Correspondence between Operational and Denotational Semantics

C.-H. L. Ong*

Oxford University Computing Laboratory

Abstract

This course introduces the operational and denotational semantics of PCF and examines the relationship between the two.

Topics:

Syntax and operational semantics of PCF, Activity Lemma, undefinability of parallel or; Context Lemma (first principles proof) and proof by logical relations

Denotational semantics of PCF induced by an interpretation; (standard) Scott model, adequacy, weak adequacy and its proof (by a computability predicate)

Domain Theory up to SFP and Scott domains; non full abstraction of the standard model, definability of compact elements and full abstraction for PCFP (PCF + parallel or), properties of order-extensional (continuous) models of PCF, Milner's model and Mulmuley's construction (excluding proofs)

Additional topics (time permitting): results on pure simply-typed lambda calculus, Friedman's Completeness Theorem, minimal model, logical relations and definability, undecidability of lambda definability (excluding proof), dI-domains and stable functions

*Homepage: `http://www.comlab.ox.ac.uk/oucl/people/luke.ong.html`

Contents

1 Historical Remarks **4**

2 Syntax of PCF **6**

2.1 Types 6

2.2 Terms 6

2.3 Typing rules for PCF 7

2.4 Term substitution versus context substitution 8

3 Operational Semantics and PCF Type Theory **10**

3.1 Plotkin-style transition relation 10

3.2 Parallel or and the Activity Lemma 11

3.3 Martin-Löf style evaluation relation 12

3.4 PCF Type Theory 13

3.5 Call-by-Value PCF 14

3.6 Turing completeness 14

4 Observational Equivalence and the Context Lemma **16**

4.1 Observational and applicative preorders 16

4.2 The Context Lemma 17

4.3 Another proof of the Context Lemma 18

5 Denotational Semantics: the Standard Model **21**

5.1 What is a model of PCF? 21

5.2 The standard model 21

5.3 Denotational semantics induced by an interpretation 22

6 Adequacy and Weak Adequacy **25**

6.1 Adequacy and weak adequacy 25

6.2 A computability argument 26

6.3 Yet another proof of the Context Lemma: logical relations 29

7 A Primer on SFP Domains **32**

7.1 Scott domains 32

7.2 Characterizations of SFP domains 33

8 Full Abstraction for PCFP **34**

8.1 The standard model is not fully abstract for PCF 34

8.2 PCFP: PCF plus Parallel Or 34

8.3 Definability of compact elements 35

8.4 Inequational full abstraction 38

8.5 A restrictive approach to full abstraction 39

9 Full Abstraction and Definability **41**

10 Simply-Typed Lambda Calculus **44**

10.1 Friedman's Completeness Theorem 44

10.2 Logical relations and definability 44

10.3 Undecidability of lambda definability 44

10.4 Minimal model 44

10.5 Loader's decidability result 44

11 dI-domains and stable functions **44**

A What is a model of PCF? **45**

B A Sample Miniproject **48**

1 Historical Remarks

In the early 1940's Gödel considered a notion of primitive recursive functionals of finite type, which we now call Gödel's System **T**, in connection with what came to be known as the Dialectica Interpretation [7, 8]. Gödel presented his results as a contribution to a *liberalised* version of Hilbert's programme.[1] Gödel's work was later extended to the Bar Recursive Functionals by Spector [27] who used them to give a constructive consistency proof for classical analysis. However the first full-blown generalization of ordinary recursion theory to higher types was made by Kleene in the late 1950's (see [11, 12] for a formulation in terms of computation schemes). In Kleene's theory, a notion of partial recursive function of higher type is defined over a total type structure. Recursion is introduced by a computation scheme which essentially encapsulates the Second Recursion Theorem. In this theory, partial recursive functions are not closed under substitution, and a natural formulation of the First Recursion Theorem fails. Kleene exhibited these features in [12] where he also observed in passing that a theory involving application of (partial) functions to partial functions might be possible.

An attempt was made by Platek in his thesis [19] to develop a recursion theory on partial functions of higher type which avoids the problems of the Kleene theory. The type structure Platek considers is that of hereditarily order-preserving functions over what we now call the flat CPO of natural numbers. Platek couched his thesis in terms of computation schemes, and recursion is introduced by a scheme amounting to the First Recursion Theorem. The essentials of the theory are definition by cases and least fixed points, and Platek does introduce a λ-calculus formulation which can be regarded as a precursor to PCF.

The formal system PCF was introduced by Scott in a famous paper [26] which has remained an unpublished manuscript for a long time until its recent appearance in the Böhm Festschrift [26]. The syntax[2] of PCF is simple enough: it is essentially the simply-typed λ-calculus augmented by general recursion in the form of a fixed point operator at every type, and further augmented by basic arithmetic. Scott intended PCF to be a "logical calculus (or algebra)" for studying computability and other logical properties of programs by using type theory [3].

A major theme of Scott's work is the relationship between the *logical* types which are the higher types, and the data types which are the ground and first-order types. The former is used to study the latter; the theory of data types requires the higher-type objects — the computable functional — for its formalization, as emphasized by Scott. Of course, Scott had in mind a semantics in terms of what we now call Scott-continuous functions and for this interpretation, a quite straightforward operational semantics is appropriate: we can think of all the computations as being finite. Hence, the theory introduced by Scott is in principle implementable and has been the focus of much attention in Computer Science. Another theme in Scott's

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

[2] In [26] Scott considered a version of PCF based on typed **S**− and **K**-combinators.

paper is completeness about which he asked several questions. One such question concerns the power of expression of the language with respect to the continuous function space model; in a word, definability. What came to be known as the full abstraction problem for PCF was thus adumbrated: Scott observed that parallel or is not definable in the language and that an implementation on a "sequential machine" would require a dovetailing strategy. System **T** and PCF (as a formal system in the sense of Scott's original presentation as opposed to a programming language) are similar in various ways, though there is an important difference: Scott's approach admits partial functions whereas Gödel's is only concerned with total functions.

2 Syntax of PCF

In [23] Plotkin presented PCF explicitly as a programming language and studied the relationship between its operational semantics and denotational semantics.

2.1 Types

PCF types or simply ***types***, ranged over by A, B, C, etc., are defined as follows:

$$\begin{array}{rcll} A & ::= & \iota & \text{natural number} \\ & | & o & \text{boolean} \\ & | & A \Rightarrow A & \text{function} \end{array}$$

We use α to range over ground types ι and o. Standardly $\Rightarrow$ associates to the right so that $A \Rightarrow B \Rightarrow C$ is parsed as $A \Rightarrow (B \Rightarrow C)$. Every PCF type can be expressed uniquely as

$$A_1 \Rightarrow A_2 \cdots \Rightarrow A_n \Rightarrow \alpha$$

(for some $n \geqslant 0$) which we abbreviate as $(A_1, A_2, \cdots, A_n, \alpha)$; in case $n = 0$, we shall write it simply as α. E.g. $\iota \Rightarrow (\iota \Rightarrow \iota)$ abbreviates to (ι, ι, ι) and $((\iota \Rightarrow \iota) \Rightarrow (\iota \Rightarrow \iota)) \Rightarrow \iota$ abbreviates to $(((\iota, \iota), (\iota, \iota)), \iota)$. The *rank* of a type is defined as follows

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

The rank measures how "higher-order" a type is. We call A an n-th order type if $\mathsf{rank}(A) = n$. Many of the mathematical difficulties associated with higher-order objects stem from nesting on the left of $\Rightarrow$.

2.2 Terms

There are four groups of PCF terms:

1. basic arithmetic
2. definition by cases or conditionals
3. recursion (fixed point operator)
4. lambda calculus.

The first two are PCF constants, or simply ***constants***, as follows:

n	:	ι	numerals, for each natural number $n \geqslant 0$
t, f	:	o	booleans
succ	:	$\iota \Rightarrow \iota$	successor
pred	:	$\iota \Rightarrow \iota$	predecessor
zero	:	$\iota \Rightarrow o$	test for zero
cond^ι	:	$o \Rightarrow \iota \Rightarrow \iota \Rightarrow \iota$	integer conditional
cond^o	:	$o \Rightarrow o \Rightarrow o \Rightarrow o$	boolean conditional

Each constant has a prescribed type as indicated above. We let c range over constants. *Candidate terms* of PCF are defined by recursion as follows:

s	$::=$	c	constant
	$\mid$	x	variable
	$\mid$	(st)	application
	$\mid$	$(\lambda x^A.s)$	λ-abstraction
	$\mid$	$\mathbf{Y}^A(s)$	Y-term.

where x ranges a given set of variables $Var = \{ x, y, z, \cdots \}$. We shall often omit the type labels and write $\mathbf{Y}^A(-)$ and cond^ι (or cond^o) simply as $\mathbf{Y}(-)$ and cond respectively. Standardly application associates to the left, so that $st_1 \cdots t_n$ abbreviates $(\cdots((st_1)t_2)\cdots t_n)$. We routinely omit as many parentheses as we safely can.

2.3 Typing rules for PCF

Typing judgements have the form

$$? \quad \vdash \quad s : A$$

where $?$ is a *typing context* which is a finite sequence of the form

$$x_1 : A_1, \cdots, x_n : A_n$$

such that the variables x_is are all distinct. We let $?$ and Δ range over typing contexts. We read $? \vdash s : A$ as "in the context $?$, the term s has type A". We say that a term candidate s is *well-typed* just in case $? \vdash s : A$ is provable for some $?$ and A (in which case we say that s has type A). *Valid typing judgements* are defined by induction over the following rules:

$$\text{(cst)} \qquad ? \vdash c : A \quad (c \text{ has prescribed type } A)$$

$$\text{(var)} \qquad x_1 : A_1, \cdots, x_n : A_n \vdash x_i : A_i \qquad \text{(fix)} \qquad \frac{? \vdash s : A \Rightarrow A}{? \vdash \mathbf{Y}^A(s) : A}$$

$$\text{(app)} \qquad \frac{? \vdash s : A \Rightarrow B \qquad ? \vdash t : A}{? \vdash (st) : B} \qquad \text{(abs)} \qquad \frac{?, x : A \vdash s : B}{? \vdash (\lambda x^A.s) : A \Rightarrow B}.$$

By a PCF term, or simply ***term***, we mean a well-typed term candidate. The notions of free and bound variables are standard; a *closed* term is defined to be a term that has no free variables. We shall use $\equiv$ to mean syntactic equality (modulo renaming of bound variables) of terms.

2.4 Term substitution versus context substitution

We write *term substitution* as $s[t/x]$ which means "in s, substitute the term t for every free occurrence of x", renaming bound variables where necessary to avoid variable capture[3]. For example $(\lambda x.xz)[zx/z]$ is $\lambda y.y(zx)$ and not $\lambda x.x(zx)$. Henceforth whenever we write $s[t/x]$ we shall assume that both s and t are well-typed, and that t and x have the same type. By a ***program*** we mean a closed term of ground type.

PCF contexts are just terms that have "holes" (represented by *hole-variables* $X, Y, Z, \cdots$) as subterms. ***Contexts***, ranged over by C, D, E, etc., are defined by recursion as follows:

$$C \quad ::= \quad X \quad | \quad c \quad | \quad x \quad | \quad (\lambda x^A.C) \quad | \quad (CC) \quad | \quad \mathbf{Y}^A(C)$$

where x ranges over Var, c over the PCF constants and X over a fixed set of hole-variables $\{X, Y, Z, \cdots\}$. We shall often write C as $C[X_1, \cdots, X_n]$ if the holes of C are taken from the set $\{X_1, \cdots, X_n\}$. E.g. we may write $C \equiv (\lambda y^o.Y)((\lambda x^{\iota \Rightarrow \iota}.x)X)$ as $C[X, Y]$, and $D \equiv X(\lambda x^\iota.X(\lambda y^\iota.x))$ as $D[X, Z]$. *Context substitution* is defined recursively as follows.

$$C[p_1, \cdots, p_n] \quad \stackrel{\text{def}}{=} \quad \begin{cases} C & \text{if } C \text{ is a variable or constant} \\ p_i & \text{if } C \text{ is the hole } X_i \\ (C_1[p_1, \cdots, p_n]C_2[p_1, \cdots, p_n]) & \text{if } C = (C_1C_2) \\ (\lambda x^A.C'[p_1, \cdots, p_n]) & \text{if } C = (\lambda x^A.C') \\ \mathbf{Y}^A(C'[p_1, \cdots, p_n]) & \text{if } C = (\mathbf{Y}^A(C')). \end{cases}$$

The crucial difference between context substitution and term substitution is that the former is *capture permitting* (variables may be bound as a result) whereas the latter is *capture avoiding*

[3]We adopt the Variable Convention in Barendregt's book [1]: bound variables are renamed afreshed whenever necessary to ensure that there is no name clash with either another bound variable or a free variable that occurs in some term in the same context.

(variables are never bound). For example, take $C[X]$ to be $\lambda x.Xx$ then $C[xy] \equiv \lambda x.(xy)x$ so that x is bound as a result; but $C[w][(xy)/w] \equiv \lambda z.(xy)z \equiv \lambda z'.(xy)z'$ (the bound variable x in $C[w] \equiv \lambda x.wx$ is renamed to z to avoid capture as a result of the term substitution $(-)[xy/w]$).

Problems

2.1 (i) Show that $\lambda x^{\iota}.xx$ is not well-typed.

(ii) Define a term of type $\iota \Rightarrow \iota$ that computes the factorial function.

2.2 Prove the following properties of the typing judgement:

(i) If $? \vdash s : A$ and x does not appear in $?$, then x is not free in s.

(ii) (*Determinacy*) If $? \vdash s : A$ and $\Delta \vdash s : A'$ and $?$ and Δ agree on the variables that occur free in s, then $A = A'$.

(iii) (*Permutation*) If $?, x : A, y : B, \Delta \vdash s : C$ then $?, y : B, x : A, \Delta \vdash s : C$.

(iv) (*Weakening*) If $? \vdash s : A$ then for any y not in $?$, we have $?, y : B \vdash s : A$.

(v) (*Contraction*) If $?, x : A, y : A \vdash s : B$ is provable then, for any z not occurring in $?$, so is $?, z : A \vdash s[z/x][z/y] : B$.

(vi) (*Substitution*) If $?, x : A \vdash s : B$ and $\Delta \vdash t : A$, and Δ and $?$ are disjoint, then $?, \Delta \vdash s[t/x] : B$.

3 Operational Semantics and PCF Type Theory

We evaluate PCF programs by a normal order reduction strategy. The strategy permits only *weak* reduction i.e. no reduction is allowed "under a lambda".

3.1 Plotkin-style transition relation

We first define the ***evaluation contexts*** by recursion as follows:

$$E \quad ::= \quad X \quad | \quad (Es) \quad | \quad (cE)$$

where s ranges over terms and c over constants. By construction every evaluation context has exactly one hole. The ***one-step reduction*** $>$ is a binary relation over PCF terms defined by induction over the following axioms

$$(\lambda x^A.s)t > s[t/x^A] \quad \mathbf{Y}^A(s) > s(\mathbf{Y}^A(s)) \quad \mathsf{succ}\, n > n+1$$

$$\mathsf{cond}^\alpha\, \mathsf{t} > \lambda x^\alpha y^\alpha.x \quad \mathsf{cond}^\alpha\, \mathsf{f} > \lambda x^\alpha y^\alpha.y \quad \mathsf{zero}\, 0 > \mathsf{t}$$

$$\mathsf{pred}\, n+1 > n \quad \mathsf{pred}\, 0 > 0 \quad \mathsf{zero}\, n+1 > \mathsf{f}$$

and over the rule

$$\frac{s > s'}{E[s] > E[s']}.$$

The unique hole of an evaluation context is intended to be the site where the reduction really takes place. We refer to (instances of) the left hand side of the nine axioms above as ***redexes***. By a ***value*** we shall mean either a λ-abstraction or a constant. For any term s, we write $s \not> $ to mean $\neg[\exists s'.s > s']$. E.g. $2 \not>$ and $\mathsf{cond}\, x \not>$.

