Domain-Theoretic Models of Typed Lambda Calculus

Hellis Tamm

March 14, 2000

1 Introduction

We describe a class of models for typed lambda calculus based on partially-ordered structures called *domains*. The main motivation here for studying domains is that they provide models with fixed-point operators. We consider the simplest class of domains, called *complete partial orders,* (*CPOs* for short)

We start by introducing general or Henkin models in Section 2. Then in Section 3 we discuss recursive definitions and fixed point operators. Section 4 and 5 present the notions of complete partial orders and function continuity, respectively, and in Section 6 we present the Henkin model of typed lambda calculus. Finally, in Section 7 we give a CPO model for PCF.

The presentation is based on Sections 4.5, 5.1 and 5.2 of [Mit96].

2 Henkin Models

Generally, a model provides means for giving mathematical meaning to expressions of a logical system, to determine whether any formula of the logic is true. For logics with variables, it is common to separate the assignment of values to variables from the general notion of model. For example, in algebra we have a set of possible values for each variable and an interpretation for each function symbol. We choose values for variables (select an environment) and use the functions to determine the value of an algebraic term. Then we can determine whether two terms are equal or not.

Similarly, in a model for typed lambda calculus we require a set of possible values for each variable and an interpretation for each constant symbol of the signature. We also need to make sense of application and lambda abstraction. Suppose $A^{\sigma\to\tau}$ is the set of values of type $\sigma \to \tau$, and A^σ is the set of values of type σ. If $f \in A^{\sigma\to\tau}$ and $x \in A^\sigma$, then we need to be able to apply f to x. But we cannot simply say that $A^{\sigma\to\tau}$ must be the set of all functions from A^σ to A^τ because every function in our model must have a fixed point but not all set-theoretic functions from σ to τ have fixed points. This forces us to define models using a more flexible condition on function spaces.

We will define a general model called *Henkin model*. The definition has three parts consisting of definition of *typed applicative structure* with two conditions. A typed applicative structure consists of families of sets and mappings indexed by types, or pairs of types, of the form necessary to give meanings to terms. To be a model, an applicative structure must be extensional which means that the interpretation of a function type must be a set of functions, and there must be enough elements so that every lambda-definable function is in the model which we specify as environment model condition.

Definition A *typed applicative structure* $\mathcal{A}$ for $\lambda^{\to}$ signature Σ is a tuple
$\langle \{A^\sigma\}, \{\mathbf{App}^{\sigma,\tau}\}, Const \rangle$
of families of sets and mappings indexed by type expressions over the type constants from Σ. For each σ and τ we assume the following conditions:
· A^σ is a set,
· $\mathbf{App}^{\sigma,\tau}$ is a map $\mathbf{App}^{\sigma,\tau} : A^{(\sigma\to\tau)} \to A^\sigma \to A^\tau$,
· $Const$ is a map from term constants of Σ to elements of the union of all the A^σ's such that if $c : \sigma$, then $Const(c) \in A^\sigma$.

Definition An applicative structure is *extensional* if it satisfies the condition
· For all $f, g \in A^{\sigma\to\tau}$, if $\mathbf{App}\ f\ d = \mathbf{App}\ g\ d$ for all $d \in A^\sigma$, then $f = g$.

In place of applicative structures, it is possible to use a definition of model that assumes extensionality if we require that $A^{\sigma\to\tau}$ is not just any set but a set of functions from A^σ to A^τ.

Definition A *type frame* is an applicative structure $\langle \{A^\sigma\}, \{\mathbf{App}^{\sigma,\tau}\}, Const \rangle$ such that $A^{\sigma\to\tau} \subseteq A^{\tau(A^\sigma)}$ and $\mathbf{App}^{\sigma,\tau}\ f\ d = f(d)$.

Since $\mathbf{App}$ is always function application in any type frame, we can just write $\mathcal{A} = \langle \{A^\sigma\}, Const \rangle$. An example type frame is the *full set-theoretic* *function hierarchy* over any base types where $A^{\sigma\to\tau}$ contains all set-theoretic functions from A^σ to A^τ.

