c}'$ by the definition of $\trianglelefteq$. □

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 $\sigma \ ::= \ \mathtt{bool} \mid \mathtt{nat} \mid \sigma \Rightarrow \sigma$. The (raw) expressions of HPCF are given by the grammar:

$\mathtt{m}$	$::=$	x	variables
	$\mid$	$\mathtt{tt}$	truth
	$\mid$	$\mathtt{ff}$	falsity
	$\mid$	$\mathtt{k}_n$	natural numbers
	$\mid$	$\mathtt{C}_\sigma$	conditional
	$\mid$	SS	successor
	$\mid$	$\mathtt{P}$	predecessor
	$\mid$	$\mathtt{Z}$	zero test
	$\mid$	$\mathtt{Y}_\sigma$	fixpoints
	$\mid$	$\mathtt{mn}$	application
	$\mid$	$\lambda x{:}\,\sigma.\mathtt{m}$	function definition

The Static Semantics of HPCF

The static semantics is presented using judgements of the form $\Gamma \vdash \mathtt{m}{:}\,\sigma$; the rules for such type assignments are completely standard and are omitted.

The Dynamic Semantics of HPCF

The *canonical* HPCF expressions consist of the subset of closed expressions which occur in the grammar

$$\mathtt{c} ::= \mathtt{tt} \mid \mathtt{ff} \mid \mathtt{C}_\sigma \mid \mathtt{k}_n \mid SS \mid \mathtt{P} \mid \mathtt{Z} \mid \mathtt{Y}_\sigma \mid \lambda x{:}\sigma.\mathtt{m} \mid \mathtt{C}_\sigma\mathtt{b} \mid \mathtt{C}_\sigma\mathtt{bm}$$

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

$$\frac{\mathtt{c}\ \text{canonical}}{\mathtt{c} \Downarrow \mathtt{c}}$$

Conditionals

$$\frac{\mathtt{m} \Downarrow \mathtt{C}_\sigma}{\mathtt{mb} \Downarrow \mathtt{C}_\sigma\mathtt{b}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{C}_\sigma\mathtt{b}}{\mathtt{mn} \Downarrow \mathtt{C}_\sigma\mathtt{bn}}$$

$$\frac{\mathtt{m} \Downarrow \mathtt{C}_\sigma\mathtt{bm}' \quad \mathtt{b} \Downarrow \mathtt{tt} \quad \mathtt{m}' \Downarrow \mathtt{c}}{\mathtt{mn} \Downarrow \mathtt{c}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{C}_\sigma\mathtt{bm}' \quad \mathtt{b} \Downarrow \mathtt{ff} \quad \mathtt{n} \Downarrow \mathtt{c}}{\mathtt{mn} \Downarrow \mathtt{c}}$$

$$\frac{\mathtt{l} \Downarrow \mathtt{C}_\sigma \quad \mathtt{b} \Downarrow \mathtt{tt} \quad \mathtt{m} \Downarrow \mathtt{c}}{\mathtt{lbmn} \Downarrow \mathtt{c}} \qquad \frac{\mathtt{l} \Downarrow \mathtt{C}_\sigma \quad \mathtt{b} \Downarrow \mathtt{ff} \quad \mathtt{n} \Downarrow \mathtt{c}}{\mathtt{lbmn} \Downarrow \mathtt{c}}$$

Arithmetic

$$\frac{\mathtt{m} \Downarrow SS \quad \mathtt{n} \Downarrow \mathtt{k}_n}{\mathtt{mn} \Downarrow \mathtt{k}_{n+1}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{P} \quad \mathtt{n} \Downarrow \mathtt{k}_{n+1}}{\mathtt{mn} \Downarrow \mathtt{k}_n} \qquad \frac{\mathtt{m} \Downarrow \mathtt{P} \quad \mathtt{n} \Downarrow \mathtt{k}_0}{\mathtt{mn} \Downarrow \mathtt{k}_0}$$

$$\frac{\mathtt{m} \Downarrow \mathtt{Z} \quad \mathtt{n} \Downarrow \mathtt{k}_0}{\mathtt{mn} \Downarrow \mathtt{tt}} \qquad \frac{\mathtt{m} \Downarrow \mathtt{Z} \quad \mathtt{n} \Downarrow \mathtt{k}_{n+1}}{\mathtt{mn} \Downarrow \mathtt{ff}}$$

Fixpoints

$$\frac{\mathtt{m} \Downarrow \mathtt{Y}_\sigma \quad \mathtt{nY}_\sigma\mathtt{n} \Downarrow \mathtt{c}}{\mathtt{mn} \Downarrow \mathtt{c}}$$

Functions

$$\frac{\mathtt{m} \Downarrow \lambda x{:}\sigma.\mathtt{m}' \quad \mathtt{m}'[\mathtt{n}/x] \Downarrow \mathtt{c}}{\mathtt{mn} \Downarrow \mathtt{c}}$$

REMARK 5.1 Plotkin originally specified the operational semantics of PCF via a single step reduction relation of the form $\mathtt{m} \rightsquigarrow \mathtt{n}$ 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 [11]. Note that good textbook accounts of such operational semantics are [14] and [6]. We omit the details, but remark that the reflexive, transitive closure of such a single step reduction relation will yield the natural semantics style reduction relation:

PROPOSITION 5.2 Let $\mathtt{m}$ and $\mathtt{c}$ be closed HPCF terms with $\mathtt{c}$ canonical. Then $\mathtt{m} \Downarrow \mathtt{c}$ iff $\mathtt{m} \rightsquigarrow^* \mathtt{c}$, where $\rightsquigarrow^*$ is the reflexive transitive closure of $\rightsquigarrow$. □

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:

- $[\![\mathtt{nat}]\!]^n \stackrel{\text{def}}{=} nat$
- $[\![\mathtt{bool}]\!]^n \stackrel{\text{def}}{=} unit + unit$
- $[\![\sigma \Rightarrow \tau]\!]^n \stackrel{\text{def}}{=} T[\![\sigma]\!]^n \Rightarrow T[\![\tau]\!]^n$
- $[\![x]\!]^n \stackrel{\text{def}}{=} u$ where u is a meta variable.
- $[\![\mathtt{tt}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inl}_{unit}(\langle\rangle))$
- $[\![\mathtt{ff}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Inr}_{unit}(\langle\rangle))$
- $[\![\mathtt{k}_n]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\mathsf{Suc}^n(\mathsf{O}))$
- $[\![\mathtt{C}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{bool}]\!]^n}(b.\mathsf{Val}(\lambda_{T[\![\sigma]\!]^n}(z.(\mathsf{Val}(\lambda_{T[\![\sigma]\!]^n}(z'.\mathsf{Let}\,(b, x.\{y.z, y.z'\}(x)\ldots)$
- $[\![SS]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Suc}(x)))))$
- $[\![\mathtt{P}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Pred}(x)))))$

- $[\![\mathtt{Z}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T[\![\mathtt{nat}]\!]^n}(y.\mathsf{Let}\,(y, x.\mathsf{Val}(\mathsf{Zero}(x)))))$
- $[\![\mathtt{Y}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T(T[\![\sigma]\!]^n \Rightarrow T[\![\sigma]\!]^n)}(y.\mathsf{Fix}_{[\![\sigma]\!]^n}(\lambda_{T[\![\sigma]\!]^n}(x.\mathsf{Let}\,(y, f.fx)))))$
- $[\![\mathtt{mn}]\!]^n \stackrel{\text{def}}{=} \mathsf{Let}\,([\![\mathtt{m}]\!]^n, f.f[\![\mathtt{n}]\!]^n))$
- $[\![\lambda x{:}\,\sigma.\mathtt{m}]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{[\![T\sigma]\!]^n}(x.[\![\mathtt{m}]\!]^n))$

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 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 9.

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 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

$$x_i{:}\,T[\![\sigma_i]\!]^n \vdash \vec{u}.[\![\mathtt{m}]\!]^n(\vec{x}){:}\,T(T[\![\sigma]\!]^n \Rightarrow T[\![\sigma]\!]^n)$$

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 write $D(-)$ for the composition

$$[\![-]\!]_{\omega\mathcal{CPO}} \circ \vec{u}.[\![-]\!]^n(\vec{x}) \colon \mathrm{HPCF} \to \mathrm{FIX} \to \omega\mathcal{CPO},$$

where once again $[\![-]\!]_{\omega\mathcal{CPO}}$ is the standard domain-theoretic semantics of FIX. We shall define a type-indexed family of relations

$$(\lhd_\sigma \subseteq D(\sigma) \times \mathit{Can}_\sigma \mid \sigma \in \mathit{Type}).$$

In the definition which follows, each

$$\trianglelefteq_\sigma \subseteq D(\sigma)_\perp \times \mathit{Exp}_\sigma$$

is a relation defined in terms of $\lhd_\sigma$, where we shall write $e \trianglelefteq_\sigma \mathtt{m}$ to mean that if $e = [d] \in D(\sigma)_\perp$ then $\mathtt{m} \Downarrow \mathtt{c}$ for some canonical $\mathtt{c}$ and $d \lhd_\sigma \mathtt{c}$. We put:

- In the cases that $\mathtt{c}$ is one of $\mathtt{tt}$, $\mathtt{ff}$, SS, $\mathtt{P}$, $\mathtt{Z}$, $\mathtt{C}_\sigma$, $\mathtt{Y}_\sigma$ we set $d \lhd_\sigma \mathtt{c}$ iff $d = [\![\ulcorner \mathtt{c} \urcorner]\!]_{\omega\mathcal{CPO}}(*) \in D(\sigma)$ where of course $\vdash \ulcorner \mathtt{c} \urcorner : [\![\sigma]\!]^n$ in FIX and $[\![\ulcorner \mathtt{c} \urcorner]\!]_{\omega\mathcal{CPO}} : 1 \to D(\sigma)$.
- $d \lhd_{\sigma\Rightarrow\sigma\Rightarrow\sigma} \mathtt{C}_\sigma\mathtt{b}$ iff

$$\forall e \in D(\sigma)_\perp.\quad \forall \mathtt{m} \in \mathit{Exp}_\sigma.\quad (e \trianglelefteq_\sigma \mathtt{m} \quad \text{implies} \quad d(e) \trianglelefteq_{\sigma\Rightarrow\sigma} \mathtt{C}_\sigma\mathtt{bm}).$$

- $d \lhd_{\sigma\Rightarrow\sigma} \mathtt{C}_\sigma\mathtt{bm}$ iff

$$\forall e \in D(\sigma)_\perp.\quad \forall \mathtt{n} \in \mathit{Exp}_\sigma.\quad (e \trianglelefteq_\sigma \mathtt{n} \quad \text{implies} \quad d(e) \trianglelefteq_\sigma \mathtt{C}_\sigma\mathtt{bmn}).$$

- $f \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{m}$ iff

$$\forall e \in D(\sigma)_\perp.\quad \forall \mathtt{m}' \in \mathit{Exp}_\sigma.\quad (e \trianglelefteq_\sigma \mathtt{m}' \quad \text{implies} \quad f(e) \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m})\mathtt{m}').$$

Now we prove some lemmas.

LEMMA 7.2 Suppose that $d_i \lhd_\sigma \mathtt{c}$ and that $\{d_i \mid i \in \omega\}$ is an ω-chain in $D(\sigma)$. Then $\bigvee_{i\in\omega} d_i \lhd_\sigma \mathtt{c}$.