Remark 3.1 By convention whenever we write $s > t$ we shall assume that there is a fixed context $?$, such that $? \vdash s : A$ provable, which determines the types of the free variables of s. We stress here that the same assumption applies to the other binary relations over terms to be introduced in the sequel e.g. $\Downarrow, \approx^{\mathrm{obs}}, \sqsubseteq^{\mathrm{app}}_{\sim}$, etc. Instead of introducing a binary relation $>$ over terms, we could adopt a formal approach and define a reduction-in-context judgement of the form $? \vdash s > t : A$, but we prefer a simpler notation.

Exercise 3.1 Check the following:

(i) For any term s, $s \not> $ if and only if s is a constant, or an abstraction, or of the form $E[x]$ for some evaluation context E.

(ii) $>$ satisfies *subject reduction*: if s has type A and $s > t$ then t has type A.

(iii) $>$ is *substitutive*: if $s > t$ then for any x and any r, we have $s[r/x] > t[r/x]$.

(iv) $>$ is deterministic i.e. if $s > t$ and $s > t'$ then $t \equiv t'$.

Proposition 3.2 *For any term s such that $s > s'$ for some s', there are a unique evaluation context $E[X]$ and a unique redex-subterm Δ of s such that $s \equiv E[\Delta]$. Hence $>$ is deterministic i.e. if $s > t$ and $s > t'$ then $t > t'$.* □

We shall call the redex Δ in (ii) above the ***active redex*** of s.

We define

$$\begin{aligned}
\gg &\stackrel{\text{def}}{=} \text{reflexive, transitive closure of } > \\
s \gg_n s' &\stackrel{\text{def}}{=} \exists\{\, s_0, \cdots, s_n \,\}.s \equiv s_0 \wedge s_n \equiv s' \wedge \forall 1 \leqslant i < n.s_i > s_{i+1} \\
s\uparrow &\stackrel{\text{def}}{=} \exists\{\, s_i : i \geqslant 1 \,\}.\forall i.s_i > s_{i+1} \\
s \downarrow_n v &\stackrel{\text{def}}{=} s \gg_n v \wedge v \not> \\
s \downarrow v &\stackrel{\text{def}}{=} \exists n.s \gg_n v \\
s\downarrow &\stackrel{\text{def}}{=} \exists v.s \downarrow v
\end{aligned}$$

where we let v range over values. E.g. for each A, $\mathbf{Y}^A(\lambda x^A.x)\uparrow$. Henceforth we write Ω^A as a shorthand for the term $\mathbf{Y}^A(\lambda x^A.x)\uparrow$.

3.2 Parallel or and the Activity Lemma

Certain parallel operations such as parallel or are not PCF definable. Scott already observed in 1969 (see [26]) that a kind of dovetailing strategy would be needed in order to compute (or simulate) parallel or in PCF.

Proposition 3.3 *Parallel or is not definable in PCF: there is no closed term $p : o \Rightarrow o \Rightarrow o$ that satisfies:*

$$\begin{array}{ll}
p\,\Omega\,\Omega\uparrow & p\,\mathsf{t}\,\Omega \downarrow \mathsf{t} \\
p\,\Omega\,\mathsf{t} \downarrow \mathsf{t} & p\,\mathsf{f}\,\mathsf{f} \downarrow \mathsf{f}
\end{array}$$

□

To prove the proposition, we need the following lemma.

Lemma 3.4 (Activity) *Suppose for some closed r of type A, $C[r, \Omega^B]$ is a program and $C[r, \Omega]\downarrow$. Then*

1. *either for every closed term s of type A, $C[s, \Omega]\downarrow$ and so also $C[s, t]\downarrow$ for any closed t*

2. *or there is a context $D[X,Y]$ such that for any closed s and t (of the appropriate types), we have $C[s,t] \gg D[s,t]$ such that*

   2.1 *either $D[s,t] \equiv s$*

   2.2 *or the active redex of $D[s,t]$ is an active redex of (a copy of) s*

   2.3 *or s is a value and the active redex of $D[s,t] \equiv su$ for some u, or cs for some constant c.*

Note that 2.1 and 2.2 are not mutually exclusive.

Proof We have seen that $>$ is substitutive. Take variables x and y of types A and B respectively. It is impossible that $C[x,y]\uparrow$ because of the supposition. Hence there is some context $D[X,Y]$ such that $C[x,y] \gg D[x,y] \not>$. Now $D[x,y]$ is either

(i) a constant, or

(ii) an abstraction, or

(iii) $E[x]$ for some evaluation context E (we reject the case of $E[y]$ because of the supposition).

Cases (i) and (ii) correspond to case 1. Consider case (iii). If $D[x,y] \equiv x$ then we have case 2.1 above; otherwise, for any closed s and t of the appropriate types, $D[s,t]$, which is either $F[s]u$ or $cF[s]$ for some evaluation context F which depends on s and t, has an active redex Δ. By structural induction on F, we have either case 2.2 or case 2.3. □

We can now prove Proposition 3.3. Suppose, for a contradiction, for some context $C[X,Y]$, $\lambda x^o y^o.C[x,y]$ satisfies the required properties. Since $C[\mathsf{t},\Omega]\downarrow$ and $C[\Omega,\Omega]\uparrow$, by the Proposition, there are three cases for $C[\Omega,\mathsf{t}]$. Case 2.3 is not possible, and cases 2.1 and 2.2 contradict $C[\Omega,\mathsf{t}]\downarrow$.

3.3 Martin-Löf style evaluation relation

We may present the operational semantics of PCF equivalently in terms of a Martin-Löf style evaluation relation (see *e.g.* [14]) or "big-step" relation. Formally, we define a relation $\Downarrow$ between closed terms and values by induction over the following rules. We read $s \Downarrow v$ as "the closed term s evaluates to the value v".

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

$$\frac{s \Downarrow 0}{\mathsf{pred}\, s \Downarrow 0} \qquad \frac{s \Downarrow \mathsf{t}}{\mathsf{cond}^\alpha\, s \Downarrow \lambda x^\alpha y^\alpha.x} \qquad \frac{s \Downarrow \mathsf{f}}{\mathsf{cond}^\alpha\, s \Downarrow \lambda x^\alpha y^\alpha.y}$$

$$\frac{s \Downarrow 0}{\mathsf{zero}\, s \Downarrow \mathsf{t}} \qquad \frac{s \Downarrow n+1}{\mathsf{zero}\, s \Downarrow \mathsf{f}} \qquad \frac{p[t/x] \Downarrow v}{(\lambda x.p)t \Downarrow v}$$

$$\frac{s\mathbf{Y}(s) \Downarrow v}{\mathbf{Y}(s) \Downarrow v} \qquad \frac{s \Downarrow v \quad vt \Downarrow v'}{st \Downarrow v'}$$

We check that $\Downarrow$ is deterministic i.e. if $s \Downarrow v_1$ and $s \Downarrow v_2$ then $v_1 \equiv v_2$.

The two presentations coincide.

Proposition 3.5 (Agreement) *For any program s, $s \Downarrow v$ if and only if $s \downarrow v$.* □

3.4 PCF Type Theory

The operational semantics for PCF encapsulates a deterministic reduction strategy for the programming language; but also it reflects an intuitive understanding of the meaning of the terms. In this view the reductions are justified as the replacement of a term by an equal term. Thus the intuitive semantics can be given expression in an equational theory. In the case of PCF this amounts to a type theory related to Scott's original formulation [26] in 1969. Our type theory for PCF is given as follows. There are two kinds of judgements:

- *typing* $\quad ? \vdash s : A$
- *equality* $\quad ? \vdash s = t : A$

We have already seen the rules for the typing judgements. *Valid equality judgements* are defined by induction over the following rules:

$$? \vdash s = s : A \qquad \frac{? \vdash s = t : A}{? \vdash t = s : A}$$

$$\frac{? \vdash s = t : A \qquad ? \vdash t = u : A}{? \vdash s = u : A} \qquad \frac{? \vdash s = t : A \qquad ?, \Delta \vdash C[s] : B}{?, \Delta \vdash C[s] = C[t] : B}$$

$$? \vdash (\lambda x : A.s)t = s[t/x] : B \qquad ? \vdash \lambda x : A.sx = s : A \Rightarrow B \ (x \text{ not free in } s)$$

$$? \vdash s(\mathbf{Y}^A(s)) = \mathbf{Y}^A(s) : A \qquad ? \vdash \mathsf{succ}\ n = n + 1 : \iota$$

$$? \vdash \mathsf{pred}\ n + 1 = n : \iota \qquad ? \vdash \mathsf{pred}\ 0 = 0 : \iota$$

$$? \vdash \mathsf{cond}^\alpha\ \mathsf{t} = \lambda x^\alpha y^\alpha.x : \alpha \Rightarrow \alpha \Rightarrow \alpha \qquad ? \vdash \mathsf{cond}^\alpha\ \mathsf{f} = \lambda x^\alpha y^\alpha.y : \alpha \Rightarrow \alpha \Rightarrow \alpha$$

$$? \vdash \mathsf{zero}\, 0 = \mathsf{t} : o \qquad ? \vdash \mathsf{zero}\, n + 1 = \mathsf{f} : o$$

We require the first four axioms (namely, reflexivity, β, η and $\mathbf{Y}$) to satisfy the *typability side condition* that the expressions on both sides of the $=$ symbol have the declared type in the given typing context. This has the consequence that whenever $? \vdash s = t : A$ is provable, it has the force that $? \vdash s : A$ and $? \vdash t : A$ are also provable.

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

Proposition 3.6 *For any programs s and t, if s and t are equal in the PCF type theory then for any ground value v*

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

Proof A more or less straightforward application of the Church-Rosser Theorem and a standardization theorem following, for example, the treatment in [21]. □

Corollary 3.7 *For any program s in the type theory and ground value v, $s = v$ is provable in the type theory if and only if $s \Downarrow v$.* □

3.5 Call-by-Value PCF

Call-by-Value (or CBV for short) PCF has the same terms as PCF, which we shall refer to as Call-by-Name PCF whenever it is necessary to distinguish it from the CBV version. The one-step reduction relation $>$ is defined by the same nine redex rules except that the (β)-axiom is replaced by

$$(\lambda x^A.p)v > p[v/x]$$

where v ranges over values. The CBV evaluation contexts are defined by recursion as

$$E \quad ::= \quad X \quad | \quad (Es) \quad | \quad (vE)$$

where v ranges over values (i.e. λ-abstractions and constants).

Exercise 3.2 Define a big-step evaluation relation for CBV PCF, and prove that it agrees with the small-step transition relation in the sense of Proposition 3.5.

3.6 Turing completeness

Take an n-argument numeric (partial) function $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$. We say that f is ***strongly definable*** by a term $s : (\underbrace{\iota, \cdots, \iota}_{n+1})$ just in case for any n-tuple $(m_1, \cdots, m_n)$

(i) $f\vec{m}$ is undefined if and only if $s\vec{m}\uparrow$, and

(ii) $f\vec{m} = n$ if and only if $s\vec{m} \downarrow n$.

Proposition 3.8 *PCF is Turing complete. That is to say, every partial recursive function is strongly definable.* □

Problems

3.1 Construct a PCF closed term that computes the equality function of type $\iota \Rightarrow \iota \Rightarrow o$.

3.2 Construct PCF closed terms $\mathsf{l\text{-}add}, \mathsf{r\text{-}add} : (\iota, \iota, \iota)$ that compute the addition function such that $\mathsf{l\text{-}add}$ examines the left argument before the right and $\mathsf{r\text{-}add}$ does the reverse.

3.3 Suppose $C[\Omega^A]\downarrow$. Is it true that for any s of type A, $C[s]\downarrow$? Justify your answer.

3.4 Prove Proposition 3.2(ii): For any closed s which is not a value, there are a unique evaluation context $E[X]$ and a unique redex-subterm Δ of s such that $s \equiv E[\Delta]$.

3.5 Prove the Agreement Proposition 3.5: For any program s, $s \Downarrow v$ if and only if $s \downarrow v$.

[Hint: define a family of evaluation relations $\Downarrow_n$, one for each $n \geqslant 0$, between closed terms and values by mutual rule induction where $s \Downarrow_n v$ means intuitively "s evaluates to v in n steps".]

3.6 Prove that there is no closed term $F : (o \Rightarrow o \Rightarrow o) \Rightarrow o$ such that for any closed $p : o \Rightarrow o \Rightarrow o$ such that $p\ s\ t\downarrow$ for some s and t

(i) if $p\ \Omega^o\ t\uparrow$ for all t then $Fp \downarrow 1$

(ii) if $p\ s\ \Omega^o\uparrow$ for all s then $Fp \downarrow 2$.

Note that F is able to ascertain which of the two input arguments of p is evaluated (by p) first.

3.7 Prove Proposition 3.8: every partial recursive function is strongly definable in PCF.

3.8 How would you reconcile the undefinability of parallel or (which can be computed by an effective procedure) with the Turing completeness of PCF?

4 Observational Equivalence and the Context Lemma

There is a purely "symbol pushing", operational notion of program equivalence called *observational equivalence* which programmers know well: two program fragments are equivalent if they can always be *interchanged* without affecting the observable outcome of the computation. Formally suppose s and t are terms of the same type. We say that s and t are ***observationally equivalent***, written $s \approx^{\text{obs}} t$, just in case for any context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, and for any value v, $C[s] \Downarrow v$ if and only if $C[t] \Downarrow v$.

4.1 Observational and applicative preorders

Definition 4.1 Suppose s, t are terms of the same type $A = (A_1, \cdots, A_n, \alpha)$.

(i) We define the *observational preorder* $\sqsubseteq^{\text{obs}}$ as follows: we say that s ***observationally approximates*** t, written $s \sqsubseteq^{\text{obs}} t$, if for any context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, and for any value v, if $C[s] \Downarrow v$ then $C[t] \Downarrow v$. Thus $s \approx^{\text{obs}} t$ if and only if $s \sqsubseteq^{\text{obs}} t$ and $t \sqsubseteq^{\text{obs}} s$.

(ii) Suppose s and t are closed terms. We say that s ***applicatively approximates*** t, written $s \sqsubseteq^{\text{app}} t$, if for any closed terms $p_1 : A_1, \cdots, p_n : A_n$, and for any value v, if $sp_1 \cdots p_n \Downarrow v$, then $tp_1 \cdots p_n \Downarrow v$. We define $s \approx^{\text{app}} t$ to be $s \sqsubseteq^{\text{app}} t$ and $t \sqsubseteq^{\text{app}} s$.

It is straightforward to check that both $\sqsubseteq^{\text{obs}}$ and $\sqsubseteq^{\text{app}}$ are preorders (i.e. reflexive and transitive).

Example 4.2 (i) We can readily see that $\Omega^{\iota \Rightarrow \iota} \approx^{\text{app}} \lambda x^\iota.\Omega^\iota$. It is also true that $\Omega^{\iota \Rightarrow \iota} \approx^{\text{obs}} \lambda x^\iota.\Omega^\iota$. For a proof, take any context C such that $C[\Omega^{\iota \Rightarrow \iota}]$ (and so also $C[\lambda x^\iota.\Omega^\iota]$) is a program. By Exercise 3.1(i), either $C[x]\uparrow$ or

(1) $C[x] \gg c$, or

(2) $C[x] \gg \lambda x^A.s$ for some s, or

(3) $C[x] \gg E[x]$ for some evaluation context E.