We now formalize the environment model condition.

Definition An *environment* η for applicative structure $\mathcal{A}$ is a mapping from variables to the union of all A^σ. If Γ is a type assignment, then we say η *satisfies* Γ ($\eta \models \Gamma$) if $\eta(x) \in A^\sigma$ for every $x:\sigma \in \Gamma$. If $d \in A^\sigma$ then $\eta[x \mapsto d]$ is the environment mapping x to d.

An applicative structure $\mathcal{A}$ satisfies the *environment model condition* if the clauses below define a total meaning function $\mathcal{A}[\![\cdot]\!]\cdot$ on terms $\Gamma \triangleright M:\sigma$ and environments η such that $\eta \models \Gamma$. The following inductive clauses correspond to typing axioms (*cst*) and (*var*), and typing rules (*add var*), ($\to$ Elim) and ($\to$ Intro).

$$
\begin{array}{rcl}
\mathcal{A}[\![\emptyset \triangleright c:\sigma]\!]\eta &=& Const(c)\\
\mathcal{A}[\![x:\sigma \triangleright x:\sigma]\!]\eta &=& \eta(x)\\
\mathcal{A}[\![\Gamma, x:\sigma \triangleright M:\tau]\!]\eta &=& \mathcal{A}[\![\Gamma \triangleright M:\tau]\!]\eta\\
\mathcal{A}[\![\Gamma \triangleright MN:\tau]\!]\eta &=& \mathbf{App}^{\sigma,\tau}\mathcal{A}[\![\Gamma \triangleright M:\sigma\to\tau]\!]\eta\mathcal{A}[\![\Gamma \triangleright N:\sigma]\!]\eta\\
\mathcal{A}[\![\Gamma \triangleright \lambda x:\sigma.M:\sigma\to\tau]\!]\eta &=& \text{the unique } f \in A^{\sigma\to\tau} \text{ such that}\\
&& \forall d \in A^\sigma.\ \mathbf{App}\ f\ d = \mathcal{A}[\![\Gamma, x:\sigma \ \triangleright M:\tau]\!]\eta[x \mapsto d]
\end{array}
$$

To obtain Henkin models of other versions of simply-typed lambda calculus, we need to extend the definitions of applicative structure, extensionality, and the environmental model condition.

For $\lambda^{\times,\to}$, a typed applicative structure is a tuple

$\langle \{A^\sigma\}, \{\mathbf{Proj_1}^{\sigma,\tau}, \mathbf{Proj_2}^{\sigma,\tau}, \mathbf{App}^{\sigma,\tau}\}, Const\rangle$ with

$\cdot$ $\mathbf{Proj_1}^{\sigma,\tau} : A^{\sigma\times\tau} \to A^\sigma$,

$\cdot$ $\mathbf{Proj_2}^{\sigma,\tau} : A^{\sigma\times\tau} \to A^\tau$.

Extensionality condition for products would be like
$\forall p, q \in A^{\sigma\times\tau}.\ (\mathbf{Proj_1}p = \mathbf{Proj_1}q \wedge \mathbf{Proj_2}p = \mathbf{Proj_2}q) \supset p = q.$

For $\lambda^{\times,\to}$ frame we have the following conditions in addition to the conditions on $A^{\sigma\to\tau}$ and **App**:

· $A^{\sigma\times\tau} \subseteq A^\sigma \times A^\tau$ is a set of ordered pairs
· $\mathbf{Proj}_1^{\sigma,\tau}\langle x, y\rangle = x$ and $\mathbf{Proj}_2^{\sigma,\tau}\langle x, y\rangle = y$