PROOF We induct on the structure of σ.

(*Case σ is* $\mathtt{nat}$): The only ω-chains in $\mathbb{N}$ are constant; result trivial.

(*Case σ is* $\mathtt{bool}$): The only ω-chains in $1 + 1$ are constant; result trivial.

(*Case σ is $\sigma \Rightarrow \tau$*): There are a number of subcases according to type matchings of $\sigma \Rightarrow \tau$ with the type of $\mathtt{c}$.

(*Sub-case* $\mathtt{c}$ *is* SS, $\mathtt{P}$, $\mathtt{Z}$, $\mathtt{Y}_\rho$, $\mathtt{C}_\rho$): In each case the ω-chain is constant and equal to $[\![\ulcorner \mathtt{c} \urcorner]\!]_{\omega\mathcal{CPO}}(*)$ and the result is immediate.

(*Sub-case* $\mathtt{c}$ *is* $\lambda x{:}\sigma.\mathtt{m}$): Suppose that $d_i \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\mathtt{m}$. Take any $e \trianglelefteq_\sigma \mathtt{m}'$; it remains to show that $(\bigvee_{i\in\omega} d_i)e \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m})\mathtt{m}'$. Suppose that $\bigvee_{i\in\omega} d_i(e) = [g] \in D(\tau)_\perp$. Then the set $J \stackrel{\text{def}}{=} \{i \in \omega \mid d_i(e) = [g_i] \in D(\tau)_\perp\}$ is non-empty. By hypothesis we have $[g_j] \trianglelefteq_\tau (\lambda x{:}\sigma.\mathtt{m})\mathtt{m}'$ and so $(\lambda x{:}\sigma.\mathtt{m})\mathtt{m}' \Downarrow \mathtt{c}$ with $g_j \lhd_\tau \mathtt{c}$ for each $j \in J$. But by the main structural induction hypothesis, $\bigvee_{j\in J} g_j = g \lhd_\tau \mathtt{c}$ and we are done.

(*Sub-case* $\mathtt{c}$ *is* $\mathtt{C}_\rho\mathtt{b}$): In this case, $\sigma \Rightarrow \tau$ type matches $\rho \Rightarrow (\rho \Rightarrow \rho)$. Suppose that $d_i \lhd_{\rho\Rightarrow\rho\Rightarrow\rho} \mathtt{C}_\rho\mathtt{b}$, take $e_1 \trianglelefteq_\rho \mathtt{m}_1$ and we shall show that $(\bigvee_{i\in\omega} d_i)e_1 \trianglelefteq_{\rho\Rightarrow\rho} \mathtt{C}_\rho\mathtt{bm}_1$. Suppose that $\bigvee_{i\in\omega} d_i(e_1) = [g] \in (D(\rho)_\perp \Rightarrow D(\rho)_\perp)_\perp$ and then we show that if $e_2 \trianglelefteq_\rho \mathtt{m}_2$ we have $g(e_2) \trianglelefteq_\rho \mathtt{C}_\rho\mathtt{bm}_1\mathtt{m}_2$. So finally suppose that $g(e_2) = [h] \in D(\rho)_\perp$ and we shall show $\mathtt{C}_\rho\mathtt{bm}_1\mathtt{m}_2 \Downarrow \mathtt{c}$ for some $\mathtt{c}$ and that $h \lhd_\rho \mathtt{c}$.

Set $J \stackrel{\text{def}}{=} \{i \in \omega \mid d_i(e_1) = [g_i] \in (D(\rho)_\perp \Rightarrow D(\rho)_\perp)_\perp\}$ which must be non-empty by the above assumptions, and of course $g = \bigvee_{j\in J} g_j$. By the original supposition we have $g_j \lhd_{\rho\Rightarrow\rho} \mathtt{C}_\sigma\mathtt{bm}_1$ and by the above assumptions the set $K \stackrel{\text{def}}{=} \{j \in J \mid g_j(e_2) = [h_j] \in D(\rho)_\perp\}$ is non-empty. Certainly we have $[h_k] \trianglelefteq_\rho \mathtt{C}_\rho\mathtt{bm}_1\mathtt{m}_2$ implying that $\mathtt{C}_\rho\mathtt{bm}_1\mathtt{m}_2 \Downarrow \mathtt{c}$ for some $\mathtt{c}$ and $h_k \lhd_\rho \mathtt{c}$, and from the structural induction hypothesis $h = \bigvee_{k\in K} h_k \lhd_\rho \mathtt{c}$.

(*Sub-case* $\mathtt{c}$ *is* $\mathtt{C}_\rho\mathtt{bm}$): Similar to the last case. □

LEMMA 7.3 Suppose that $e_i \trianglelefteq_\sigma \mathtt{m}$ and that $\{e_i \mid i \in \omega\}$ is an ω-chain in $D(\sigma)_\perp$. Then $\bigvee_{i\in\omega} e_i \trianglelefteq_\sigma \mathtt{m}$.

PROOF Immediate from Lemma 7.2. □

LEMMA 7.4 If $d \lhd_{\sigma\Rightarrow\tau} \mathtt{c}$, where $\mathtt{c}$ runs over the possible cases of SS, $\mathtt{P}$, $\mathtt{Z}$, $\mathtt{Y}_\sigma$, $\mathtt{C}_\sigma$, $\mathtt{C}_\sigma\mathtt{b}$, $\mathtt{C}_\sigma\mathtt{bm}$, $\lambda x{:}\sigma.\mathtt{m}$, then whenever $e \trianglelefteq_\sigma \mathtt{m}$, we have $d(e) \trianglelefteq_\tau \mathtt{cm}$.

PROOF

(*Case* $\mathtt{c}$ *is* $\mathtt{C}_\sigma\mathtt{b}$, $\mathtt{C}_\sigma\mathtt{bm}$, $\lambda x{:}\sigma.\mathtt{m}$): These all follow by definition.