Cases (1) and (2) are trivial. In case (3), $E[x]$ is either X or $E^-[X]s$ or $cE^-[X]$. The first is impossible. Consider the second for an illustration. The active redex of $E[\Omega^{\iota \Rightarrow \iota}]$ must be $\Omega^{\iota \Rightarrow \iota}$ itself, so $C[\Omega]\uparrow$. The active redex of $E[\lambda x^\iota.\Omega^\iota]$ must be of the form $(\lambda x^\iota.\Omega^\iota)t$ for some t, and so, $C[\lambda x^\iota.\Omega^\iota]\uparrow$.

(ii) The reader is invited to check that $\mathsf{l\text{-}add} \approx^{\text{app}} \mathsf{r\text{-}add}$ and $\mathsf{l\text{-}add} \approx^{\text{obs}} \mathsf{r\text{-}add}$. Because of the quantification over contexts, it is in general quite hard to check directly if two given terms are observationally equivalent.

(iii) Verify that $x \not\approx^{\text{obs}} y$. Is it true that if s and t do not have the same set of free variables, then $s \not\approx^{\text{obs}} t$?

Remark 4.3 In the definition of $\approx^{\text{obs}}$, the qualification that C ranges only over those such that both $C[s]$ and $C[t]$ are programs is not redundant. The reason is that assuming $s \approx^{\text{obs}} t$, it does not necessarily follow that for any context, $C[s]$ is a program if and only if $C[t]$ is a program: consider $(\lambda x.2)y \approx^{\text{obs}} 2$ and take $C[X]$ to be X.

We state a useful property of the two equivalences: $\approx^{\text{obs}}$ and $\approx^{\text{app}}$ are ***PCF theories*** in the following sense. (See Problems 4.6 and 4.7 for hints for a proof.)

Theorem 4.4 *If* $? \vdash s = t : A$ *is provable then* $s \approx^{\text{obs}} t$ *and* $s \approx^{\text{app}} t$. □

4.2 The Context Lemma

We aim now to establish an important connexion between the two preorders. By an *applicative context*, we shall mean an evaluation context of the form $Xp_1 \cdots p_n$ where each p_i is a closed term. Suppose s and t are closed terms of the same type. We observe that $s \sqsubseteq^{\text{app}} t$ is equivalent to the statement:

> for any applicative context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, for any v, if $C[s] \Downarrow v$ then $C[t] \Downarrow v$.

Thus $s \sqsubseteq^{\text{obs}} t$ implies $s \sqsubseteq^{\text{app}} t$. In fact the converse is also true.

Theorem 4.5 (Context Lemma) *Suppose s and t are closed terms of the same type $A = (A_1, \cdots, A_n, \alpha)$. If $s \sqsubseteq^{\text{app}} t$ then $s \sqsubseteq^{\text{obs}} t$.*

Proof For $n \geqslant 0$ let $\mathbf{H}(n)$ be the statement:

> For any closed terms s and t of the same type, if $s \sqsubseteq^{\text{app}} t$ then for any context $C[X]$ such that $C[s]$ and $C[t]$ are programs, for any value v, if $C[s] \downarrow_n v$ then $C[t] \Downarrow v$.

We aim to prove $\forall n.\mathbf{H}(n)$ by induction on n. The base case $\mathbf{H}(0)$ is true, since $C[s] \downarrow_0 v$ implies that *either* $C[X] \equiv X$ and $s \equiv v$, and so, $t \Downarrow v$ as $s \sqsubseteq^{\text{app}} t$; *or* $C[X]$ is a ground value v, in which case, so is $C[t]$.

We establish the inductive case $\mathbf{H}(n+1)$ by a case analysis. Observe that $C[X]$ has one of the following syntactic shapes: for $m \geqslant 0$

(1) $cC_1[X] \cdots C_m[X]$ where c is a constant

(2) $(\lambda x^A.C_0[X])C_1[X] \cdots C_m[X]$ where $m \geqslant 1$

(3) $\mathbf{Y}^A(C_0[X])C_1[X] \cdots C_m[X]$

(4) $XC_1[X] \cdots C_m[X]$.

(1) We consider the case of $c \equiv \mathsf{cond}^{\iota}$ for an illustration. Suppose $\mathsf{cond}\, C_1[s]C_2[s]C_3[s] \downarrow_{n+1} v$. Then we must have $C_1[s] \downarrow_{l_1} \mathsf{t}$ or f. Suppose the former, we then have $C_2[s] \downarrow_{l_2} v$. Note that both $l_1, l_2 \leqslant n$. Hence, by the induction hypothesis, we have $C_1[t] \Downarrow \mathsf{t}$ and $C_2[t] \Downarrow v$, and so, $\mathsf{cond}\, C_1[t]C_2[t]C_3[t] \Downarrow v$. The argument is entirely similar for the case of $C_1[s] \downarrow_{l_1} \mathsf{f}$.

(2) Suppose $C[s] \equiv (\lambda x^A.C_0[s])C_1[s]\cdots C_m[s] \downarrow_{n+1} v$. Then

$$C[s] > C_0[s][C_1[s]/x^A]C_2[s]\cdots C_m[s] \downarrow_n v.$$

Let $D[X] \equiv C_0[X][C_1[X]/x^A]C_2[X]\cdots C_m[X]$. Since s and t are closed terms, $C[s] > D[s] \downarrow_n v$. By $\mathbf{H}(n)$, we infer that $D[t] \Downarrow v$; and so, $C[t] \Downarrow v$.

(3) Suppose $C[s] \equiv \mathbf{Y}^A(C_0[s])C_1[s]\cdots C_m[s] \downarrow_{n+1} v$. Define a new context

$$D[X] \quad \equiv \quad C_0[X]\mathbf{Y}^A(C_0[X])C_1[X]\cdots C_m[X].$$

We can infer that $C[s] > D[s] \downarrow_n v$. By $\mathbf{H}(n)$, we have $D[t] \Downarrow v$. Since $C[t] > D[t]$, we can conclude that $C[t] \Downarrow v$.

(4) Suppose $C[s] \equiv sC_1[s]\cdots C_m[s] \downarrow_{n+1} v$. Observe that s must have one of the following syntactic shapes:

(a) $c\vec{p}$ where c is a constant

(b) $(\lambda x.p)\vec{q}$,

(c) $\mathbf{Y}(p)\vec{q}$.

Set $D[X] \equiv sC_1[X]\cdots C_m[X]$, we then have $C[s] \equiv D[s] \downarrow_{n+1} v$. Note that if s has shape (a), (b), or (c) respectively, then $D[X]$ is a context of type (1), (2), or (3) respectivel). Applying $\mathbf{H}(n+1)$ to contexts of type (1), (2) or (3) as the case may be, we infer that $D[t] \Downarrow v$. But $D[t] \equiv sC_1[t]\cdots C_m[t]$, and so, since $s \sqsubseteq^{\mathrm{app}} t$, we have $tC_1[t]\cdots C_m[t] \equiv C[t] \Downarrow v$. $\square$

The Context Lemma tells us that applicative contexts alone are enough to determine the observational preorder (between closed terms). The proof we have presented is due to Berry [2]. Milner [15] has a similar result for a family of languages expressed in the formalism of Combinatory Logic.

4.3 Another proof of the Context Lemma

We present a proof due to Martin Hyland. Fix hole-variables X and Y. We define by simultaneous recursion:

$$\begin{array}{rcl} C_Y & ::= & Y \mid c \mid x \mid (\lambda x^A.C_Y) \mid (C_Y C_Y) \mid \mathbf{Y}^A(C_Y) \\ F & ::= & X \mid (FC_Y) \mid (cF) \end{array}$$

where c and x above range over constants and variables respectively. By construction $F[X,Y]$ has exactly one occurrence of X but finitely many occurrences of Y (and no other hole-variable);

in addition, for any term s, $F[X, s]$ is an evaluation context. Thus we call F a *pre-evaluation context*. Observe also that if $s \precsim^{\mathrm{app}} t$ then (by Problem 4.2) for any pre-evaluation context F and for any term u such that $F[s, u]$ and $F[t, u]$ are well-typed, we have $F[s, u] \precsim^{\mathrm{app}} F[t, u]$.

Suppose s and t are closed terms such that $s \precsim^{\mathrm{app}} t$. W.l.o.g. we can assume that s is an abstraction. Take any context C such that $C[s]$ and $C[t]$ are programs, and suppose $C[s]\Downarrow$. We aim, and it suffices (see Problem 4.3), to prove $C[t]\Downarrow$. By assumption, there are finitely many distinct pre-evaluation context $F_1, \cdots, F_n$ (say) such that

(i) $C[s] \gg F_n[s, s] \gg \cdots \gg F_2[s, s] \gg F_1[s, s] \gg c$ for some ground value c, and

(ii) for some fresh variable x, for each $1 \leqslant i < n$, $F_{i+1}[s, x] \gg F_i[x, x]$, and $F_1[s, x] \gg c$.

Intuitively (ii) above says that for each i, the intermediate reduction steps between $F_{i+1}[s, s]$ and $F_i[s, s]$ do not "touch" a new copy of s (other than the one occupying the X-hole of F_{i+1}).

By (ii) and by the substitutivity of $>$ (see Exercise 3.1(iii)), we have $F_1[s, t]\Downarrow$. Since $s \precsim^{\mathrm{app}} t$, by the preceding observation, $F_1[s, t] \precsim^{\mathrm{app}} F_1[t, t]$, and so, we have $F_1[t, t]\Downarrow$. For the same reason as before, we have $F_2[s, t] \gg F_1[t, t]\Downarrow$, so also $F_2[s, t]\Downarrow$. Since $s \precsim^{\mathrm{app}} t$, we have $F_2[t, t]\Downarrow$. Thus inductively, we have $C[t] \gg F_n[t, t]\Downarrow$ as desired.

Problems

4.1 Prove from first principles that $\Omega^{\iota \Rightarrow \iota} \approx^{\mathrm{obs}} \lambda x^{\iota}.\Omega^{\iota}$, and $\Omega^{\iota} \approx^{\mathrm{obs}} \mathbf{Y}^{\iota}(\lambda x^{\iota}.\mathsf{succ}\, x)$.

4.2 (i) Prove from first principles that if $s \precsim^{\mathrm{app}} t$ then $E[s] \precsim^{\mathrm{app}} E[t]$ for any evaluation context E.

(ii) Suppose $s \precsim^{\mathrm{obs}} t$. Is it true that for any context C such that both $C[s]$ and $C[t]$ are well-typted, $C[s] \precsim^{\mathrm{obs}} C[t]$? Justify your answer.

4.3 For closed term u, we write $r\Downarrow$ to mean $\exists v.r \Downarrow v$. Suppose s and t are terms of the same type.

(i) Prove that $s \precsim^{\mathrm{obs}} t$ if and only if for any context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, if $C[s]\Downarrow$ then $C[t]\Downarrow$.

(ii) For closed terms s and t, is $s \precsim^{\mathrm{app}} t$ equivalent to "for any closed $\vec{p}$ (of the appropriate types), if $s\vec{p}\Downarrow$ then $t\vec{p}\Downarrow$"?

4.4 Consider the following variant notions of observational preorder defined on open terms. Suppose s and t are terms of the same type.

(i) *closure by abstraction*: $s \precsim^{\mathrm{abs}} t$ is said to hold if for any $x_1 : A_1, \cdots, x_n : A_n$ such that $\lambda\vec{x} : \vec{A}.s$ and $\lambda\vec{x} : \vec{A}.t$ are closed terms, and for any context $C[X]$ such that $C[\lambda\vec{x} : \vec{A}.s]$ and $C[\lambda\vec{x} : \vec{A}.t]$ are programs, if $C[\lambda\vec{x} : \vec{A}.s]\Downarrow$ then $C[\lambda\vec{x} : \vec{A}.t]\Downarrow$

(ii) *closure by substitution*: $s \sqsubseteq^{\text{sub}} t$ is said to hold if for all closing substitutions θ (i.e. s_θ and t_θ are closed terms of the same type) and for all contexts $C[X]$ such that $C[s_\theta]$ and $C[t_\theta]$ are programs, if $C[s_\theta]\Downarrow$ then $C[t_\theta]\Downarrow$.

Prove that $\sqsubseteq^{\text{obs}}$, $\sqsubseteq^{\text{abs}}$ and $\sqsubseteq^{\text{sub}}$ are equivalent to each other.

4.5 Propose a way by which $\sqsubseteq^{\text{app}}$ can be extended to a preorder between terms in general. Does the new preorder conincide with $\sqsubseteq^{\text{obs}}$?

4.6 (i) For any A and for any closed term s and t of type A, prove that

$$(\forall v. s \Downarrow v \iff t \Downarrow v) \quad \Longrightarrow \quad s \approx^{\text{obs}} t.$$

(ii) Is the converse true?

4.7 (i) Prove that $(\lambda x^A.s)t \approx^{\text{obs}} s[t/x]$, $\lambda x^A.px \approx^{\text{obs}} p$ (where x does not occur free in p), and $\mathbf{Y}^A(s) \approx^{\text{obs}} s\mathbf{Y}^A(s)$.

(ii) Prove that if $? \vdash s = t : A$ is provable then $s \approx^{\text{obs}} t$.

(iii) Is the converse of (ii) true?

4.8 Prove the Context Lemma for Call-by-Value PCF. Does the proof of Theorem 4.5 still work?

5 Denotational Semantics: the Standard Model

5.1 What is a model of PCF?

Since PCF is a typed functional language, an interpretation of the language should first provide a system of "domains" which gives an interpretation of the types, one domain for each type. The denotation of a PCF term is then given by a type-respecting valuation function which takes a term and an environment (giving denotations of the variables occurring free in the term) and returns a denotation of the term as an element of the appropriate domain. Following the style of denotational semantics, the valuation function should be defined compositionally.

Models of the simply-typed λ-calculus[4] are cartesian closed categories [13, 4]. The interpretation is standard: given a cartesian closed category, types are interpreted as objects, and terms as maps. An important feature of PCF is recursion in the form of a fixed point operator at every type. Here we appeal to Knaster-Tarski's Fixed point Theorem [30] to construct the fixed point of a continuous function from a complete partial order (CPO) to itself. Thus, combining the two, we seek a cartesian closed category of CPOs and continuous functions.

By a CPO, we mean a partially ordered set such that every directed subset has a least upper bound (lub). A function $f : D \longrightarrow E$ from a CPO D to another E is said to be *continuous* if $f(\bigsqcup X) = \bigsqcup\{\, f(x) : x \in X \,\}$ for all directed subsets X of D. Then the set $[D \Rightarrow E]$ of all continuous functions $D \longrightarrow E$ is itself a CPO under the induced pointwise ordering.

The question of what is a model of PCF is best addressed in abstract, categorical terms; see Appendix A for a concise account. Here we shall introduce a class of models, based on the cartesian closed category of CPOs and continuous functions, but we shall focus on one (canonical interpretation) first studied by Scott in 1969 [26]. Our approach shall be concrete, without any attempt to be general.

A ***collection of value domains*** for PCF is a family $\{\, D^A \,\}$ of CPOs, one for each type A, such that $D^{A \Rightarrow B} = [D^A \Rightarrow D^B]$. It is said to be *standard* if if D^ι is $\mathbb{N}_\bot$ the flat CPO of natural numbers, and D^o is $\mathbb{B}_\bot$ the flat CPO of booleans. An ***interpretation*** $\mathcal{I}$ of PCF is given by

- a collection $\{\, D^A \,\}$ of value domains
- a map $[\![-]\!]^{\mathcal{I}} : \mathbf{A} \longrightarrow \bigcup_A D^A$, where $\mathbf{A}$ is the set of PCF constants, which is type-respecting i.e. if c is a constant of type A, then $[\![c]\!]^{\mathcal{I}}$ is an element of D^A
- a (semantic) fixed point operator $\mathsf{fix}^{\mathcal{I}}_A : D^{A \Rightarrow A} \longrightarrow D^A$ for each type A.