The meaning of a $\lambda^{\times,\to}$ term is given by extending the definition of the meaning function for $\lambda^{\to}$ terms with the additional clauses of

$$\begin{array}{rcl}
[\![\Gamma \triangleright \mathbf{Proj}_1^{\sigma,\tau} M : \sigma]\!]\eta & = & \mathbf{Proj}_1^{\sigma,\tau}[\![\Gamma \triangleright M : \sigma\times\tau]\!]\eta \\
[\![\Gamma \triangleright \mathbf{Proj}_2^{\sigma,\tau} M : \tau]\!]\eta & = & \mathbf{Proj}_2^{\sigma,\tau}[\![\Gamma \triangleright M : \sigma\times\tau]\!]\eta \\
[\![\Gamma \triangleright \langle M, N\rangle : \sigma\times\tau]\!]\eta & = & \text{the unique } p \in A^{\sigma\times\tau} \text{ such that} \\
 & & \ \mathbf{Proj}_1^{\sigma,\tau} p = [\![\Gamma \triangleright M : \sigma]\!]\eta \ \text{ and} \\
 & & \ \mathbf{Proj}_2^{\sigma,\tau} p = [\![\Gamma \triangleright N : \tau]\!]\eta
\end{array}$$

3 Recursive Definitions and Fixed Point Operators

For recursive function definitions we have a fix-point operator fix_σ that returns a fixed point of any function from σ to σ. We will use properties of fix reduction to motivate the semantic interpretation of fix. Let's review the factorial example. We can write $fact \stackrel{\text{def}}{=} fix_{nat\to nat}\ F$, where

$$F \stackrel{\text{def}}{=} \lambda f : nat \to nat.\lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * f(y-1).$$

We compute $fact\ n$:

$$\begin{array}{rcl}
fact\ n & = & (\lambda f : nat \to nat.\ f(fix\ f))\ F\ n \\
 & = & F\ (fix\ F)\ n \\
 & = & (\lambda f : nat \to nat.\lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * f(y-1))(fix\ F)\ n \\
 & = & \texttt{if}\ Eq?\ n\ 0\ \texttt{then}\ 1\ \texttt{else}\ n * (fix\ F)(n-1)
\end{array}$$

When $n = 0$, we can use the reduction axiom for conditional to simplify $fact\ 0$ to 1. For $n > 0$, we obtain $n * (fix\ F)(n-1)$, and continue as above. For any numeral n, we will evantually reduce $fact\ n$ to the numeral for $n!$.

Another approach to understanding $fact$ is to consider the finite expansions of $fix\ F$. Let us think of $nat \to nat$ as a collection of partial functions on the natural numbers, represented by sets of ordered pairs. Using a constant *diverge* for the nowhere defined function (the empty set), we define

$fix^{[0]}\ F = diverge$ and $fix^{[n+1]}\ F = F\ (fix^{[n]}\ F)$. Computationally, $fix^{[n]}\ F$ describes the recursive function computed using at most n evaluations of the body of F. The finite expansions of $fix\ \ F$ are linearly ordered by set-theoretic containment. Namely, $fix^{[0]}\ F = \emptyset$ is the least element in this ordering, and $fix^{[n+1]}\ F = (fix^{[n]}\ F) \cup \langle n, n! \rangle$ properly contains all $fix^{[i]}\ F$ for $i \leq n$. Since every terminating computation involving factorial uses only a finite number of recursive calls, we can let $fact = \bigcup_n (fix^{[n]}\ F)$. And since any fixed point of F must contain the functions defined by finite expansions of F, $\bigcup_n (fix^{[n]}\ F)$ will be the least fixed point of F.

A specific problem related to recursion is the interpretation of terms that define nonterminating computations. For example, it is impossible to simplify the PCF expression

$\texttt{letrec}\ f : nat \to nat = \lambda x : nat.f(x+1)\ \texttt{in}\ f\ 3$

to a numeral, although the type of this expression is nat. Instead of saying that the value of this expression is undefined, we include an additional value $\perp_{nat}$ to the domain of natural numbers to represent nonterminating computations of type nat. Since the ordering of a domain is intended to characterize a "degree of definedness", we consider $\perp_{nat}$ to be the least element in the ordering on the domain of natural numbers.

4 Complete Partial Orders, Lifting and Cartesian Products

Now we describe complete partial orders, ordered structures with useful properties for our purposes. But first we need a few definitions.