(*Case* $\mathtt{c}$ *is* SS, $\mathtt{P}$, $\mathtt{Z}$): These are all similar; we illustrate with SS. Let $s \lhd_{\mathtt{nat}\Rightarrow\mathtt{nat}} SS$ and $e \trianglelefteq_{\mathtt{nat}} \mathtt{m}$. Unravelling the definition of $\lhd_{\mathtt{nat}\Rightarrow\mathtt{nat}}$, $s \in \mathbb{N}_\perp \Rightarrow \mathbb{N}_\perp$ sends $\perp$ to $\perp$ and $[n]$ to $[n+1]$. If $e = \perp$ we are done. If $e = [n]$ then $\mathtt{m} \Downarrow \mathtt{k}_n$ and clearly $s(e) = [n+1] \trianglelefteq_{\mathtt{nat}} SS\mathtt{m}$.

(*Case* $\mathtt{c}$ *is* $\mathtt{C}_\sigma$): Let $k \lhd_{\mathtt{bool}\Rightarrow\sigma\Rightarrow\sigma\Rightarrow\sigma} \mathtt{C}_\sigma$, and so

$$k \in (1+1)_\perp \Rightarrow (D(\sigma)_\perp \Rightarrow (D(\sigma)_\perp \Rightarrow D(\sigma)_\perp)_\perp)_\perp.$$

Let $e \trianglelefteq_{\mathtt{bool}} \mathtt{b}$, note that by definition of k we must have

$$k(e) = [k'] \in D(\sigma)_\perp \Rightarrow (D(\sigma)_\perp \Rightarrow D(\sigma)_\perp)_\perp$$

and hence it remains to show that $k' \lhd_{\sigma\Rightarrow\sigma\Rightarrow\sigma} \mathtt{C}_\sigma\mathtt{b}$. Using the definitions, we repeat the procedure with $e_1 \trianglelefteq_\sigma \mathtt{m}_1$ and $e_2 \trianglelefteq \mathtt{m}_2$, and noting that

$$k'(e_1) = [k''] \in (D(\sigma)_\perp \Rightarrow D(\sigma)_\perp)_\perp$$

by the definition of k, it remains to see that $k''(e_2) \trianglelefteq_\sigma \mathtt{C}_\sigma \mathtt{b}\mathtt{m}_1\mathtt{m}_2$. If $e = \bot$, then $k''(e_2) = \bot$ and we are done. Otherwise without loss of generality we have $e = [i(*)] \in (1+1)_\bot$ and $\mathtt{b} \Downarrow \mathtt{tt}$. If $e_1 = \bot$, then as $k''(e_2) = e_1$ we are done. Otherwise $k''(e_2) = e_1 = [d_1] \in D(\sigma)_\bot$, so $\mathtt{m}_1 \Downarrow \mathtt{c}_1$ for some $\mathtt{c}_1$, $d_1 \lhd_\sigma \mathtt{c}_1$, and $\mathtt{b} \Downarrow \mathtt{tt}$ implies that $\mathtt{C}_\sigma \mathtt{b}\mathtt{m}_1\mathtt{m}_2 \Downarrow \mathtt{c}_1$.

(*Case* $\mathtt{c}$ *is* $\mathtt{Y}_\sigma$): Suppose that $y \lhd_{(\sigma\Rightarrow\sigma)\Rightarrow\sigma} \mathtt{Y}_\sigma$ and $f \trianglelefteq_{\sigma\Rightarrow\sigma} \mathtt{m}$. If $f = \bot$ then $y(f) = \bot \in D(\sigma)_\bot$ by the definition of y and we are done. If not, then $f = [g] \in (D(\sigma)_\bot \Rightarrow D(\sigma)_\bot)_\bot$, so $\mathtt{m} \Downarrow \mathtt{c}$ for some $\mathtt{c}$ and $g \lhd_{\sigma\Rightarrow\sigma} \mathtt{c}$. We have to see that $y(f) \stackrel{\text{def}}{=} \bigvee_{i\in\omega} g^i(\bot) \trianglelefteq_\sigma \mathtt{Y}_\sigma\mathtt{m}$. To prove this, we first claim that $g^i(\bot) \trianglelefteq_\sigma \mathtt{Y}_\sigma\mathtt{m}$ for each $i \in \omega$. For $i = 0$ we are okay. Assume inductively that $g^i(\bot) \trianglelefteq_\sigma \mathtt{Y}_\sigma\mathtt{m}$. We already have $g \lhd_{\sigma\Rightarrow\sigma} \mathtt{c}$ and by type inference $\mathtt{c}$ can not be $\mathtt{Y}_\rho$ for any type ρ. Therefore, it follows from the previous cases of this proof that $g^{i+1}(\bot) \trianglelefteq_\sigma \mathtt{c}(\mathtt{Y}_\sigma\mathtt{m})$. If $g(\bot) = \bot$ the claim holds. If not, then from the inductive assumption $\mathtt{Y}_\sigma\mathtt{m} \Downarrow \mathtt{c}'$. But it certainly follows that $\mathtt{c}\mathtt{Y}_\sigma\mathtt{m} \Downarrow \mathtt{c}'$ and if $g^{i+1}(\bot) = [d_{i+1}]$ then $d_{i+1} \lhd_\sigma \mathtt{c}'$, from which we deduce $g^{i+1}(\bot) \trianglelefteq_\sigma \mathtt{Y}_\sigma\mathtt{m}$. Appealing to Lemma 7.3 we are done. □

We can use the lemmas to show the next theorem.