5.2 The standard model

By the standard model, we mean the continuous function space model first proposed by Scott in 1969. We say that the ***standard interpretation*** is the standard collection of value domains

[4] To be exact, there is an equivalence between the category of simply-typed lambda calculi with surjective pairing, and the category of cartesian closed categories; see e.g. [13].

together with the following interpretation (we write $[\![-]\!]^{\mathcal{S}}$ where $\mathcal{S}$ is for "standard" or for "Scott"): for $n \in D^{\iota}$, $b \in D^{o}$

$$[\![\mathsf{t}]\!]^{\mathcal{S}} = \mathsf{t} \qquad [\![\mathsf{f}]\!]^{\mathcal{S}} = \mathsf{f}$$

$$[\![\mathsf{cond}]\!]^{\mathcal{S}} bde = \begin{cases} d & \text{if } b = \mathsf{t} \\ e & \text{if } b = \mathsf{f} \\ - & \text{if } b = - \end{cases} \qquad [\![\mathsf{zero}]\!]^{\mathcal{S}} n = \begin{cases} \mathsf{t} & \text{if } n = 0 \\ \mathsf{f} & \text{if } n > 0 \\ - & \text{if } n = - \end{cases}$$

$$[\![\mathsf{succ}]\!]^{\mathcal{S}} n = \begin{cases} n+1 & \text{if } n \geqslant 0 \\ - & \text{if } n = - \end{cases} \qquad [\![\mathsf{pred}]\!]^{\mathcal{S}} n = \begin{cases} n-1 & \text{if } n \geqslant 1 \\ 0 & \text{if } n = 0 \\ - & \text{if } n = - \end{cases}$$

$$[\![n]\!]^{\mathcal{S}} = n \qquad (n \geqslant 0)$$

and a fixed point operation $\mathsf{fix}^{\mathcal{S}}_A : D^{A \Rightarrow A} \longrightarrow D^A$ for each A defined by

$$\mathsf{fix}^{\mathcal{S}}_A(f) \quad = \quad \bigsqcup \{ f^n(-^A) : n \in \omega \}.$$

Fix an interpretation $\mathcal{I}$ of PCF. We say that an *environment* is a function from the set of variables Var to $\bigcup D^A$; environments are ranged over by ρ, ρ', etc. We say that ρ is a $?$***-environment*** just in case for every $x : A$ in $?$, $\rho(x) \in D^A$. We write the set of all $?$-environments as $Env(?)$. The update operation is defined as follows: for $d \in D^A$,

$$\rho[x \mapsto d](y) \quad \stackrel{\text{def}}{=} \quad \begin{cases} d & \text{if } y = x \\ \rho(x) & \text{otherwise.} \end{cases}$$

The undefined environment $-$ sends every variable x to $-$.

5.3 Denotational semantics induced by an interpretation

We define the denotational semantics induced by an interpretation $\mathcal{I}$ of PCF as follows. Take a typing judgement $? \vdash s : A$ and take a $?$-environment ρ, we define

$$[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho} \in D^A$$

by recursion on s as follows:

$$\begin{aligned}
[\![? \vdash x : A]\!]^{\mathcal{I}}_{\rho} &= \rho(x) \\
[\![? \vdash c : A]\!]^{\mathcal{I}}_{\rho} &= [\![c]\!]^{\mathcal{I}} \\
[\![? \vdash st : B]\!]^{\mathcal{I}}_{\rho} &= [\![? \vdash s : A \Rightarrow B]\!]^{\mathcal{I}}_{\rho} [\![? \vdash t : A]\!]^{\mathcal{I}}_{\rho} \\
[\![? \vdash \lambda x^A.s : A \Rightarrow B]\!]^{\mathcal{I}}_{\rho} &= \boldsymbol{\lambda} d \in D^A.[\![?, x : A \vdash s : B]\!]^{\mathcal{I}}_{\rho[x \mapsto d]} \\
[\![? \vdash \mathbf{Y}^A(s) : A]\!]^{\mathcal{I}}_{\rho} &= \mathsf{fix}^{\mathcal{I}}_A([\![? \vdash s : A \Rightarrow A]\!]^{\mathcal{I}}_{\rho})
\end{aligned}$$

where $\mathsf{fix}_A^{\mathcal{I}}$ is the fixed point operation specified by the interpretation $\mathcal{I}$. In the fourth clause above, the denotation is the set-theoretic function given by $d \mapsto [\![?, x : A \vdash s : B]\!]^{\mathcal{I}}_{\rho[x \mapsto d]}$, for $d \in D^A$.

We need to check that the definition is good: if $? \vdash s : A$ is provable then for any $?$-environment ρ, $[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho}$ is an element of the value domain D^A.

Proposition 5.1 *Let $\mathcal{I}$ be an interpretation of PCF. Then the following good structural properties of the induced denotational semantics are satisfied:*

(i) ***Weakening****. Suppose $? \vdash s : A$ and ρ and ρ' are $?$-environments. If $\rho(x) = \rho'(x)$ for every x that occurs free in s then*

$$[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho} = [\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho'}.$$

(ii) ***Substitution****. For all $(?, \Delta)$-environments ρ,*

$$[\![?, \Delta \vdash s[t/y] : A]\!]^{\mathcal{I}}_{\rho} = [\![?, y : B \vdash s : A]\!]^{\mathcal{I}}_{\rho[y \mapsto [\![\Delta \vdash t : B]\!]^{\mathcal{I}}_{\rho}]}$$

(iii) ***Compositionality****. If for all $?$-environments ρ, we have $[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho} = [\![? \vdash t : A]\!]^{\mathcal{I}}_{\rho}$, then for any context $C[X]$ such that $?, \Delta \vdash C[s] : B$ and $?, \Delta \vdash C[t] : B$ are provable, we have for all $(?, \Delta)$-environment ρ,*

$$[\![?, \Delta \vdash C[s] : B]\!]^{\mathcal{I}}_{\rho} = [\![?, \Delta \vdash C[t] : B]\!]^{\mathcal{I}}_{\rho}.$$

Notation 5.2 **(i)** Henceforth we will only consider the denotation $[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho}$ of a provable judgement $? \vdash s : A$ in a $?$-environment ρ.

(ii) To save writing, we shall often write $[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho}$ simply as $[\![s]\!]^{\mathcal{S}}_{\rho}$; whenever we do so, we shall assume that the free variables of s are implicitly typed according to a fixed $?$.

The standard interpretation induces a model of PCF in the following sense.

Proposition 5.3 *The denotational semantics induced by the standard interpretation is a model of the PCF type theory i.e.*

(i) if $? \vdash s : A$ is provable then $[\![s]\!]^{\mathcal{S}}_{\rho} \in D^A$ for any $?$-environment ρ

(i) if $? \vdash s = t : A$ is provable then $\mathcal{S} \vDash ? \vdash s = t : A$ (or simply $\mathcal{S} \vDash s = t : A$ when the typing context is known) i.e. $[\![s]\!]^{\mathcal{S}}_{\rho} = [\![t]\!]^{\mathcal{S}}_{\rho} \in D^A$ for any $?$-environment ρ. $\square$

Problems

5.1 Let D be a CPO. Prove that the fixpoint operator $\mathsf{fix} : [D \Rightarrow D] \longrightarrow D$ is a continuous function.

5.2 Prove that every interpretation of PCF induces a denotational semantics which is compositional, and satisfies the properties of substitution and weakening in the sense defined in this section.

5.3 Prove Proposition 5.1.

5.4 Prove Proposition 5.3.

6 Adequacy and Weak Adequacy

From an operational point of view, two program fragments are equal if they can always be interchanged without affecting the observable outcome of the computation – this is the definition of observational equivalence. According to denotational semantics, the meaning of a program or a term is what the semantic function denotes. Thus from a denotational point of view, two programs are equal if they have the same denotation in the model.

Definition 6.1 (i) A denotational semantics $[\![-]\!]^{\mathcal{I}}$ is said to be ***adequate*** for PCF if denotational equality implies observational equivalence i.e. for any terms s and t such that $? \vdash s : A$ and $? \vdash t : A$ are provable

$$\forall \rho \in Env(?).[\![s]\!]^{\mathcal{I}}_{\rho} = [\![t]\!]^{\mathcal{I}}_{\rho} \quad \Longrightarrow \quad s \approx^{\text{obs}} t.$$

We say that $? \vdash s = t : A$ is ***valid*** in $\mathcal{I}$, written $\mathcal{I} \vdash ? \vdash s = t : A$, just in case for every $?$-environment ρ, we have $[\![? \vdash s : A]\!]^{\mathcal{I}}_{\rho} = [\![? \vdash t : A]\!]^{\mathcal{I}}_{\rho}$.

(ii) If the converse is also true, then the denotational semantics is said to be (equationally) ***fully abstract*** for PCF.

We define the ***theory induced by*** $\mathcal{I}$ to be the collection of equality judgements that are valid in $\mathcal{I}$. Thus $\mathcal{I}$ is *equationally fully abstract* (for PCF) just in case the theory induced by it coincides with observational equivalence.

(iii) We say that the denotational semantics is ***weakly adequate*** if for any program s and for any ground value v

$$[\![s]\!]^{\mathcal{I}}_{-} = [\![v]\!]^{\mathcal{I}}_{-} \quad \Longleftrightarrow \quad s \Downarrow v.$$

6.1 Adequacy and weak adequacy

Proposition 6.2 *For any interpretation of PCF, weak adequacy implies adequacy.*

Proof Take terms s and t such that $[\![s]\!]^{\mathcal{I}}_{\rho} = [\![t]\!]^{\mathcal{I}}_{\rho}$ for any environment ρ. Given any program context $C[X]$ such that both $C[s]$ and $C[t]$ are programs, suppose $C[s] \Downarrow v$ for some value v. By weak adequacy, $[\![C[s]]\!]^{\mathcal{I}}_{-} = [\![v]\!]^{\mathcal{I}}_{-}$. By compositionality (Proposition 5.1(ii)), we have $[\![C[s]]\!]^{\mathcal{I}}_{-} = [\![C[t]]\!]^{\mathcal{I}}_{-}$. Hence, we deduce $[\![C[t]]\!]^{\mathcal{I}}_{-} = [\![v]\!]^{\mathcal{I}}_{-}$ for any program context $C[X]$. By weak adequacy again, we have $C[t] \Downarrow v$. In exactly the same way, we can show that $C[t] \Downarrow v$ implies $C[s] \Downarrow v$. □

Theorem 6.3 (Weak Adequacy) *The standard model is weakly adequate. For any program s and any ground value v, whenever $[\![s]\!]^{\mathcal{S}}_{-} = [\![v]\!]^{\mathcal{S}}_{-}$ then $s \Downarrow v$.* □

Notation 6.4 In the following we shall write the standard semantics $[\![-]\!]^{\mathcal{S}}_{\rho}$ simply as $[\![-]\!]_{\rho}$ whenever it is clear what we mean. For closed term s, or for $\rho = -$, we shall write $[\![s]\!]_{\rho}$ as $[\![s]\!]$.

The proof strategy is based on what is sometimes called *computability argument* in [23]. It illustrates a common feature of inductive reasoning in typed λ-calculus: it is often necessary to establish a property uniformly (for example, one that is applicable to all terms as opposed to just the closed terms; or to all types as opposed to just ground types) for terms across all types, when we are really only interested in its validity for a smaller collection of terms.

6.2 A computability argument

Let v range over ground values. We define a predicate called *computability* for *all* terms. At ground type, a closed term is said to be computable if it satisfies the the Weak Adequacy Theorem. The notion of computability is then extended to higher-type terms.

Definition 6.5 The ***computability*** predicate is defined as follows:

(i) If $s : \alpha$ is closed, then s is computable if $[\![s]\!] = [\![v]\!]$ implies $s \Downarrow v$.

(ii) If $s : A \Rightarrow B$ is closed, then s is computable just in case st is computable for every closed computable term $t : A$.

(iii) If $s : A$ has free variables[5] $y_1^{B_1}, \cdots, y_m^{B_m}$, then s is computable if for every substitution θ mapping each $y_i^{B_i}$ to a closed computable term t_i, the term

$$s_\theta \quad \equiv \quad s[t_1/y_1, \cdots, t_m/y_m]$$

is computable.

(We regard a substitution as a type-respecting map from variables to terms.) Combining the above, we observe that a term $s : (A_1, \cdots, A_n, \alpha)$ is computable if and only if

- for any substitution θ mapping free variables of s to closed computable terms, and
- for any closed computable terms $t_1 : A_1, \cdots, t_n : A_n$,

the term $s_\theta t_1 \cdots t_n : a$ is computable.

Proposition 6.6 *Every PCF term is computable.*

Proof We prove by induction on the structure of $s : A$ where $A = (A_1, \cdots, A_n, \alpha)$. A substitution θ is said to be *closed* and *computable* for s if θ maps the free variables of s to closed, computable terms.

CASE $s \equiv x^A$: Take any closed, computable substitution θ for x^A. For any sequence of closed computable terms $t_i : A_i$, by definition, $x_\theta \vec{t}$ is computable.

[5] We assume that $\Gamma \vdash s : A$ with $\Gamma(x_i) = A_i$ for free variables $x_1, \cdots, x_n$ of s.

CASE $s \equiv c$ where c is a constant: We shall just consider the case of cond^α for an illustration. For any closed, computable terms t_2, t_3 of type a and t_1 of type o, suppose $[\![\, \mathsf{cond}^\alpha\, t_1 t_2 t_3 \,]\!] = [\![\, v \,]\!]$. By definition of the standard interpretation, w.l.o.g. say $[\![\, t_1 \,]\!] = \mathsf{t}$, and so $[\![\, t_2 \,]\!] = [\![\, v \,]\!]$. Since t_1 and t_2 are closed, computable terms, we have $t_1 \Downarrow \mathsf{t}$ and $t_2 \Downarrow v$. Hence, $\mathsf{cond}^\alpha\, t_1 t_2 t_3 \Downarrow v$.

CASE $s \equiv \lambda x^{A_1}.p$: For any closed, computable substitution θ for s, and for any sequence of closed computable terms $t_i : A_i$, suppose $[\![\, (\lambda x^{A_1}.p)_\theta \vec{t} \,]\!] = [\![\, v \,]\!]$. Note that the following are valid:

(A) $[\![\, (\lambda x^{A_1}.p)_\theta t_1 \,]\!] = [\![\, (p_\theta[t_1/x^{A_1}]) \,]\!] = [\![\, (p[t_1/x^{A_1}])_\theta \,]\!]$

(B) $(p[t_1/x^{A_1}])_\theta \equiv p_{\theta'}$ where $\theta' \stackrel{\text{def}}{=} \theta[x^{A_1} \mapsto t_1]$.

Putting (A) and (B) together, we observe that $[\![\, p_{\theta'} t_2 \cdots t_n \,]\!] = [\![\, (\lambda x^{A_1}.p)_\theta t_1 \cdots t_n \,]\!] = [\![\, v \,]\!]$. By the induction hypothesis, p is computable; and so, since the substitution θ' is closed and computable for p, we have $p_{\theta'} t_2 \cdots t_n \Downarrow v$. By an appeal to (B), we have $(\lambda x^{A_1}.p)\vec{t} \Downarrow v$.