Definitions

A *partial order* $\langle D, \leq \rangle$ is a set D with a relation $\leq$ that is reflexive ($d \leq d$ for every $d \in D$), anti-symmetric (if $d \leq d'$ and $d' \leq d$ then $d = d$) and transitive (if $d \leq d'$ and $d' \leq d''$ then $d \leq d''$). Any set can be considered a partial order using the *discrete order* $x \leq y$ iff $x = y$.

If $\langle D, \leq \rangle$ is a partial order, then an *upper bound* of a subset $S \subseteq D$ is an element $x \in D$ with $y \leq x$ for every $y \in S$. A *least upper bound* of S is an upper bound which is $\leq$ every upper bound of S.

If $\langle D, \leq \rangle$ is a partial order, then a subset $S \subseteq D$ is *directed* if every finite $S_0 \subseteq S$ has an upper bound in S.

A partial order $\langle D, \leq \rangle$ is a *complete partial order* (*CPO* for short) if every directed $S \subseteq D$ has a least upper bound, written $\bigvee S$.

Examples

Any set may be considered a complete partial order using the discrete order. Any finite partial order is a CPO. Any closed interval of the real line (under the standard order) is a CPO. With respect to the ordinary arithmetic ordering, the natural numbers are not a CPO because although the set $\mathcal{N}$ is directed, it has no least upper bound. But if we add an extra element ∞ greater than all the ordinary numbers then we have a CPO.

In interpreting fix, we will be particularly interested in CPOs that have least elements.

Definition If $\mathcal{D} = \langle D, \leq \rangle$ is a partial order with least element then we say $\mathcal{D}$ is *pointed*. We write $\perp_D$ for the least element of D.

Lifting

We can create a pointed CPO from any set by *lifting*. Namely, if A is any set then we write $A_\perp$ for the CPO whose elements are $A \cup \{\perp\}$, ordered so that $x \leq y$ iff $x = y$ or $x = \perp$. We may also lift any CPO $\mathcal{D} = \langle D, \leq_D \rangle$, and the result $\mathcal{D}_\perp$ has elements $D \cup \{\perp\}$, where $\perp$ is distinct from all elements of D. We order $\mathcal{D}_\perp$ so that $x \leq y$ iff $x = \perp$ or $x, y \in D$ with $x \leq_D y$.

Lemma 1 If $\mathcal{D}$ is a CPO then $\mathcal{D}_\perp$ is a pointed CPO (CPO with a least element).

Two examples are the lifted natural numbers $\mathcal{N}_\perp$ and the lifted booleans $\mathcal{B}_\perp$.

Cartesian products

If $\mathcal{D} = \langle D, \leq_D \rangle$ and $\mathcal{E} = \langle E, \leq_E \rangle$ are CPOs, we define a cartesian product CPO as follows. We let $\mathcal{D} \times \mathcal{E} = \langle D \times E, \leq_{D \times E} \rangle$, where the ordering $\leq_{D \times E}$ is defined as follows:

$\langle d, e \rangle \leq_{D \times E} \langle d', e' \rangle$ iff $d \leq_D d'$ and $e \leq_E e'$

For $S \subseteq D \times E$ we define sets $\mathbf{Proj}_1 S = \{d \mid \langle d, e \rangle \in S\}$ and $\mathbf{Proj}_2 S = \{e \mid \langle d, e \rangle \in S\}$.

Lemma 2 Suppose $\mathcal{D}$ and $\mathcal{E}$ are CPOs. The product $\mathcal{D} \times \mathcal{E}$ is a CPO which is pointed if both $\mathcal{D}$ and $\mathcal{E}$ are pointed. If $S \subseteq D \times E$ is directed then $\bigvee S = \langle \bigvee S_1, \bigvee S_2 \rangle$, where $S_i = \mathbf{Proj}_i S$.

5 Continuous Functions

As a part of constructing a model with each type a CPO we need to be able to interpret function types as CPOs. We now give a definition of function continuity and state that the collection of all continuous functions from one CPO to another forms a CPO.