THEOREM 7.5 Let $x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n \vdash \mathtt{m}{:}\sigma$ be a term-in-context of HPCF and let $e_i \trianglelefteq_{\sigma_i} \mathtt{m}_i$ for each i. Then the continuous function

$$D(\Gamma \vdash \mathtt{m}){:}\, D(\sigma_1)_\bot \times \ldots D(\sigma_n)_\bot \to D(\sigma)_\bot$$

satisfies $D(\Gamma \vdash \mathtt{m})(\vec{e}) \trianglelefteq_\sigma \mathtt{m}[\vec{\mathtt{m}}/\vec{x}]$.

PROOF The proof proceeds by a structural induction on the raw term $\mathtt{m}$. We shall write $\tilde{\mathtt{m}}$ for $\mathtt{m}[\vec{\mathtt{m}}/\vec{x}]$ and similarly for $\tilde{\mathtt{n}}$.

(*Case* $\mathtt{m}$ is x_i): $D(\Gamma \vdash x_i)(\vec{e}) = e_i \trianglelefteq_{\sigma_i} \mathtt{m}_i = x_i[\vec{\mathtt{m}}/\vec{x}]$.

(*Case* $\mathtt{m}$ is $\mathtt{tt}$, $\mathtt{ff}$, $\mathtt{k}_n$, SS, $\mathtt{P}$, $\mathtt{Z}$, $\mathtt{Y}_\sigma$, $\mathtt{C}_\sigma$): All of these cases are essentially identical; we illustrate with $\mathtt{Y}_\sigma$. $D(\Gamma \vdash \mathtt{Y}_\sigma)(\vec{e}) = D(\vdash \mathtt{Y}_\sigma)(*) \trianglelefteq_{(\sigma\Rightarrow\sigma)\Rightarrow\sigma} \mathtt{Y}_\sigma$ which is immediate for $D(\vdash \mathtt{Y}_\sigma)(*) = [[\![\ulcorner \mathtt{Y}_\sigma \urcorner]\!]_{\omega\mathcal{CPO}}(*)]$ and $[\![\ulcorner \mathtt{Y}_\sigma \urcorner]\!]_{\omega\mathcal{CPO}}(*) \lhd_{(\sigma\Rightarrow\sigma)\Rightarrow\sigma} \mathtt{Y}_\sigma$.

(*Case* $\mathtt{m}$ is $\lambda x{:}\sigma.\mathtt{m}$): We need to see that $D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m})(\vec{e}) \trianglelefteq_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\tilde{\mathtt{m}}$, where, say, $\Gamma, x{:}\sigma \vdash \mathtt{m}{:}\tau$. One can check that

$$D(\Gamma \vdash \lambda x{:}\sigma.\mathtt{m}) = \eta_{D(\sigma)_\bot \Rightarrow D(\tau)_\bot} \circ \lambda(D(\Gamma, x{:}\sigma \vdash \mathtt{m}))$$

and so it remains to prove that $\lambda(D(\Gamma, x{:}\sigma \vdash \mathtt{m}))(\vec{e}) \lhd_{\sigma\Rightarrow\tau} \lambda x{:}\sigma.\tilde{\mathtt{m}}$. Let $e' \trianglelefteq_\sigma \mathtt{m}'$ and we shall show that $D(\Gamma, x{:}\sigma \vdash \mathtt{m})(\vec{e}, e') \trianglelefteq_\tau (\lambda x{:}\sigma.\tilde{\mathtt{m}})\mathtt{m}'$. By structural induction, we have $D(\Gamma, x{:}\sigma \vdash \mathtt{m})(\vec{e}, e) \trianglelefteq_\tau \tilde{\mathtt{m}}[\mathtt{m}'/x]$. So suppose that $D(\Gamma, x{:}\sigma \vdash \mathtt{m})(\vec{e}, e) = [d] \in D(\tau)_\bot$. Then $\tilde{\mathtt{m}}[\mathtt{m}'/x] \Downarrow \mathtt{c}$ for some $\mathtt{c}$ and $d \lhd_\tau \mathtt{c}$. But then $(\lambda x{:}\sigma.\tilde{\mathtt{m}})\mathtt{m}' \Downarrow \mathtt{c}$ also, and we are done.

(*Case* $\mathtt{m}$ is $\mathtt{m}\mathtt{n}$): We need to see that $D(\Gamma \vdash \mathtt{m}\mathtt{n})(\vec{e}) \trianglelefteq_\tau \tilde{\mathtt{m}}\tilde{\mathtt{n}}$ (call this $(*)$) where, say, $\Gamma \vdash \mathtt{m}{:}\sigma \Rightarrow \tau$ and $\Gamma \vdash \mathtt{n}{:}\sigma$. Suppose that $D(\Gamma \vdash \mathtt{m}\mathtt{n})(\vec{e})$ is not bottom. Unravelling the

definitions, this implies that we must have $D(\Gamma \vdash \mathtt{m})(\vec{e}) = [f] \in (D(\sigma)_\perp \Rightarrow D(\tau)_\perp)_\perp$. By structural induction we have $D(\Gamma \vdash \mathtt{m})(\vec{e}) \trianglelefteq_{\sigma\Rightarrow\tau} \tilde{\mathtt{m}}$ and $D(\Gamma \vdash \mathtt{n})(\vec{e}) \trianglelefteq_\sigma \tilde{\mathtt{n}}$, and so $\tilde{\mathtt{m}} \Downarrow \mathtt{c}$ with $f \lhd_{\sigma\Rightarrow\tau} \mathtt{c}$ and $(*)$ now becomes $f(D(\Gamma \vdash \mathtt{n})(\vec{e})) \trianglelefteq_\tau \tilde{\mathtt{m}}\tilde{\mathtt{n}}$. We have to consider the sub-cases of $\vdash \mathtt{c}\colon \sigma \Rightarrow \tau$. Note that here $\mathtt{c}$ could be SS, $\mathtt{P}$, $\mathtt{Z}$, $\mathtt{Y}_\sigma$, $\mathtt{C}_\sigma$, $\mathtt{C}_\sigma\mathtt{b}$, $\mathtt{C}_\sigma\mathtt{bm}$ or $\lambda x\colon\sigma.\mathtt{m}$. The proof is very similar for each of these sub-cases; we give just two.