CASE $s \equiv \mathbf{Y}^A(p)$: This case requires us to introduce *syntactic approximants*

$$\{\, \mathbf{Y}^A_l(s) : l \geqslant 0 \,\}$$

of the fixed point operator. We extend the definition of typing judgements of PCF by the following rule: for each $l \geqslant 0$,

$$\frac{? \vdash s : A \Rightarrow A}{? \vdash \mathbf{Y}^A_l(s) : A.}$$

We extend the rules defining the one-step reduction $>$ by the axioms

$$\begin{array}{rcl} \mathbf{Y}^A_0(s) & > & \mathbf{Y}^A_0(s) \\ \mathbf{Y}^A_{l+1}(s) & > & s\mathbf{Y}^A_l(s) \end{array}$$

It is easy to see what the denotation of the approximant should be. We extend the inductive definition of the denotational semantics by the following rule:

$$[\![\, ? \vdash \mathbf{Y}^A_l(s) : A \,]\!]^{\mathcal{S}}_\rho \quad = \quad f^l(-)$$

where f is $[\![\, ? \vdash s : A \Rightarrow A \,]\!]^{\mathcal{S}}_\rho$. We note that $[\![\, \mathbf{Y}^A_{l+1}(s) \,]\!] = [\![\, s\mathbf{Y}^A_l(s) \,]\!]$. We need the following claim whose proof we leave as an exercise to the reader:

Claim

(i) For any $s : A$ we have $[\![\, \mathbf{Y}^A(s) \,]\!] = \bigsqcup_{l \geqslant 0} [\![\, \mathbf{Y}^A_l(s) \,]\!]$.

(ii) If $\mathbf{Y}^A_l(p)\vec{t} \Downarrow v : \alpha$ for some v then $\mathbf{Y}^A(p)\vec{t} \Downarrow v$.

For any closed, computable substitution θ for p, and for any sequence of closed, computable terms $t_i : A_i$, suppose $[\![\, \mathbf{Y}^A(p_\theta)\vec{t} \,]\!] = [\![\, v \,]\!]$. By Claim(i) and by compositionality, we have

$$\begin{array}{rcl} [\![\, v \,]\!] & = & (\bigsqcup_{l \geqslant 0} [\![\, \mathbf{Y}^A_l(p_\theta) \,]\!])[\![\, t_1 \,]\!] \cdots [\![\, t_n \,]\!] \\ & = & \bigsqcup_{l \geqslant 0} [\![\, (\mathbf{Y}^A_l(p_\theta)) t_1 \cdots t_n \,]\!]. \end{array}$$

The last step is justified because application is a continuous operation. Now, since D^a is a flat CPO, $[\![(\mathbf{Y}_l^A(p_\theta))t_1 \cdots t_n]\!] = [\![v]\!]$ for some l. Since $\mathbf{Y}_m^A(p)$ is computable, we have $(\mathbf{Y}_l^A(p_\theta))t_1 \cdots t_n \Downarrow v$. By Claim(ii), we have the desired result $(\mathbf{Y}^A(p_\theta))t_1 \cdots t_n \Downarrow v$.

CASE $s \equiv \mathbf{Y}_l^A(p)$, for each $l \geqslant 0$: We prove by induction on l. The base case is obvious. For any closed, computable substitution θ for $\mathbf{Y}_{l+1}^A(p)$, and any sequence of closed, computable terms $t_i : A_i$, suppose $[\![\mathbf{Y}_{l+1}^A(p_\theta)\vec{t}]\!] = [\![v]\!]$. Since $\mathbf{Y}_{l+1}^A(p_\theta) > p_\theta(\mathbf{Y}_l^A(p_\theta))$, and that the semantic function is sound, we have

$$[\![p_\theta(\mathbf{Y}_l^A(p_\theta))\vec{t}]\!] \quad = \quad [\![v]\!].$$

By the induction hypothesis, p is computable, and so, $p_\theta(\mathbf{Y}_l^A(p_\theta))\vec{t} \Downarrow v$. Hence, $\mathbf{Y}_{l+1}^A(p_\theta)\vec{t} \Downarrow v$.

CASE s is an application: Consider the possible shapes of s:

- $xp\vec{q}$,
- $cp\vec{q}$ for some constant c,
- $(\lambda x.p)p\vec{q}$,
- $\mathbf{Y}_n(p)p\vec{q}$,
- $\mathbf{Y}(r)p\vec{q}$.

Consider the last case for an illustration. For any closed, computable substitution θ for $\mathbf{Y}(r)p\vec{q}$, and for any sequence of closed, computable terms $t_i : A_i$, suppose $[\![(\mathbf{Y}(r)p\vec{q})_\theta\vec{t}]\!] = [\![v]\!]$. By the induction hypothesis, p_θ and $\vec{q_\theta}$ are closed and computable, and so, since $\mathbf{Y}(r)$ is computable as established in the previous case, $(\mathbf{Y}(r)p\vec{q})_\theta\vec{t} \Downarrow v$. The rest are entirely similar. □

The computability argument central to the proof can be traced back to an important paper of Tait [29]. A variation of the theme called *logical relation* was independently developed by Plotkin [20]; see also [28]. Logical relations are a very useful tool for proving properties of typed lambda calculi.

Plotkin's adequacy result for PCF has been extended to richer computational settings. For example, in [25], it is shown that adequacy holds for a meta-language for denotational semantics which has an elaborate type system including recursive types. An interesting line of research is to prove adequacy for as general a computational setting as possible. A related problem is the investigation of proof techniques for adequacy. For example, Pitts in [18] applies a general and powerful machinery mixing inductive and co-inductive reasoning to investigate adequacy proofs with respect to domains defined by recursive domain equations.

Denotational semantics prescribes meanings to syntactic entities *compositionally* i.e. the meaning of a composite term is defined in terms of the meanings of the respective components. If the mathematical properties of the denotational model are well understood, then the denotational approach lends itself to an "algebraic" mode of reasoning about global properties of programs (such as termination). How useful the denotational semantics is for predicting the behaviour of programs depends on how well it fits with the operational behaviour of programs. Adequacy establishes just such a "goodness-of-fit" criterion which is of fundamental importance.

6.3 Yet another proof of the Context Lemma: logical relations

Henceforth we shall only consider the standard model, so that D^A refers to the value domain of the standard collection. Let PCF^o_A denote the collection of closed terms of type A.

Definition 6.7 For each type A, we define a relation $\lhd^A \subseteq D^A \times \mathrm{PCF}^o_A$ as follows:

$$\begin{array}{rcl} d \lhd^\alpha s & \iff & d \sqsubseteq [\![s]\!] \\ d \lhd^{A \Rightarrow B} s & \iff & \forall e \in D^A . \forall t \in \mathrm{PCF}^o_A . e \lhd^A t \implies de \lhd^B st. \end{array}$$

Observe that the following immediate consequences of the definition.

(i) For $A = (A_1, \cdots, A_n, \alpha)$, $d \lhd^A s$ if and only if

$$\forall e_i : D^{A_i} . \forall t_i \in \mathrm{PCF}^o_{A_i} . \bigwedge_{1 \leqslant i \leqslant n} e_i \lhd^{A_i} t_i \implies d\vec{e} \lhd^\alpha s\vec{t}.$$

(ii) If $d \lhd^{A \Rightarrow B} s$ and $e \lhd^A t$ then $de \lhd^B st$.

To saving writing, we shall often omit the type label and write $\lhd^A$ simply as $\lhd$. First we establish a basic property of logical relation.

Lemma 6.8 (Transitivity) *For any type $A = (A_1, \cdots, A_n, \alpha)$, if $d^- \sqsubseteq d \lhd^A s \sqsubseteq^{\mathrm{app}}_{\sim} s^+$ then $d^- \lhd^A s^+$.*

Proof Take e_i and t_i such that for each i, $e_i \lhd^{A_i} t_i$. By assumption $d^- \vec{e} \sqsubseteq d\vec{e}$. Since $d\vec{e} \sqsubseteq [\![s\vec{t}]\!]$ by assumption, we have $d^- \vec{e} \sqsubseteq [\![s\vec{t}]\!]$. W.l.o.g. suppose $d^- \vec{e} = n$, say. By Weak Adequacy, $s\vec{t} \Downarrow n$, and so, as we have $s \sqsubseteq^{\mathrm{app}}_{\sim} s^+$ by assumption, $s^+ \vec{t} \Downarrow n$. Thus by Weak Adequcy, $[\![s^+ \vec{t}]\!] = n$ as required. □

We need another result about the logical relations, which is sometimes called the Fundamental Theorem of Logical Relations.

Theorem 6.9 (Reflexivity) *For any closed term s, $[\![s]\!] \lhd s$.* □

For types A, B and C, we name the following closed terms as follows:

$$\begin{array}{rcl} S^{A,B,C} & \equiv & \lambda xyz.xz(yz) : (A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C) \\ K^{A,B} & \equiv & \lambda xy.x : A \Rightarrow B \Rightarrow A \end{array}$$

Because of the restriction to closed terms, we shall consider terms generated from constants, $S^{A,B,C}$ and $K^{A,B}$ by application and the fixed point operators. We call these terms *PCF combinators*. Note the following result

Proposition 6.10 *For each closed term t, there is a PCF combinator $\overline{t}$ of the same type such that $\vdash t = \overline{t} : A$.* □

We need to check the following cases:

(i) for every constant c, $[\![c]\!] \lhd c$

(ii) for every A, B and C, $[\![S^{A,B,C}]\!] \lhd S^{A,B,C}$ and $[\![K^{A,B}]\!] \lhd K^{A,B}$

(iii) for each A, $[\![\lambda g^{A \Rightarrow A}.\mathbf{Y}^A(g)]\!] \lhd^{(A \Rightarrow A) \Rightarrow A} \lambda g^{A \Rightarrow A}.\mathbf{Y}^A(g)$.

We leave the verification of the first two cases as an exercise for the reader. For (iii), set $A = (A_1, \cdots, A_n, \alpha)$ and take terms s, t_i and domain elements f, e_i such that $f \lhd^{A \Rightarrow A} s$ and $e_i \lhd^{A_i} t_i$ for each i. We aim to prove

$$[\![\lambda g^{A \Rightarrow A}.\mathbf{Y}^A(g)]\!] f\vec{e} \;=\; (\bigsqcup_{n \geqslant 0} f^n -)\vec{e} \;\sqsubseteq\; [\![s\mathbf{Y}^A(s)\vec{t}]\!] \;=\; [\![\lambda g^{A \Rightarrow A}.\mathbf{Y}^A(g) s\vec{t}]\!].$$

W.l.o.g. suppose the LHS is some non-$-$ element a of the appropriate ground type value domain. Then for some $m \geqslant 0$, $f(f^m -)\vec{e} = a$. Inductively we have $f^m - \lhd^A s^m \Omega^A$, and so, $f(f^m -)\vec{e} \lhd^\alpha s(s^m \Omega^A)\vec{t}$. That is to say,

$$(\bigsqcup_{n \geqslant 0} f^n -)\vec{e} \;=\; f(f^m -)\vec{e} \;\sqsubseteq\; [\![s(s^m \Omega^A)\vec{t}]\!] \;=\; [\![s\mathbf{Y}^A(s)\vec{t}]\!]$$

and we are done.

Finally to prove the Reflexivity Lemma, take any closed term t. By Proposition 6.10, there is a combinator $\overline{t}$ such that $\vdash t = \overline{t} : A$. Since $\approx^{\text{app}}$ is a PCF theory in the sense of Theorem 4.4, we have $t \approx^{\text{app}} \overline{t}$; and since the standard interpretation induces a model of the PCF Type Theory in the sense of Proposition 5.3, we have $[\![t]\!] = [\![\overline{t}]\!]$. We have just shown that $[\![\overline{t}]\!] \lhd \overline{t}$. The result then follows from the Transitivity Lemma 6.8.

Corollary 6.11 *If $[\![s]\!] \lhd t$ then for any closed context C such that both $C[s]$ and $C[t]$ are well-typed, then $[\![C[s]]\!] \lhd C[t]$.*

Proof The proof is by induction on the structure of context C such that $C[s]$ and $C[t]$ are programs. Note that the base cases of $C[X] \equiv c$ and X are just the Reflexivity Lemma. The inductive cases are completely straightforward. □

We can now prove the Context Lemma. Take closed s and t of a given type A such that $s \sqsubseteq^{\text{app}}_{\approx} t$. Take closed context C and suppose $C[s] \Downarrow n$. By Lemma 6.8 and Theorem 6.9, we have $[\![s]\!] \lhd^A t$. By Corollary 6.11, $[\![C[s]]\!] \lhd^B C[t]$. By Weak Adequacy, we have $[\![C[s]]\!] = n$, and so, $[\![C[t]]\!] = n$ by definition. Hence by Weak Adequacy, we have $C[t] \Downarrow n$.

Problems

6.1 Prove Proposition 6.10.

6.2 Prove the following:

(i) for every constant c, $[\![c]\!] \lhd c$

(ii) for every A, B and C, $[\![S^{A,B,C}]\!] \lhd S^{A,B,C}$ and $[\![K^{A,B}]\!] \lhd K^{A,B}$.

6.3 (i) How would you extend the logical relation $\lhd^A$ (as introduced in Definition 6.7) to terms which are not necessarily closed?

[Hint: Consider (closed) substitution σ which maps variables to type-compatible closed terms.]

(ii) Use the new definition to rework the proof of the Context Lemma. You should not (need to) use the $S^{A,B,C}$ and $K^{A,B}$ terms in your argument.

6.4 Does the logical relation argument work for the Call-by-Value PCF?

6.5 Define a new type-indexed family of logical relations $\lhd^A \subseteq D^A \times \mathrm{PCF}^o_A$ as follows:

$$\begin{aligned} d \lhd^\alpha s &\iff d \in \mathbb{N} \cup \mathbb{B} \implies s \Downarrow d \\ d \lhd^{A \Rightarrow B} s &\iff \forall e \in D^A . \forall t \in \mathrm{PCF}^o_A . e \lhd^A t \implies de \lhd^B st. \end{aligned}$$

Use the logical relations to prove the Weak Adequacy Theorem 6.3.

7 A Primer on SFP Domains

A *complete partial order* (or CPO, for short) is a partially ordered set $\langle D, \sqsubseteq \rangle$ such that every directed subset of D has a least upper bound (lub) in D. Recall that a subset X of a poset $\langle D, \sqsubseteq \rangle$ is said to be *directed* just in case every pair of elements of X has an upper bound in X. Note that a CPO D always has a least element which is denoted $-^D$ (or simply $-$) since it is the lub of the directed subset $\varnothing$ of D. An element d of a CPO D is *compact* (or finite, or isolated) if for any directed subset X of D, whenever $d \sqsubseteq \bigsqcup X$, then there is some $x \in X$ such that $d \sqsubseteq x$.

7.1 Scott domains

A CPO D is ***algebraic*** if for every element $d \in D$, the subset $\{ x \sqsubseteq d : x \text{ is compact} \}$ of D is directed, and its lub is d. Further, D is ***ω-algebraic*** if D has denumerably many compact elements. A subset X of the CPO D is said to be ***consistent*** (or ***compatible***) if it has an upper bound in D, and we write $\uparrow X$, or $x \uparrow y$ if X is the set $\{ x, y \}$.

Proposition 7.1 *For any CPO D, the following properties are equivalent:*

(1) consistent completeness*: every consistent subset of D has a lub in D.*

(2) every compatible pair of elements of D has a lub in D,

(3) every nonemtpy subset has a greatest lower bound (glb) in D. □

A CPO is said to be ***consistently complete*** if any one of the above conditions is satisfied. We shall call any consistently complete, ω-algebraic CPO a ***Scott domain***. A key property of the category of Scott domains is that it is closed under function space construction. In fact, it is a cartesian closed category. A good reference is [24], see also [6].