Definition Suppose $\mathcal{D} = \langle D, \leq_D \rangle$ and $\mathcal{E} = \langle E, \leq_E \rangle$ are CPOs and $f : D \to E$ is a function on the underlying sets. If $S \subseteq D$, we write $f(S)$ for the subset of E given by $f(S) = \{f(d) \mid d \in S\}$. We say f is *monotonic* if $d \leq d'$ implies $f(d) \leq f(d')$. It can be seen that if f is monotonic and S is directed then $f(S)$ is directed. Now, a monotonic function f is *continuos* if for every directed $S \subseteq D$ we have $f(\bigvee S) = \bigvee f(S)$.

Examples

Every function on a discretely ordered CPO is continuous. A constant function on any CPO is continuous. Any continuous function in the ordinary sense of calculus, on a closed interval $[x, y]$ of the real line is also a continuous function when we regard $[x, y]$ as a CPO. Any monotonic function from a lifted set $A_\perp$ to any CPO is continuous because in $A_\perp$ all nontrivial directed sets have the form $\{\perp, a\}$ and every monotonic function f must map $\bigvee\{\perp, a\} = a$ to $\bigvee f(\{\perp, a\}) = \bigvee\{f(\perp), f(a)\} = f(a)$.

Definition If $\mathcal{D}$ and $\mathcal{E}$ are CPOs, and $f : D \to E$ is continuous, we define the *lifted function* $f_\perp : (D \cup \{\perp\}) \to (E \cup \{\perp\})$ by

$$f_\perp(d) = \begin{cases} f(d) & \text{if } d \in D \\ \perp & \text{otherwise} \end{cases}$$

Definition A function f on pointed CPOs is *strict* if $f(\perp) = \perp$.

Lemma 3 If $\mathcal{D}$ and $\mathcal{E}$ are CPOs, and $f : \mathcal{D} \to \mathcal{E}$ is continuous, then $f_\perp : \mathcal{D}_\perp \to \mathcal{E}_\perp$ is strict and continuous.

We want to view the collection of continuous functions from one CPO to another as a CPO. For this we will partially order functions point-wise as follows. Let $\mathcal{D} = \langle D, \leq_D \rangle$ and $\mathcal{E} = \langle E, \leq_E \rangle$ be CPOs. For continuous $f, g : D \to E$, we say $f \leq_{D \to E} g$ if for every $d \in D$, we have $f(d) \leq_E g(d)$. We write $\mathcal{D} \to \mathcal{E} = \langle D \to E, \leq_{D \to E} \rangle$ for the collection of continuous functions from $\mathcal{D}$ to $\mathcal{E}$, ordered pointwise. For a set of functions $S \subseteq D \to E$ and $d \in D$ we have a notation for a set $S(d) = \{f(d) \mid f \in S\}$.

Lemma 4 For any CPOs $\mathcal{D}$ and $\mathcal{E}$, the collection $\mathcal{D} \to \mathcal{E}$ of continuous functions, ordered point-wise, is a CPO. In particular, if $S \subseteq D \to E$ is directed, the least upper bound is the function f given by $f(d) = \bigvee S(d)$. The CPO $\mathcal{D} \to \mathcal{E}$ has a least element if $\mathcal{E}$ does.

Lemma 5 Pairing, projection, application and composition are continuous:
(a) If $S \subseteq D$ and $T \subseteq E$ are directed, then $\langle \bigvee S, \bigvee T \rangle = \bigvee \{\langle s, t \rangle \mid s \in S, t \in T\}$,
(b) If $S \subseteq D \times E$ is directed, then $\mathbf{Proj}_i(\bigvee S) = \bigvee \{\mathbf{Proj}_i(x) \mid x \in S\}$,
(c) If $S \subseteq D \to E$ and $T \subseteq D$ are directed, then $\bigvee S(\bigvee T) = \bigvee \{f(x) \mid f \in S, x \in T\}$,
(d) If $S \subseteq D \to E$ and $T \subseteq E \to F$ are directed, then $(\bigvee S) \circ (\bigvee T) = \bigvee \{g \circ f \mid f \in S, g \in T\}$.