(*Sub-case* $\mathtt{c}$ *is* $\mathtt{Y}_\sigma$): Suppose that $y \lhd_{(\sigma\Rightarrow\sigma)\Rightarrow\sigma} \mathtt{Y}_\sigma$, and of course

$$D(\Gamma \vdash \mathtt{n})(\vec{e}) \trianglelefteq_{\sigma\Rightarrow\sigma} \tilde{\mathtt{n}}$$

and $\tilde{\mathtt{m}} \Downarrow \mathtt{Y}_\sigma$ as deduced above. By Lemma 7.4, we have

$$f(D(\Gamma \vdash \mathtt{n})(\vec{e})) \trianglelefteq_\sigma \mathtt{Y}_\sigma\tilde{\mathtt{n}},$$

and it is easy to see that $f(D(\Gamma \vdash \mathtt{n})(\vec{e})) \trianglelefteq_\sigma \tilde{\mathtt{m}}\tilde{\mathtt{n}}$ follows from $\tilde{\mathtt{m}} \Downarrow \mathtt{Y}_\sigma$.

(*Sub-case* $\mathtt{c}$ *is* $\lambda x\colon\sigma.\mathtt{m}$): We have $\tilde{\mathtt{m}} \Downarrow \lambda x\colon\sigma.\mathtt{m}'$ and $f \lhd_{\sigma\Rightarrow\tau} \lambda x\colon\sigma.\mathtt{m}'$. Hence from Lemma 7.4 we have $f(D(\Gamma \vdash \mathtt{n})(\vec{e})) \trianglelefteq_\tau (\lambda x\colon\sigma.\mathtt{m}')\tilde{\mathtt{n}}$. If

$$f(D(\Gamma \vdash \mathtt{n})(\vec{e})) = [d] \in D(\tau)_\perp$$

then $(\lambda x\colon\sigma.\mathtt{m}')\tilde{\mathtt{n}} \Downarrow \mathtt{c}$ for some $\mathtt{c}$ and $d \lhd_\tau \mathtt{c}$. But then $\mathtt{m}'[\tilde{\mathtt{n}}/x] \Downarrow \mathtt{c}$ and so $\tilde{\mathtt{m}}\tilde{\mathtt{n}} \Downarrow \mathtt{c}$. □

LEMMA 7.6 With the call by name interpretation of HPCF, and $x\colon\sigma \vdash \mathtt{m}\colon\tau$, $\vdash \mathtt{n}\colon\sigma$ HPCF terms in context, we have

$$\vdash [\![\mathtt{m}[\mathtt{n}/x]]\!]^n = [\![\mathtt{m}]\!]^n[[\![\mathtt{n}]\!]^n/u],$$

where $[\![x]\!]^v \stackrel{\text{def}}{=} u$ and $[[\![\mathtt{n}]\!]^n/u]$ is substitution in the meta-λ-calculus.

PROOF Trivial induction. □

THEOREM 7.7 ["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 $\mathtt{m} \Downarrow \mathtt{c}$ 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 $\mathtt{m} \Downarrow \mathtt{c}'$.

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 $\Downarrow$ 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

$$\begin{array}{rrl} \vdash [\![\mathtt{mn}]\!]^n & \stackrel{\text{def}}{=} & \mathsf{Let}\,([\![\mathtt{m}]\!]^n, f.f[\![\mathtt{n}]\!]^n) \\ & = & \lambda(x.[\![\mathtt{m}']\!]^n)[\![\mathtt{n}]\!]^n \\ & = & [\![\mathtt{m}']\!]^n[[\![\mathtt{n}]\!]^n/x] \\ \text{which via Lemma 7.6} & = & [\![\mathtt{c}]\!]^n \end{array}$$

as required.

(*Case* Fixpoints): Using minimality of $\Downarrow$, 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.

For the converse, suppose that we have $\vdash [\![\mathtt{m}]\!]^n = [\![\mathtt{c}]\!]^n$. Of course $\vdash [\![\mathtt{m}]\!]^n = \mathsf{Val}(\lceil \mathtt{c} \rceil)$ and hence it is the case that $D(\vdash \mathtt{m})(*)$ is not $\bot$, say $[d] \in D(\sigma)_\bot$. Appealing to Theorem 7.5, we can deduce that $[d] \trianglelefteq_\sigma \mathtt{m}$ and hence there is some canonical $\mathtt{c}'$ for which $\mathtt{m} \Downarrow \mathtt{c}'$. □

8 An Alternative Translation of Fixpoints

All of the results of Sections 6 and 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 3.

The translation of the fixpoint constants $\mathtt{Y}_\sigma$ now takes the form

$$[\![\mathtt{Y}_\sigma]\!]^n \stackrel{\text{def}}{=} \mathsf{Val}(\lambda_{T(T[\![\sigma]\!]^n \Rightarrow T[\![\sigma]\!]^n)}(y.\mathsf{Let}\,(y, x.\mathsf{Fix}_{[\![\sigma]\!]^n}(x)))).$$

In order to prove a computational adequacy result which uses this new translation, we shall need

LEMMA 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

$$\frac{\Gamma, \Lambda \vdash \Diamond(\mathsf{Let}\,(E, F), \Phi)}{\Gamma, \Lambda \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}$$

PROOF The labelling of steps in the prooftrees is informal and for guidance only. We have