An ***inverse system of projections*** for a CPO D is a countable collection $\langle \psi_n \rangle_{n \in \omega}$ of continuous functions from D to D satisfying the following conditions: for each $n \in \omega$,

(i) *projection*: $\psi_n \circ \psi_n = \psi_n$

(ii) *finiteness*: the image $\psi_n(D)$ is finite (and hence a finite CPO)

(iii) *approximation*: $\psi_n \sqsubseteq \psi_{n+1}$

(iv) *ω-limit*: $\mathsf{id}_D = \bigsqcup_{n \in \omega} \psi_n$.

Proposition 7.2 *If a CPO D has an inverse system of projections then it is ω-algebraic. Further $\{ \psi_n d : n \geqslant 0, d \in D \}$ is the set of compact elements of D.* □

7.2 Characterizations of SFP domains

Proposition 7.3 *Given any CPO D, the following conditions are equivalent:*

(1) D has an inverse system of projections.

(2) D is the colimit of a directed diagram of finite posets over the category of CPOs and embedding-projection pairs.

(3) D is ω-algebraic, and satisfies

- ***Condition (M)**: for any finite subset X of compact elements of D, the set $\mathrm{MUB}(X)$ of minimal upper bounds of X is finite, and moreover* complete *in the sense that for any upper bound u of X, there is some $x \in \mathrm{MUB}(X)$ such that $x \sqsubseteq u$.*
- *Further, the set $\mathcal{U}^*(X)$ is finite, where*

$$\begin{aligned} \mathcal{U}^0(X) &\stackrel{\text{def}}{=} X, \\ \mathcal{U}^{n+1}(X) &\stackrel{\text{def}}{=} \bigcup\{\, \mathrm{MUB}(U) : U \subseteq \mathcal{U}^n(X) \,\}, \\ \mathcal{U}^*(X) &\stackrel{\text{def}}{=} \textstyle\bigcup_{k\in\omega} \mathcal{U}^k(X). \end{aligned}$$

□

*A CPO D is said to be **strongly algebraic** if anyone of the above conditions are satisfied.*

A strongly algebraic CPO is also known as an ***SFP domain*** [22]. The name is an acronym for "Sequence of Finite Posets" which arises from one of the main descriptions of an SFP domain as the bilimit (i.e. a direct limit of embeddings, or equivalently, by the "limit-colimit coincidence" due to Scott, an inverse limit of projections) of a sequence of finite posets. For this reason, an SFP domain is also called a ***bifinite CPO***. The following references are recommended: [24], [10] or [9, Ch. 10].

Lemma 7.4 *An SFP domain in which any two elements have a greatest lower bound is a Scott domain.* □

Problems

7.1 Prove Lemma 7.4: Ann SFP domain in which any two elements have a greatest lower bound is a Scott domain.

8 Full Abstraction for PCFP

In [23] Plotkin proved that the standard model is not fully abstract for PCF, but it is for PCF extended by a parallel conditional construct. Here we shall prove full abstraction of the standard model for a different extension of PCF, PCF extended by parallel or. The proof is due to Curien [5].

8.1 The standard model is not fully abstract for PCF

Theorem 8.1 *The standard model is not fully abstract for PCF.*

Proof Set

$$C[X] \quad \equiv \quad \lambda p^{o \Rightarrow o \Rightarrow o}.\mathsf{cond}^o\ (p\,\mathsf{t}\,\Omega)\,(\mathsf{cond}^o\ (p\,\Omega\,\mathsf{t})(\mathsf{cond}^o\ (p\,\mathsf{f}\,\mathsf{f})\,\Omega\,X)\,\Omega)\,\Omega.$$

Define T and F to be $C[\mathsf{t}]$ and $C[\mathsf{f}]$ respectively. By considering the small-step reduction $>$, we have: for any closed term p of type $o \Rightarrow o \Rightarrow o$

- $Tp \downarrow \mathsf{t}$ if and only if p satisfies the operational properties of parallel or in Proposition 3.3.
- $Fp \downarrow \mathsf{f}$ if and only if p satisfies the same properties.

In view of the Proposition 3.3, we have $T \sqsubseteq^{\mathrm{app}}_{\sim} F$ and $F \sqsubseteq^{\mathrm{app}}_{\sim} T$ and so, by the Context Lemma 4.5, we have $T \approx^{\mathrm{obs}} F$. Let *por* be the function $D^o \longrightarrow D^o \Rightarrow D^o$ (recall that D^o is $\mathbb{B}_\perp$ the flat CPO of booleans) defined by

$$por\, b_1\, b_2 \quad = \quad \begin{cases} \mathsf{t} & \text{if } b_1 = \mathsf{t} \text{ or } b_2 = \mathsf{t}, \\ \mathsf{f} & \text{if } b_1 = b_2 = \mathsf{f}, \\ \perp & \text{otherwise.} \end{cases}$$

Since *por* is monotone, it is in $D^{o \Rightarrow o \Rightarrow o}$ of the standard model. As $[\![\, T \,]\!]^{\mathcal{S}}\, por = \mathsf{t}$ and $[\![\, F \,]\!]^{\mathcal{S}}\, por = \mathsf{f}$, we have $[\![\, T \,]\!]^{\mathcal{S}} \neq [\![\, F \,]\!]^{\mathcal{S}}$. □

The preceding proposition shows that the standard model is too rich a structure for PCF. The observation was first made by Scott in 1969 [26]. There are two ways by which we can try to achieve full abstraction. The first fixes the model and extends the language, the second cuts the model down to size to fit the language. We shall call the first approach *expansive* and the second *restrictive*.

8.2 PCFP: PCF plus Parallel Or

PCFP is PCF augmented by parallel or. Curien has shown that such an addition gives enough expressive power to PCF so that all compact elements of the standard model are definable. Full abstraction is then a straightforward consequence.

PCFP is obtained from PCF by augmenting it by a constant $\mathsf{p\text{-}or}$ of type $o \Rightarrow o \Rightarrow o$. The one-step reduction $>$ is extended by the following axioms: for PCFP term b of type o

$$\begin{array}{lcl} \mathsf{p\text{-}or}\,\mathsf{t}\,b & > & \mathsf{t} \\ \mathsf{p\text{-}or}\,b\,\mathsf{t} & > & \mathsf{t} \\ \mathsf{p\text{-}or}\,\mathsf{f}\,\mathsf{f} & > & \mathsf{f}. \end{array}$$

and the definition of evaluation contexts has two additional rules:

$$E \quad ::= \quad \mathsf{p\text{-}or}\,E\,s \quad | \quad \mathsf{p\text{-}or}\,s\,E.$$

Note that the new one-step reduction is no longer deterministic e.g.

$$\begin{array}{lcl} \mathsf{p\text{-}or}\,(\mathsf{zero}\ 1)((\lambda x^o.x)\mathsf{f})) & > & \mathsf{p\text{-}or}\,\mathsf{f}((\lambda x^o.x)\mathsf{f})) \\ \mathsf{p\text{-}or}\,(\mathsf{zero}\ 1)((\lambda x^o.x)\mathsf{f})) & > & \mathsf{p\text{-}or}\,(\mathsf{zero}\ 1)\mathsf{f} \end{array}$$

Nevertheless $>$ is Church-Rosser i.e. if $s \gg t_1$ and $s \gg t_2$ then $t_1 \gg u$ and $t_2 \gg u$ for some u. Thus the evaluation of PCFP is still well-defined.

Exercise 8.1 Define an evaluation relation $\Downarrow$ for PCFP which agrees with $\downarrow$.

For each $n \geqslant 2$, we define an n-ary *parallel or* recursively as

$$\mathsf{p\text{-}or}_{n+1} \quad \equiv \quad \lambda x_1^o \cdots x_{n+1}^o.\mathsf{p\text{-}or}(\mathsf{p\text{-}or}_n x_1 \cdots x_n)x_{n+1}.$$

We define a *negation* term neg of type $o \Rightarrow o$, *parallel and* $\mathsf{p\text{-}and}$ and *left and* $\mathsf{l\text{-}and}$ both of type $o \Rightarrow o \Rightarrow o$, and the *n-ary parallel and* as follows:

$$\begin{array}{rcl} \mathsf{neg} & \equiv & \lambda x^o.\mathsf{cond}^o\,x\mathsf{ft} \\ \mathsf{p\text{-}and} & \equiv & \lambda x^o.\lambda y^o.\mathsf{neg}(\mathsf{p\text{-}or}(\mathsf{neg}x)(\mathsf{neg}y)) \\ \mathsf{p\text{-}and}_{n+1} & \equiv & \lambda x_1^o \cdots x_{n+1}^o.\mathsf{p\text{-}and}\,(\mathsf{p\text{-}and}_n x_1 \cdots x_n)x_{n+1} \\ \mathsf{l\text{-}and} & \equiv & \lambda x^o.\lambda y^o.\mathsf{cond}^o\,x(\mathsf{cond}^o\,y\mathsf{tf})\mathsf{f}. \end{array}$$

Similarly we extend $\mathsf{l\text{-}and}$ to an n-ary version $\mathsf{l\text{-}and}_n$.

8.3 Definability of compact elements

We recall a useful fact in Domain Theory.

Lemma 8.2 (Scott) *(i) If D and E are Scott domains, so is $[D \Rightarrow E]$, the CPO of continuous functions form D to E under the induced pointwise ordering.*

(ii) The compact elements of $[D \Rightarrow E]$ *are least upper bounds of finite sets of* ***step functions*** $(d \searrow e)$ *with* d *and* e *ranging over the compact elements of* D *and* E *respectively where for* $x \in D$

$$(d \searrow e)(x) \quad = \quad \begin{cases} e & \text{if } d \sqsubseteq x \\ - & \text{otherwise} \end{cases}$$

□

Since the ground type value domains $D^\iota = \mathbb{N}_\perp$ and $D^o = \mathbb{B}_\perp$ of the standard model are both Scott domains, by Lemma 8.2, every value domain D^A is a Scott domain.

For each Scott domain D, we write $\mathsf{K}(D)$ for the collection of compact elements of D. Fix a type $A = (A_1, \cdots, A_n, \alpha)$. By Lemma 8.2, any compact element of D^A is just $\bigsqcup F$ where F ranges over finite subsets of D^A satisfying:

(1) each element of F has the shape $(a_1 \searrow (a_2 \searrow \cdots (a_n \searrow v) \cdots))$ such that $a_i \in \mathsf{K}(D^{A_i})$ for each $1 \leqslant i \leqslant n$, and $v \in \mathsf{K}(D^a)$ with $v \neq -$,

(2) for any two elements $(a_1 \searrow (a_2 \searrow \cdots (a_n \searrow v) \cdots))$ and $(a'_1 \searrow (a'_2 \searrow \cdots (a'_n \searrow v') \cdots))$ of F, if a_i and a'_i are compatible for every $1 \leqslant i \leqslant n$, then $v = v'$.

In fact, a set satisfying (1) above also satisfies (2) if and only if F is consistent as a subset of D^A. Following Gunter, we call a set satisfying (1) and (2) ***crisp***.

We can interpret PCFP in the standard model by extending the standard interpretation by setting $[\![\, \mathsf{p\text{-}or} \,]\!]^{\mathcal{S}} = por \in D^{o \Rightarrow o \Rightarrow o}$ (see the proof of Theorem 8.1).

Lemma 8.3 (Definability) *For each type* A*, all compact elements of the domain* D^A *in the standard model are definable in PCFP.*

Proof It suffices to prove: for every type A

(1) for each $a \in \mathsf{K}(D^A)$, there is a PCFP term $M[a]$ defining a

(2) for each $a \in \mathsf{K}(D^A)$, there is a PCFP term $T[a]$ defining $(a \searrow \mathsf{t})$

(3) for each $a, b \in \mathsf{K}(D^A)$, if a and b are incompatible there is a PCFP term $D[a, b]$ defining $(a \searrow \mathsf{t}) \sqcup (b \searrow \mathsf{f})$.

We prove the above simultaneously by structural induction on A.

(1): The base cases corresponding to ground types o and ι are straightforward. For the inductive case of $A = (A_1, \cdots, A_n, \alpha)$, take any compact element $a \in \mathsf{K}(D^A)$. We argue by induction on the least number m such that there is a crisp set $F = \{\, f^j : 1 \leqslant j \leqslant m \,\}$ satisfying $a = \bigsqcup F$; call m the breadth of a. Let $\mathbf{H}(m)$ be the statement that every compact element of D^A of breadth m is definable. If m is zero then $F = \varnothing$, and so, a is just $-$. For $m \geqslant 1$ we write

$$f^j \quad = \quad (a^j_1 \searrow (a^j_2 \searrow \cdots (a^j_n \searrow v^j) \cdots)).$$

There are two subcases (modulo permutations of $f^1, \cdots, f^m$):

(1a) $v^1 = v^2 = \cdots = v^m$

(1b) there is some k where $1 \leqslant k < m$ such that $v^1 = \cdots = v^k$, and $v^1 \neq v_j$ for each $k+1 \leqslant j \leqslant m$.

In the case of (1a), by induction hypotheses (1) and (2) respectively, there are PCFP terms v and $T_i^j \equiv T[a_i^j]$, defining v_1 and $(a_i^j \searrow \mathsf{t})$ respectively, for each $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. It is easy to see that

$$M[a] \equiv \lambda x_1^{A_1} \cdots x_n^{A_n}.\mathsf{cond}^\alpha\, [\mathsf{p\text{-}or}_m(\mathsf{l\text{-}and}_n(T_1^1 x_1) \cdots (T_n^1 x_n)) \cdots (\mathsf{l\text{-}and}_n(T_1^m x_1) \cdots (T_n^m x_n))] v \Omega$$

defines a.

In the case of (1b), since F is crisp, for each pair (u, v) where $1 \leqslant u \leqslant k$ and $k+1 \leqslant v \leqslant m$, there is an i, depending on (u, v), such that a_i^u and a_i^v are incompatible. By the induction hypothesis (3), there is a PCFP term $D^{u,v} \equiv D[a_i^u, a_i^v]$ defining $(a_i^u \searrow \mathsf{t}) \sqcup (a_i^v \searrow \mathsf{f})$. By induction hypotheses $\mathbf{H}(k)$ and $\mathbf{H}(m-k)$ respectively, there are PCFP terms s_1 and s_2 defining the compact elements $f^1 \sqcup \cdots \sqcup f^k$ and $f^{k+1} \sqcup \cdots \sqcup f^m$ respectively. It remains to observe that if there is a PCFP term D such that

for any i, for any $d_i \in D^{A_i}$, if $ad_1 \cdots d_n \neq -$ then

(i) if $ad_1 \cdots d_n = v_1$ then $[\![D]\!]_- d_1 \cdots d_n = \mathsf{t}$

(ii) if $ad_1 \cdots d_n \neq v_1$ then $[\![D]\!]_- d_1 \cdots d_n = \mathsf{f}$

then (it is straightforward to see that) the term

$$M[a] \equiv \lambda x_1^{A_1} \cdots x_n^{A_n}.\mathsf{cond}^\alpha\, (Dx_1 \cdots x_n)(s_1 x_1 \cdots x_n)(s_2 x_1 \cdots x_n)$$

defines a.