6 Fixed Points and the Full Continuous Hierarchy

Our aim is to construct Henkin models of typed lambda calculi with fixed-point operators. The type frame, $\mathcal{A}$, called the *full continuous* $\lambda^{\times,\to}$ *hierarchy over* CPOs $\langle A^{b_0}, \leq_0 \rangle, \ldots, \langle A^{b_k}, \leq_k \rangle$ is defined by taking $A^{b_0}, \ldots, A^{b_k}$ as base types and

$A^{\sigma \times \tau} = A^\sigma \times A^\tau$

$A^{\sigma \to \tau} =$ all continuous $f : \langle A^\sigma, \leq_\sigma \rangle \to \langle A^\tau, \leq_\tau \rangle$

with $A^{\sigma \times \tau}$ ordered coordinate-wise by $\leq_{\sigma \times \tau}$ and $A^{\sigma \to \tau}$ ordered point-wise by $\leq_{\sigma \to \tau}$. By Lemmas 2 and 4, each $\langle A^\sigma, \leq_\sigma \rangle$ is a CPO. The main result here will be that the full continuous type hierarchy over any CPOs forms a Henkin model. But first let's consider the interpretation of *fix*.

Definition We say a is the *least fixed point* of f if $a = f(a)$ and, whenever $b = f(b)$, we have $a \leq b$.

The following lemma says that the least fixed point of a continuous function f is the least upper bound of the set obtained by repeatedly applying f to the least element of its domain.

Lemma 6 If $\mathcal{D}$ is a pointed CPO and $f : \mathcal{D} \rightarrow \mathcal{D}$ is continuous, then f has a least fixed point

$fix_D\ f = \bigvee\{f^n(\bot) \mid n \geq 0\}$

and the map fix_D is continuous.

Example A trivial example is $fix\ id$, where $id : \mathcal{D} \rightarrow \mathcal{D}$ is the identity function on the pointed CPO $\mathcal{D}$. We can calculate the least fixed point of id as follows.

$$\begin{aligned} fix\ id &= \bigvee\{id^n(\bot_D) \mid n \geq 0\} \\ &= \bigvee\{\bot_D\} \\ &= \bot_D \end{aligned}$$

This is the right answer because $\bot$ is a fixed point of the identity function and the least element of D.

Example Let Σ be an algebraic signature and let $\mathcal{E}$ be the set of all well-formed equations between Σ-terms. Let $\mathcal{C}$ be the set of all subsets of $\mathcal{E}$. Then $\mathcal{C}$ is a CPO, ordered by set containment. Let $F : \mathcal{C} \rightarrow \mathcal{C}$ be the function mapping any set $H \in \mathcal{C}$ to the set of all equations that are either axioms, elements of H or provable from H with a single inference rule ($F(H) = A \cup H \cup P(H)$). It can be shown that:

(a) F is monotonic. (If $H_1 \subseteq H_2$ then $P(H_1) \subseteq P(H_2)$ and therefore $F(H_1) \subseteq F(H_2)$.)

(b) F is continuous. (Let C be a directed subset of $\mathcal{C}$. The least upper bound of C is $\bigcup C$. It can be shown that $F(\bigcup C) = \bigcup F(C)$.)

(c) The least fixed point of F is the set of all valid formulas. (It may be proved that $F^n(\emptyset)$ is the set of all equations provable from axioms with proofs of depth less than n. Since every proof is finite, $fix\ F = \bigcup\{F^n(\emptyset) \mid n \geq 0\}$ is the set of all equations provable from the axioms.)

(d) Let $T(H) = \bigvee\{F^n(H) \mid n \geq 0\}$, $H \in \mathcal{C}$. Then $E \in T(H)$ iff E is provable from H. (Similarly to (c), $F^n(H)$ is the set of all equations provable from axioms and equations of H with proofs of depth less than n, and $T(H) = \bigcup\{F^n(H) \mid n \geq 0\}$ is the set of all equations provable from axioms and H.)