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x{:}\,\alpha, y{:}\,\beta, \Lambda, \Phi(y), F(x) = \mathsf{Val}(y) \vdash \Diamond(F(x), \Phi)}(\Diamond\mathtt{i})}{\Gamma, y{:}\,\beta, \Lambda, \Diamond(E, x.\Phi(y) \wedge F(x) = \mathsf{Val}(y)) \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}(*)}{\Gamma, y{:}\,\beta, \Lambda, \Phi(y) \wedge \Diamond(E, x.F(x) = \mathsf{Val}(y)) \vdash \Diamond(E, x.\Diamond(F(x), \Phi))}(\mathtt{fr})$$

and

$$\dfrac{}{\Gamma, y{:}\,\beta, \Lambda, \mathsf{Let}\,(E, F) = \mathsf{Val}(y), \Phi(y) \vdash \Phi(y) \wedge \Diamond(E, x.F(x) = \mathsf{Val}(y))}(\mathtt{mod})$$

where the step $(*)$ follows from Lemma 1.1 and rule (`fr`) is proved in Proposition 1.2. Applying the cut rule 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 8.2 Theorem 7.7 remains true if we replace the translation of the constants $\mathtt{Y}_\sigma$ given on Page 18 with that given on Page 23.

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 $\Downarrow$, 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 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 1.3), 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 the dynamic adequacy of Section 7. □

9 Prospects for Further Research and Acknowledgements

• We have shown that we can use a translation of two simple languages in the FIX-logic to reason about the original languages: this makes the induction principles of FIX available to prove properties of programmes written in the source languages. We understand that the programming languages studied in this paper are very much toy languages. Work is in progress to see just how useful logics such as FIX are for reasoning about realistic languages. Recent work of Pitts and Stark has highlighted the problems associated with local store in the language ML; see [9]. Work in progress is considering "realistic" fragments of ML (not involving local state) and developing programming logics which are based on both FIX and also Evaluation Logic [8].

We hope to implement a theorem prover based around the FIX-logic (or a similar monadic logic) which will mechanise the procedure of reasoning in the FIX-logic. Practicalities of such an approach will be assessed by proving toy programmes in HPCF and QL correct, and also considering similar computational adequacy results for fragments of working functional languages. In particular, this includes looking at ML.

• We have investigated the possibility of proving dynamic adequacy of QL and HPCF using a 2-categorical version of gluing. Partial results seem to indicate that this may not provide much of a simplification over the logical relations method presented in this paper.

• I would like to thank Andrew Pitts for conversations about this work, part of which was conducted at the Computer Laboratory, Cambridge University. Martin Hyland and Eugenio Moggi made valuable comments on a primeval form of this paper. Finally I must thank the anonymous referee for some suggestions on how to improve the presentation of this paper.

• This research was supported by the ESPRIT CLICS Project, BRA 3003, and an EPSRC Research Fellowship.

Appendix: Martin Löf's Theory of Arities and Expressions

Let *Gnd* be a fixed collection of *ground types*. Then the *simple types* over *Gnd* are defined by the grammar $\sigma ::= \gamma \mid \sigma \Rightarrow \sigma$ where γ is a ground type. We shall write $\sigma_1 \Rightarrow \sigma_2 \Rightarrow \ldots \Rightarrow \sigma_n$ for the simple type $\sigma_1 \Rightarrow (\sigma_2 \Rightarrow \ldots (\sigma_{n-1} \Rightarrow \sigma_n)\ldots)$. For each σ we are given a countably infinite set of variables which are tagged with type information, namely $Var^\sigma \stackrel{\text{def}}{=} \{x_1^\sigma, x_2^\sigma, x_3^\sigma, \ldots\}$; formally an element of Var^σ is given by a pair (x, σ) where x is an atomic symbol and σ is a simple type. We are also given a (possibly empty) collection of constants which are also type tagged, denoted by *Con*. The collection of *raw λ-terms* is given by the grammar $M ::= c^\sigma \mid x^\sigma \mid M(N) \mid x^\sigma.M$ where c^σ is a constant. We write $\vec{x^\sigma}.M$ for $x_1^\sigma.x_2^\sigma.\ldots.x_n^\sigma.M$

A *typing* is a judgement of the form $M \in \sigma$ where M is a raw λ-term and σ simple type. These judgements are generated by the following rules

Simply Typed λ-Calculus with Type Tagged Terms

$$\frac{x^\sigma \in Var^\sigma}{x^\sigma \in \sigma} \qquad \frac{c^\sigma \in Con}{c^\sigma \in \sigma} \qquad \frac{M \in \sigma \Rightarrow \tau \quad N \in \sigma}{M(N) \in \tau} \qquad \frac{x^\sigma \in \sigma \quad M \in \tau}{x^\sigma.M \in \sigma \Rightarrow \tau}$$

It is assumed that the reader is familiar with the notion of free and bound variables; $fv(M)$ will denote the free variables of the raw λ-term M. We shall write $M = N$ to denote that the raw λ-terms M and N are α-equivalent. A *λ-term* is an α-equivalence class of raw λ-terms M satisfying the judgement $M \in \sigma$ for some (necessarily unique) simple type σ. σ is called the *type* of the λ-term M. We shall not distinguish notationally between a raw λ-term M and the λ-term which it (may) denote. We shall write $M[N/x]$ for the λ-term M with occurrences of x replaced by the λ-term N with renaming of free variables in N to avoid capture. Note that substitution is well defined up to α equivalence.