Consider the term

$$\begin{aligned} D \;\equiv\; & \lambda x_1^{A_1} \cdots x_n^{A_n}.\mathsf{p\text{-}or}_k(\mathsf{p\text{-}and}_{m-k}(D^{1,k+1} x_1 \cdots x_n) \cdots (D^{1,m} x_1 \cdots x_n)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ & \qquad\qquad (\mathsf{p\text{-}and}_{m-k}(D^{k,k+1} x_1 \cdots x_n) \cdots (D^{k,m} x_1 \cdots x_n)). \end{aligned}$$

For $p \geqslant 2$, recall that the PCFP term $\mathsf{p\text{-}and}_p$ has type $\underbrace{o \Rightarrow o \Rightarrow \cdots \Rightarrow o}_{p+1}$ and has denotation satisfying: for $b_i \in \mathbb{B}_-$,

$$[\![\mathsf{p\text{-}and}_p]\!]_- b_1 \cdots b_p \quad = \quad \begin{cases} \mathsf{t} & \text{if } b_i = \mathsf{t} \text{ for every } i \leqslant p \\ \mathsf{f} & \text{if } b_i = \mathsf{f} \text{ for some } i \leqslant p \\ - & \text{otherwise;} \end{cases}$$

and similarly for $\mathsf{p\text{-}or}_k$. To verify that D satisfies the necessary properties, for any $d_i \in D^{A_i}$, suppose $ad_1 \cdots d_n = v_1$. Then there must be some u_0 with $1 \leqslant u_0 \leqslant k$ such that $a_i^{u_0} \sqsubseteq d_i$ for every i. Thus

$$[\![\mathsf{p\text{-}and}_{m-k}(D^{u_0,k+1} d_1 \cdots d_n) \cdots (D^{u_0,m} d_1 \cdots d_n)]\!]_- = \mathsf{t};$$

and so, $[\![D]\!]_{_} d_1 \cdots d_n = \mathsf{t}$. Now suppose $ad_1 \cdots d_n \neq v_1$ and $\neq -$. Since D^α is a flat CPO, there must be some v_0 where $k+1 \leqslant v_0 \leqslant m$ such that $a_i^{v_0} \sqsubseteq d_i$ for every i. Hence for every u such that $1 \leqslant u \leqslant k$, $[\![D^{u,v_0}]\!]_{_} d_1 \cdots d_n = \mathsf{f}$; and so, $[\![D]\!]_{_} d_1 \cdots d_n = \mathsf{f}$.

(2) For simplicity, we may write $a = (a^1 \searrow \overline{a}^1) \sqcup \cdots \sqcup (a^p \searrow \overline{a}^p)$. By induction hypotheses (1) and (2) respectively, there are PCFP terms $M[a^j]$ and $T[\overline{a}^j]$ defining compact elements a^j and $(\overline{a}^j \searrow \mathsf{t})$ respectively. The following term defines $(a \searrow \mathsf{t})$:

$$\lambda x^A.\mathsf{l\text{-}and}_m(T[\overline{a}^1](xM[a^1])) \cdots (T[\overline{a}^p](xM[a^p])).$$

(3) We write a as in the previous case, and write $b = (b^1 \searrow \overline{b}^1) \sqcup \cdots \sqcup (b^q \searrow \overline{b}^q)$. Since a and b are incompatible, there are u and v such that $(a^u \searrow \overline{a}^u)$ and $(b^v \searrow \overline{b}^v)$ are incompatible; that is to say, a^u and b^v are compatible, and $\overline{a}^u$ and $\overline{b}^v$ are not. Applying the induction hypotheis (3), $D[\overline{a}^u, \overline{b}^v]$ is a term that defines $(\overline{a}^u \searrow \mathsf{t}) \sqcup (\overline{b}^v \searrow \mathsf{f})$. Similarly $M[a^u \sqcup b^v]$, $T[a]$ and $T[b]$ define $a^u \sqcup b^v$, $(a \searrow \mathsf{t})$ and $(b \searrow \mathsf{t})$ respectively. Hence

$$D[a,b] \quad \equiv \quad \lambda x^A.\mathsf{cond}^\alpha\,(D[\overline{a}^u, \overline{b}^v](xM[a^u \sqcup b^v]))(T[a]x)(\mathsf{neg}(T[b]x)).$$

defines $(a \searrow \mathsf{t}) \sqcup (b \searrow \mathsf{f})$. This concludes our proof of the definability result. □

8.4 Inequational full abstraction

Plotkin's has shown in [23] that the standard model is ***inequationally fully abstract*** for PCF extended by a parallel conditional construct i.e. for terms s and t such that $? \vdash s : A$ and $? \vdash t : A$ are provable

$$\forall \rho \in Env(?).[\![s]\!]^{\mathcal{S}}_\rho \sqsubseteq [\![t]\!]^{\mathcal{S}}_\rho \quad \Longleftrightarrow \quad s \sqsubseteq^{\mathrm{obs}}_{\approx} t.$$

Theorem 8.4 (Plotkin) *The standard model is inequationally fully abstract for PCFP.*

Proof Suppose $? \vdash s : A$ and $? \vdash t : A$, where $A = (A_1, \cdots, A_n, \alpha)$ and $? = x_1 : B_1, \cdots, x_m : B_m$, are provable in PCFP. We assume that $s \sqsubseteq^{\mathrm{obs}}_{\approx} t$ and aim to show that for any $?$-environment ρ, $[\![s]\!]_\rho \sqsubseteq [\![t]\!]_\rho$. By algebraicity of the domains, it suffices to consider only those ρ such that $\rho(x_j) \in \mathsf{K}(D^{B_j})$ for $1 \leqslant j \leqslant m$. Now take any $d_i \in \mathsf{K}(D^{A_i})$ for $1 \leqslant i \leqslant n$. By the Definability Lemma, there are PCFP terms p_j and q_i that define $\rho(x_j)$ and d_i respectively. Set $C[X] \equiv (\lambda x_1 \cdots x_m.X)p_1 \cdots p_m q_1 \cdots q_n$. We have

$$\begin{array}{rcll}
[\![s]\!]_\rho d_1 \cdots d_n & = & [\![(\lambda x_1 \cdots x_m.s)p_1 \cdots p_m]\!]_{_} d_1 \cdots d_n & \text{Proposition 5.1(ii)} \\
& = & [\![C[s]]\!]_{_} & \\
& \sqsubseteq & [\![C[t]]\!]_{_} & s \sqsubseteq^{\mathrm{obs}}_{\approx} t \text{ and Weak Adequacy} \\
& = & [\![t]\!]_\rho d_1 \cdots d_n. & \text{Proposition 5.1(ii)}
\end{array}$$

By algebraicity again, we have $[\![s]\!]_\rho \sqsubseteq [\![t]\!]_\rho$ and we are done. □

Milner has shown that there is an order-extensional fully abstract model for PCF, and it is unique up to isomorphism [15].

Theorem 8.5 (Milner) *There is a unique (up to isomorphism) inequationally fully abstract, order-extensional, continuous model of PCF.* □

Milner's model is essentially a term model. The construction uses a suitable completion of terms quotiented by an appropriate equivalence relation in a way which is reminiscent of the construction of SFP domains (see [22]).

8.5 A restrictive approach to full abstraction

In [16, 17] Mulmuley gave a construction of Milner's model by a mixture of semantic and syntactic methods. His starting point was the continuous functon space model built up from the standard *lattice* interpretation of the numerals and booleans. First a family of logical relations $d \trianglelefteq s$ indexed by types is defined between elements of the standard model and closed PCF terms as follows:

ground type	$d \trianglelefteq s$	$\iff$	$d \sqsubseteq [\![s]\!]$
function type $A \Rightarrow B$	$d \trianglelefteq s$	$\iff$	$e \trianglelefteq s \implies de \trianglelefteq st$ for every $e \in D^A$ and $t : A$.

For each type A a preorder $\leqslant$ is then defined over the domain D^A:

$$d \leqslant e \quad \iff \quad \text{for every term } s. d \trianglelefteq s \implies e \trianglelefteq s$$

Finally the standard model is quotiented by the equivalence associated with the preorder and then made complete. As every value domain D^A has a top element, the quotients may be turned into a retraction. Milner's fully abstract model is then obtained by removing the top element of each domain.

Problems

8.1 Give an evaluation relation $\Downarrow$ presentation of the operational semantics of PCFP.

8.2 (i) Prove that the one-step reduction relation $>$ of PCFP is Church-Rosser.

(i) Show that the relation $\downarrow$ of PCFP is deterministic i.e. if $s \downarrow v$ and $s \downarrow v'$ then $v = v'$.

8.3 *Parallel conditional* is not PCF definable: there is no closed term $p : o \Rightarrow o \Rightarrow o \Rightarrow o$ that satisfies: for boolean values v and v'

$$p\,\mathtt{t}\,v\,v' \Downarrow v$$

$$p\,\mathtt{f}\,v\,v' \Downarrow v'$$

$$p\,\Omega\,v\,v \Downarrow v$$

8.4 Extend PCF by two parallel conditional constants $\mathsf{p\text{-}cond}^a : o \Rightarrow a \Rightarrow a \Rightarrow a$ where a is either ι or o. Extend the definition of $>$ by the following axioms: for any value v of the appropriate type

$$\begin{aligned} \mathsf{p\text{-}cond}^\alpha\, b\, v\, v &> v \\ \mathsf{p\text{-}cond}^\alpha\, \mathsf{t} &> \lambda x^\alpha y^\alpha . x \\ \mathsf{p\text{-}cond}^\alpha\, \mathsf{f} &> \lambda x^\alpha y^\alpha . y \end{aligned}$$

and extend the definition of evaluation context by the rules:

$$E \quad ::= \quad \mathsf{p\text{-}cond}^\alpha\, E \quad | \quad \mathsf{p\text{-}cond}^\alpha\, b\, E\, s \quad | \quad \mathsf{p\text{-}cond}^\alpha\, b\, s\, E$$

Prove that parallel or is definable in the above extended language, and parallel conditional is definable in PCFP.

8.5 Prove Lemma 8.2.

8.6 Let D and E be Scott domains. Prove that compact elements of the Scott domain $[D \Rightarrow E]$ are exactly $\bigsqcup F$ where F ranges over crisp sets.

9 Full Abstraction and Definability

Theorem 9.1 (Definability) *Let $\mathcal{D}$ be a model of PCF (in the sense of Definition A.5). Suppose $\mathcal{D}$ is enriched over the category* **CPO** *of CPOs and continuous functions, and it is order-extensional.*

(i) The interpretation D^A of each type A is a Scott domain.

(ii) $\mathcal{D}$ is fully abstract if and only if every compact element of each value domain D^A is definable. □

We divide this result into a couple of propositions and prove each in turn.

Proposition 9.2 *Let $\mathcal{D}$ be as given in Theorem 9.1. Then the interpretation D^A of each type A is a Scott domain.*

Proof It suffices to prove: for each type A

- D^A has an inverse system $\{\, \psi_n^A : n \in \omega \,\}$ of projections
- any two elements x, y of D^A have a meet $x \sqcap y$ in D^A.

We show that such semantic objects as the projection function ψ_n^A as well as the meet operation exist by exhibiting the terms $\mathbf{\Psi}_n^A : A \Rightarrow A$ and $\mathsf{glb}^A : A \Rightarrow (A \Rightarrow A)$ that define them respectively.

PCF does not have a constant which tests for equality of two ground values of integer type, though such a test is definable in terms of the test-for-zero constant and the fixed point operator. Given any two integers, decrement both in step until one of them reaches zero, then check if the other is zero. The following term of type $\mathsf{eq} : \iota \Rightarrow \iota \Rightarrow o$ implements the algorithm:

$$\mathbf{Y}^{\iota\Rightarrow\iota\Rightarrow o}(\lambda f^{\iota\Rightarrow\iota\Rightarrow o}.\lambda x^\iota.\lambda y^\iota.\mathsf{cond}^\iota\,(\mathsf{l\text{-}or}(\mathsf{zero}\,x)(\mathsf{zero}\,y))(\mathsf{l\text{-}and}\,(\mathsf{zero}\,x)(\mathsf{zero}\,y))(f(\mathsf{pred}\,x)(\mathsf{pred}\,y)));$$

where the $\mathsf{l\text{-}or}$ ("left or") and $\mathsf{l\text{-}and}$ ("left and") both of type $o \Rightarrow o \Rightarrow o$ are defined as follows:

$$\begin{aligned}
\mathsf{l\text{-}and} &\stackrel{\text{def}}{=} \lambda x^o.\lambda y^o.\mathsf{cond}^o\,x(\mathsf{cond}^o\,y\mathsf{tf})\mathsf{f},\\
\mathsf{l\text{-}or} &\stackrel{\text{def}}{=} \lambda x^o.\lambda y^o.\mathsf{cond}^o\,x\mathsf{t}(\mathsf{cond}^o\,y\mathsf{tf}).
\end{aligned}$$

For each $n \in \omega$ we define

$$\begin{aligned}
\mathbf{\Psi}_n^o &\stackrel{\text{def}}{=} \lambda x^o.x,\\
\mathbf{\Psi}_0^\iota &\stackrel{\text{def}}{=} \lambda x^\iota.\mathsf{cond}^\iota\,(\mathsf{zero}\,x)0\Omega,\\
\mathbf{\Psi}_{n+1}^\iota &\stackrel{\text{def}}{=} \lambda x^\iota.\mathsf{succ}\,(\mathbf{\Psi}_n^\iota(\mathsf{pred}\,x)),\\
\mathsf{glb}^o &\stackrel{\text{def}}{=} \lambda x^o.\lambda y^o.\mathsf{cond}^o\,x(\mathsf{cond}^o\,y\mathsf{t}\Omega)(\mathsf{cond}^o\,y\Omega\mathsf{f}),\\
\mathsf{glb}^\iota &\stackrel{\text{def}}{=} \lambda x^\iota.\lambda y^\iota.\mathsf{cond}^\iota\,(\mathsf{eq}xy)x\Omega.
\end{aligned}$$

It is easy to see that for any $m, n \geqslant 0$ $\mathbf{\Psi}^{\iota}_{n} m \Downarrow n$ if $m \geqslant n$; otherwise, $\mathbf{\Psi}^{\iota}_{n} m \Downarrow m$. Recursively, for any A and B, we define:

$$\begin{aligned}
\mathbf{\Psi}^{A \Rightarrow B}_{n} &\stackrel{\text{def}}{=} \lambda x^{A \Rightarrow B}.\lambda y^{A}.\mathbf{\Psi}^{B}_{n}(x(\mathbf{\Psi}^{A}_{n} y)),\\
\mathsf{glb}^{A \Rightarrow B} &\stackrel{\text{def}}{=} \lambda x^{A \Rightarrow B}.\lambda y^{A \Rightarrow B}.\lambda z^{A}.\mathsf{glb}^{B}(xz)(yz).
\end{aligned}$$

The $(n{+}1)$-ary version of the term glb^{A} is defined as $\mathsf{glb}^{A}_{n+1} \stackrel{\text{def}}{=} \lambda x^{o}_{1} \cdots x^{o}_{n+1}.\mathsf{glb}^{A}(\mathsf{glb}^{A}_{n} x_1 \cdots x_n) x_{n+1}$. $\square$

Proposition 9.3 (Milner) *Let $\mathcal{D}$ be as given in Theorem 9.1. If $\mathcal{D}$ is fully abstract then every compact element of every value domain D^A is definable.*

Proof We prove the contraposition. Suppose there is a type A and a natural number m such that some compact element $\epsilon = \psi^{A}_{m}(\epsilon)$ of D^A is not definable. Let $A = (A_1, \cdots, A_n, \alpha)$ be a "minimal" such type, so that every compact element of D^{A_i} is definable, for each $1 \leqslant i \leqslant n$.