We now proceed towards the Henkin model by stating a few more lemmas. Lemmas 7 and 8 are used for inductive proof of the environment model condition.

Lemma 7 If $f : \mathcal{C} \to (\mathcal{D} \to \mathcal{E})$ and $g : \mathcal{C} \to \mathcal{D}$ are continuous, then so is the function $App\ f\ g$ given by $(App\ f\ g)\ c = (f\ c)(g\ c)$ for all $c \in C$. Also the map App is continuous.

Lemma 8 If $f : \mathcal{C} \times \mathcal{D} \to \mathcal{E}$ is continuous then there is a unique continuous function $(Curry\ f) : \mathcal{C} \to (\mathcal{D} \to \mathcal{E})$ such that for all $c \in C$ and $d \in D$, $(Curry\ f)\ c\ d = f\langle c, d\rangle$. Also the map $Curry$ is continuous.

Lemma 9 The full continuous hierarchy over any collection of CPOs is a Henkin model.

This is shown by seeing that the full continuous hierarchy for $\lambda^{\times,\to}$ is an extensional applicative structure and by proving that the environment model condition is satisfied (using Lemmas 7 and 8).

The following theorem is a consequence of Lemmas 6 and 9.

Theorem 10 The full continuous $\lambda^{\times,\to}$ hierarchy over any collection of CPOs is a Henkin model with the property that whenever A^σ is pointed there is a least fixed point operator $fix_\sigma \in A^{\sigma\to\sigma}$.

If A^τ is pointed then $A^{\sigma\to\tau}$ is pointed, and if both A^σ and A^τ are pointed then $A^{\sigma\times\tau}$ is pointed. If we choose pointed CPOs for the type constants, then every A^σ is pointed and we have least-fixed-point operators at all types.

7 CPO Model for PCF

The CPO model for PCF, $\mathcal{A}_{\text{PCF}}$ is the full continuous hierarchy over $A^{nat}_{\text{PCF}} = \mathcal{N}_\perp$ and $A^{bool}_{\text{PCF}} = \mathcal{B}_\perp$. Since the type constants of PCF are interpreted as pointed CPOs, and cartesian product and continuous function space constructors preserve pointedness, all of the types of PCF are interpreted as pointed CPOs. So we have least-fixed-point operators at all types and we can interpret the fixed-point constants of PCF as in Lemma 6.

We interpret constants $0, 1, 2, \ldots$ and $true$, $false$ as the standard natural number and boolean elements of the lifted sets $\mathcal{N}_\perp$ and $\mathcal{B}_\perp$. If we choose continuous functions for basic PCF operations $+$, $Eq?$ and conditional, then every term has a meaning in $\mathcal{A}_{\text{PCF}}$ (Theorem 10).

We interpret $+$ and $Eq?$ as the lifted versions of the standard functions. Namely, we interpret $+$ as the extension of addition with $\perp_{nat} + x = x + \perp_{nat} = \perp_{nat}$ for $x \in A^{nat}_{\text{PCF}}$. Similarly, $Eq?\ M\ N$ has value $true$ if M and N are the same element of $\mathcal{N}_\perp$ different from $\perp$, value $false$ if M and N are different and neither of them is $\perp$, and value $\perp$ if either M or N is $\perp$. The equational axioms are satisfied by this interpretation since they only mention sums and equality tests between non-$\perp$ elements of $\mathcal{N}_\perp$.

For conditional we cannot use the same simple approach because in the case that M is $\perp$ but N is not, we would have

$\texttt{if}\ false\ \texttt{then}\ M\ \texttt{else}\ N = \perp$,

which contradicts the equational axiom of PCF.