Now we shall define a binary relation between λ-terms, denoted $M \twoheadrightarrow M'$, by the following rules:

Reduction Rules

$$\frac{}{x^\sigma.M(N) \twoheadrightarrow M[N/x^\sigma]}\,(\texttt{beta}) \qquad \frac{}{x^\sigma.M(x^\sigma) \twoheadrightarrow M}\,(\texttt{eta})\ [x^\sigma \notin fv(M)]$$

$$\frac{M \twoheadrightarrow M'}{M(N) \twoheadrightarrow M'(N)}\,(\texttt{apl}) \qquad \frac{M \twoheadrightarrow M'}{N(M) \twoheadrightarrow N(M')}\,(\texttt{apr}) \qquad \frac{M \twoheadrightarrow M'}{x^\sigma.M \twoheadrightarrow x^\sigma.M'}\,(\texttt{la})$$

We shall write $\twoheadrightarrow^*$ for the reflexive, transitive closure of $\twoheadrightarrow$. The λ-term M is said to be in *$\beta\eta$ normal form* if there is no λ-term N for which $M \twoheadrightarrow N$, or equivalently if for every such N we have $M \twoheadrightarrow^* N$ implies $M = N$. Of course the Church Rosser and strong normalisation properties hold: A consequence is that every λ-term M has a *unique* normal form. This will be the final λ-term in any maximal sequence of the form $M_1 \twoheadrightarrow M_2 \twoheadrightarrow \ldots \twoheadrightarrow N$. We can define an equivalence relation on the collection of λ-terms, which is given by the reflexive, symmetric, transitive closure of $\twoheadrightarrow$. We refer to this as *$\beta\eta$ equality*. If two λ-terms are $\beta\eta$ equal, written $M =_{\beta\eta} M'$, then M and M' have the same simple type, and they have the same (unique) $\beta\eta$-normal form. With this observation, we shall define the *canonical representative* of the $\beta\eta$ equivalence class of the λ-term M to be the $\beta\eta$-normal form of M.

We are now able to set up the Martin Löf-style metalanguage which will be used to present the FIX-logic. An *abstract syntax signature* Σ is specified by a pair (GAr, Con). Here, GAr is a collection of *ground arities* and if we view GAr as a collection of ground types, then we shall refer to the simple types generated from GAr as *arities*. Con is a collection of *metaconstants*, which are tagged with an arity: Formally a metaconstant consists of a pair (c, a) where c is a formal symbol and a is an arity, but we shall sometimes write this as c^a or even just c if the arity is clear. An *abstract syntax* is the collection of $\beta\eta$ equivalence classes of λ-terms generated by the above data. An *expression* of the abstract syntax is a $\beta\eta$ equivalence class.

Associated with a FIX- signature is a collection of *raw terms* and *raw propositions*. Each raw term (proposition) will be an expression of a certain Martin Löf-style abstract syntax. We define an abstract syntax signature, $\Sigma(Sg) = (GAr, Con)$, from which we construct the raw terms and propositions. GAr is the two point set $\{\textsc{term}, \textsc{prop}\}$. Con consists of the following elements: There is a metaconstant $(f, \textsc{term}^n \to \textsc{term})$ for each n-ary basic function symbol and a metaconstant $(R, \textsc{term}^n \to \textsc{prop})$ for each n-ary basic relation symbol. There are also metaconstants representing the simply typed lambda calculus with finite (co)products and so on for the remaining term syntax, and metaconstants representing equality, truth, falsity, conjunction and so on for the remaining proposition syntax. For example, for each type α there is a metaconstant $(\lambda_\alpha, (\textsc{term} \Rightarrow \textsc{term}) \Rightarrow \textsc{term})$ for functional abstraction; because the arity is well known, we normally just write λ_α for this. There would also be a metaconstant $(\mathsf{Inl}_\beta, \textsc{term} \Rightarrow \textsc{term})$ representing left coproduct insertion, and so on. See [1]. Finally, Con also contains a stock of *object level variables*, each of arity TERM.

The raw terms are exactly the *closed* expressions of the abstract syntax generated by $\Sigma(Sg)$ which are of arity TERM and the raw propositions are the *closed* expressions which are of arity PROP.

For example, suppose there is a FIX-term $\Gamma, x{:}\,\alpha \vdash F(x){:}\,\beta$ (so F is a term of the metalanguage for which $F \in \textsc{term} \Rightarrow \textsc{term}$, $x \in \textsc{term}$ is an object level variable (x is a constant of the metalanguage!) and $F(x)$ is application in the metalanguage). The functional abstraction is $\Gamma \vdash \lambda_\alpha(F){:}\,\alpha \Rightarrow \beta$, where $\lambda_\alpha(F)$ is again application in the metalanguage.

References

[1] R. L. Crole. *Programming Metalogics with a Fixpoint Type*. PhD thesis, Computer Laboratory, University of Cambridge, 1991.

[2] R. L. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. Cambridge University Press, 1993.

[3] 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.

[4] R.L. Crole and A.M. Pitts. New foundations for fixpoint computations: FIX hyperdoctrines and the FIX logic. *Information and Computation*, 98:171–210, 1992.

[5] M. Dummett. *Elements of Intuitionism*. Oxford University Press, 1977.

[6] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Foundations of Computing. MIT Press, 1992.

[7] 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.

[8] A. M. Pitts. Evaluation logic. In G. Birtwistle, editor, *IVth Higher Order Workshop, Banff 1990*, Workshops in Computing, pages 162–189. Springer-Verlag, Berlin, 1991.

[9] A. M. Pitts and I. Stark. Properties of higher order functions that dynamically create local names, or: What's new? In *Proc. International Symp. on Math. Foundations of Computer Science*. Springer-Verlag, 1993.

[10] G.D. Plotkin. Call by name, call by value and the λ calculus. *Theoretical Computer Science*, 1:125–129, 1975.

[11] G.D. Plotkin. L.C.F considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[12] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI–FN 19, Department of Computer Science, University of Aarhus, Denmark, 1981.

[13] G.D. Plotkin. Denotational semantics with partial functions. Unpublished lecture notes from CSLI summer school, 1985.

[14] G. Winskel. *The Formal Semantics of Programming Languages*. Foundations of Computing. The MIT Press, Cambridge, Massachusetts, 1993.