Let K be the subset of definable elements of $\psi^{A}_{m}(D^A)$. We partition $K = K^{+} \cup K^{-}$ by $K^{+} \stackrel{\text{def}}{=} \{\, d \in K : \epsilon \sqsubseteq d \,\}$. Since $\psi^{A}_{m}(D^A)$ is a finite set, we may write $K^{+} = \{\, \gamma_i : 1 \leqslant i \leqslant p \,\}$ and $K^{-} = \{\, \delta_i : 1 \leqslant i \leqslant q \,\}$, for some $p, q \geqslant 0$. By assumption, there are terms $?_i$ and Δ_i which define the elements γ_i and δ_i respectively.

First we consider the case of $K^{+} \neq \varnothing$. Write $? \equiv \mathbf{\Psi}^{A}_{m}(\mathsf{glb}^{A}_{p} ?_1 \cdots ?_p)$ and set $\gamma \stackrel{\text{def}}{=} [\![\, ? \,]\!]_{-}$. We have

$$\gamma = \psi^{A}_{m}(\gamma_1 \sqcap \cdots \sqcap \gamma_p) = \psi^{A}_{m}(\gamma_1) \sqcap \cdots \sqcap \psi^{A}_{m}(\gamma_p) = \gamma_1 \sqcap \cdots \sqcap \gamma_p.$$

Hence we have $\epsilon \sqsubseteq \gamma$ and $\mathbf{\Psi}^{A}_{m}(\gamma) = \gamma$, that is to say, $\gamma \in K^{+}$. Note that $\epsilon \sqsubset \gamma$ since ϵ is not definable. Hence by order-extensionality, there are $\sigma_1 \in D^{A_1}, \cdots, \sigma_n \in D^{A_n}$ such that $\epsilon\sigma_1 \cdots \sigma_n \sqsubset \gamma\sigma_1 \cdots \sigma_n$. Moreover, by the assumption of the minimality of A, there are terms $s_1, \cdots, s_n$ defining $\sigma_1, \cdots, \sigma_n$ respectively. Similarly for $1 \leqslant i \leqslant q$ and $1 \leqslant j \leqslant n$, there are terms t^{i}_{j} defining τ^{i}_{j} such that for each $1 \leqslant i \leqslant q$, $\epsilon\tau^{i}_{1} \cdots \tau^{i}_{n} \not\sqsubseteq \delta_i \tau^{i}_{1} \cdots \tau^{i}_{n}$. We claim:

> There are a ground type α_0 and a compact $d_0 \in D^{\alpha_0}$ such that for any ground type α, and for any compact $d \in D^{\alpha}$, the step function $(d \searrow d_0) : D^{\alpha} \Rightarrow D^{\alpha_0}$ is PCF definable. We write $\lceil (d \searrow d_0) \rceil$ for a term that defines the element $(d \searrow d_0)$.

We can take $\alpha_0 = o$:

$$\begin{aligned}
\lceil (- \searrow \mathsf{t}) \rceil &\equiv \lambda x^{o}.\mathsf{t}\\
\lceil (n \searrow \mathsf{t}) \rceil &\equiv \lambda x^{\iota}.\mathsf{cond}^{o}\,(\mathsf{eq} x n)\mathsf{t}\Omega\\
\lceil (\mathsf{t} \searrow \mathsf{t}) \rceil &\equiv \lambda x^{o}\mathsf{cond}^{o}\, x\mathsf{t}\Omega\\
\lceil (\mathsf{f} \searrow \mathsf{t}) \rceil &\equiv \lambda x^{o}\mathsf{cond}^{o}\, x\Omega\mathsf{t}.
\end{aligned}$$

Consider the two terms of type $A \Rightarrow \alpha_0$:

$$\begin{aligned}
\Xi_1 \stackrel{\text{def}}{=} \lambda x^A.\mathsf{glb}^{\alpha_0}_{q+1}(\lceil(\epsilon\tau^1_1 \cdots \tau^1_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^1_1 \cdots t^1_n)) \\
\vdots \\
(\lceil(\epsilon\tau^q_1 \cdots \tau^q_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^q_1 \cdots t^q_n)) \\
(\lceil(\epsilon\sigma_1 \cdots \sigma_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x s_1 \cdots s_n));
\end{aligned}$$

$$\begin{aligned}
\Xi_2 \stackrel{\text{def}}{=} \lambda x^A.\mathsf{glb}^{\alpha_0}_{q+1}(\lceil(\epsilon\tau^1_1 \cdots \tau^1_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^1_1 \cdots t^1_n)) \\
\vdots \\
(\lceil(\epsilon\tau^q_1 \cdots \tau^q_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^q_1 \cdots t^q_n)) \\
(\lceil(\gamma\sigma_1 \cdots \sigma_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x s_1 \cdots s_n)).
\end{aligned}$$

It is straightforward to see that $[\![\, \Xi_1 \,]\!]_-\epsilon = d_0$ whereas $[\![\, \Xi_2 \,]\!]_-\epsilon = -$. Now we show that $\Xi_1 \approx^{\text{obs}} \Xi_2$. By the Context Lemma 4.5 and the Weak Adequacy Theorem 6.3, it suffices to prove $[\![\, \Xi_1 t \,]\!]_- = [\![\, \Xi_2 t \,]\!]_-$ for any closed term t of type A. Note that $\psi^A_m([\![\, t \,]\!]_-) \in K$. There are two cases:

- $[\![\, \mathbf{\Psi}^A_m t \,]\!]_- \in K^+$: we have $[\![\, \Xi_1 t \,]\!]_- = [\![\, \Xi_2 t \,]\!]_- = d_0$;
- $[\![\, \mathbf{\Psi}^A_m t \,]\!]_- \in K^-$: we have $[\![\, \Xi_1 t \,]\!]_- = [\![\, \Xi_2 t \,]\!]_- = -$.

Next, we consider the case of $K^+ = \varnothing$. The same result can be achieved with the terms:

$$\begin{aligned}
\Xi_1 \stackrel{\text{def}}{=} \lambda x^A.\mathsf{glb}^{\alpha_0}_{q}(\lceil(\epsilon\tau^1_1 \cdots \tau^1_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^1_1 \cdots t^1_n)) \\
\vdots \\
(\lceil(\epsilon\tau^q_1 \cdots \tau^q_n \searrow d_0)\rceil(\mathbf{\Psi}^A_m x t^q_1 \cdots t^q_n));
\end{aligned}$$

$$\Xi_2 \stackrel{\text{def}}{=} \Omega.$$

□

Finally Theorem 9.1 can be proved by the same argument as the proof of Theorem 8.4.

Problems

9.1 Verify that $\{\, \mathbf{\Psi}^A_n \,\}$ and $\{\, \mathsf{glb}^A \,\}$ define respectively a system of projections and the collection of meet operations for the value domains.

Additional Topics

10 Simply-Typed Lambda Calculus

10.1 Friedman's Completeness Theorem

10.2 Logical relations and definability

10.3 Undecidability of lambda definability

10.4 Minimal model

10.5 Loader's decidability result

11 dI-domains and stable functions

A What is a model of PCF?

Definition A.1 A ***c-fix category*** is a cartesian closed category $\mathbb{C}$ equipped with the following additional structure:

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

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

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

$$\begin{tikzcd}
A \times A \cong \mathbf{1} \times A \times A \arrow[r, "\mathsf{t} \times \mathbf{1} \times \mathbf{1}"] \arrow[dr, "\mathsf{fst}"'] & B \times A \times A \arrow[d, "\gamma_A"] & \mathbf{1} \times A \times A \cong A \times A \arrow[l, "\mathsf{f} \times \mathbf{1} \times \mathbf{1}"'] \arrow[dl, "\mathsf{snd}"] \\
& A &
\end{tikzcd}$$

commutes.

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

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

with the property that the diagram

$$\begin{tikzcd}
(A \Rightarrow A) \times (A \Rightarrow A) \arrow[r, "\mathbf{1} \times \mathsf{Y}_A"] & (A \Rightarrow A) \times A \arrow[d, "\mathsf{ev}"'] \\
A \Rightarrow A \arrow[u, "\Delta"] \arrow[r, "\mathsf{Y}_A"'] & A
\end{tikzcd}$$

commutes.

Definition A.2 Suppose that $\mathbb{V}$ is a symmetric monoidal category. By a ***c-fix category enriched over*** $\mathbb{V}$ we mean the following: a category $\mathbb{C}$ enriched over $\mathbb{V}$,

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

(ii) whose underlying category is an ordinary c-fix category.

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

Definition A.3 Suppose that $\mathbb{C}$ is a category with finite products and that N is an object of $\mathbb{C}$ equipped with maps

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

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

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

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

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

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

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

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

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

Definition A.4 Suppose that $\mathbb{C}$ is a category with finite products (a terminal object suffices) equipped with a diagram

$$\mathbf{1} \underset{\mathsf{f}}{\overset{\mathsf{t}}{\rightrightarrows}} B.$$

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

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

$$N \xrightarrow{\;p\;} N \qquad\qquad N \xrightarrow{\;z\;} B.$$

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

when the diagrams

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

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

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

Definition A.5 (Concrete version) A ***categorical model for PCF*** is a c-fix category equipped with a simple object of numerals.

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

In general such categories will not be concrete in the natural way; that is, the global sections functor will not be faithful. When it is, that is when

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

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

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

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

Problems

A.1 Let $\mathcal{C}$ be a categorical model of PCF. Show that $\mathcal{C}$ is a a model of the PCF type theory.

B A Sample Miniproject

MFoCS Miniproject

Correspondence between Operational and Denotational Semantics

---

Do as many of the following as you can, but only one of Problems 4 and 5.

1. Prove that for any program s, $s \Downarrow v$ if and only if $s \downarrow v$.

[Hint: define a family of evaluation relations $\Downarrow_n$, one for each $n \geqslant 0$, between closed terms and values by mutual rule induction where $s \Downarrow_n v$ means intuitively "s evaluates to v in n steps".]

2. Prove that there is no closed term $F : (o \Rightarrow o \Rightarrow o) \Rightarrow o$ such that for any closed $p : o \Rightarrow o \Rightarrow o$ such that $p\ s\ t\downarrow$ for some closed s and t of type o

(i) if $p\ \Omega^o\ t\uparrow$ for all t then $Fp \downarrow 1$

(ii) if $p\ s\ \Omega^o\uparrow$ for all s then $Fp \downarrow 2$.

Note that F is able to ascertain which of the two input arguments of p is evaluated (by p) first.

3. Prove that the denotational semantics induced by the standard interpretation is a model of PCF Type Theory: if $? \vdash s = t : A$ is provable then $[\![s]\!]^{\mathcal{S}}_{\rho} = [\![t]\!]^{\mathcal{S}}_{\rho} \in D^A$ for any ?-environment ρ.

4. **(i)** How would you extend the logical relation $\lhd^A$ (as introduced in Section 6) to terms which are not necessarily closed?

[Hint: Consider (closed) substitution σ which maps variables to type-compatible closed terms.]

(ii) Use the new definition to rework the proof of the Context Lemma. You should not (need to) use the $S^{A,B,C}$ and $K^{A,B}$ terms in your argument.

5. Define a new type-indexed family of logical relations $\lhd^A \subseteq D^A \times \mathrm{PCF}^o_A$ as follows:

$$d \lhd^{\alpha} s \iff d \in \mathbb{N} \cup \mathbb{B} \implies s \Downarrow d$$

$$d \lhd^{A \Rightarrow B} s \iff \forall e \in D^A . \forall t \in \mathrm{PCF}^o_A . e \lhd^A t \implies de \lhd^B st.$$

Use the logical relations to prove the Weak Adequacy Theorem.

C-HLO May 1999

References

[1] H. Barendregt. *The Lambda Calculus.* North-Holland, revised edition, 1984.

[2] G. Berry. Modèles complètement adéquats et stables des lambda calculs typés. Technical report, Université Paris VII, 1979. Thèse de Doctorat d'Etat.

[3] A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[4] R. L. Crole. *Categories for Types.* Cambridge University Press, 1993.

[5] P.-L. Curien. Observable algorithms on concrete data structures. *Information and Computation*, 1993. To appear.

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

[7] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, pages 280–287, 1958.

[8] K. Gödel. *Kurt Gödel Collected Works, Volumes I and II,* S. Feferman, *et al* (editors). Oxford Univ. Press, 1990.

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

[10] C. A. Gunter and D. S. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 635–674. Elsevier, 1990.

[11] S. C. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. American Mathematical Society*, 91:1–52, 1959.

[12] S. C. Kleene. Recursive functionals and quantifiers of finite types II. *Trans. American Mathematical Society*, 108:106–142, 1963.

[13] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Mathematics No. 7. Cambridge University Press, 1986.

[14] P. Martin-Löf. Constructive mathematics and computer programming. In *International Congress for Logic, Methodology and Philosophy of Science*, pages 538–571. North-Holland, 1979.

[15] R. Milner. Fully abstract models of typed lambda-calculus. *Theoretical Computer Science*, 4:1–22, 1977.

[16] K. Mulmuley. Fully abstract submodels of typed lambda calculus. *Journal of Computer and System Sciences*, 33:2–46, 1986.

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

[18] A. M. Pitts. Computational adequacy via "mixed" inductive definitions. In *Proc. 18th Int. Symp. Mathematical Foundations of Computer Science, IX, New Orleans, 1993.* Springer, 1994. LNCS. To appear.

[19] R. A. Platek. *Foundations of Recursion Theory.* PhD thesis, Standford University, 1966.

[20] G. D. Plotkin. A set-theoretical definition of application. Technical Report MIP-R-95, School of A.I., Univ. of Edinburgh, 1972.

[21] G. D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[22] G. D. Plotkin. A powerdomain construction. *SIAM J. Computing*, 5(3):452–487, 1976.

[23] G. D. Plotkin. LCF as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[24] G. D. Plotkin. Cpo's: Tools for making meanings. Post-Graduate Lecture Notes in Advanced Domain Theory, Dept. of Computer Science, Univ. of Edinburgh, 1981.

[25] G. D. Plotkin. Types and partial functions. Post-Graduate Lecture Notes, Dept. of Computer Science, University of Edinburgh, 1985.

[26] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM and OWHY. *Theoretical Computer Science*, 121:411–440, 1993.

[27] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In *Recursive Function Theory,* Proc. Symposia in Pure Mathematics V, pages 1–27. American Mathematical Society, Providence, RI, 1962.

[28] R. Statman. Logical relations and the typed λ-calculus. *Information and Control*, 65:85–97, 1985.

[29] W. W. Tait. Intensional interpretation of functionals of finite type i. *J. Symb. Logic*, 32:198–212, 1967.

[30] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Mathematics*, 5:285–309, 1955.

Index

?-environment, 22
ω-algebraic, 32

active redex, 11
adequate, 25
algebraic, 32
applicatively approximates, 16

bifinite CPO, 33

c-fix category, 45
c-fix category enriched over , 45
categorical model for PCF, 47
collection of value domains, 21
compatible, 32
computability, 26
Condition (M), 33
conditionals, 45
consistent, 32
consistently complete, 32
constants, 7
Contexts, 8
crisp, 36

evaluation contexts, 10
extensional, 47

fixed points, 45
fully abstract, 25

inequationally fully abstract, 38
interpretation, 21
inverse system of projections, 32

numeralwise represents, 46

observationally approximates, 16
observationally equivalent, 16
one-step reduction, 10
order-extensional, 47

PCF theories, 17
PCF Type Theory, 13
program, 8

rank, 6
redexes, 10

Scott domain, 32
SFP domain, 33
simple object of numerals, 46
standard interpretation, 21
step functions, 36
strongly algebraic, 33
strongly definable, 14

term, 8
theory induced by $\mathcal{I}$, 25
types, 6
Typing judgements, 7

valid, 25
value, 10

weakly adequate, 25