The interpretation of conditional is the following.

$$[\![\ \texttt{if}\ P\ \texttt{then}\ M\ \texttt{else}\ N\ ]\!]\eta = \begin{cases} [\![M]\!]\eta & \text{if } [\![P]\!]\eta = true \\ [\![\mathrm{N}]\!]\eta & \text{if } [\![P]\!]\eta = false \\ \perp & \text{otherwise} \end{cases}$$

The equational axioms are satisfied because they only mention conditional expressions when the first argument is $true$ or $false$. This completes the definition of $\mathcal{A}_{\text{PCF}}$.

The following statements about soundness show that if we can prove $M = N$ or reduce one term to the other then the terms have the same meaning in $\mathcal{A}_{\text{PCF}}$.

Definition A model $\mathcal{A}$ *satisfies* an equation $\Gamma \triangleright M = N : \sigma$ if $\mathcal{A}[\![\Gamma \triangleright M : \sigma]\!]\eta = \mathcal{A}[\![\Gamma \triangleright N : \sigma]\!]\eta$ for every η satisfying Γ.

Theorem 11 Let M and N be expressions of PCF over typed variables from Γ. If $\Gamma \triangleright M = N : \sigma$ is provable from the axioms for PCF, then the model $\mathcal{A}_{\mathrm{PCF}}$ satisfies the equation $\Gamma \triangleright M = N : \sigma$.

Corollary 12 If $\Gamma\ M : \sigma$ is a well-typed term of PCF, and $M \twoheadrightarrow N$, then $\mathcal{A}_{\mathrm{PCF}}$ satisfies the equation $\Gamma \triangleright M = N : \sigma$.

Example

We had the factorial function as $fact \stackrel{\mathrm{def}}{=} fix_{nat \to nat}\ F$, where

$$F \stackrel{\mathrm{def}}{=} \lambda f : nat \to nat . \lambda y : nat.\ \texttt{if}\ \ Eq?\ y\ 0\ \ \texttt{then}\ \ 1\ \ \texttt{else}\ \ y * f(y-1).$$

From the definition of fix in CPO we see that the meaning of factorial is the least upper bound $\bigvee\{\overline{F}^n(\bot) \mid n \geq 0\}$, where $\overline{F} = \mathcal{A}_{\mathrm{PCF}}[\![F]\!]$. Let us work out the first cases of $\overline{F}^n(\bot)$. We discard writing $\mathcal{A}_{\mathrm{PCF}}[\![]\!]$ and use lambda terms as names for elements of the corresponding domains.

$$\begin{array}{rcl}
\overline{F}^0(\bot) & = & \bot_{nat \to nat} \\
\overline{F}^1(\bot) & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * (\overline{F}^0(\bot))(y-1) \\
 & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * (\bot_{nat \to nat})(y-1) \\
 & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ \bot_{nat} \\
\overline{F}^2(\bot) & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * (\overline{F}^1(\bot))(y-1) \\
 & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * \\
 & & \qquad (\lambda x : nat.\ \texttt{if}\ Eq?\ x\ 0\ \texttt{then}\ 1\ \texttt{else}\ \bot_{nat})(y-1) \\
 & = & \lambda y : nat.\ \texttt{if}\ Eq?\ y\ 0\ \texttt{then}\ 1\ \texttt{else}\ y * \\
 & & \qquad (\texttt{if}\ Eq?\ (y-1)\ 0\ \texttt{then}\ 1\ \texttt{else}\ \bot_{nat})
\end{array}$$

We can see that $\overline{F}^n(\bot)$ computes $y!$ if $0 \leq y < n$ and has value $\bot_{nat}$ otherwise. This means that $\overline{F}^n(\bot)$ is defined for $0 \leq y < n$ and undefined otherwise. For an argument $y \neq \bot$ there are two possible values for $\overline{F}^n(\bot)\ y$. If $n \leq y$ then the value is $\bot$, for $n > y$ the value is $y!$. By Lemma 4, the least upper bound of $\{\overline{F}^n(\bot) \mid n \geq 0\}$ is the function mapping any $y \in \mathcal{N}_\bot$

to the least upper bound of the set $\{\overline{F}^n(\bot)\, y \mid n \geq 0\}$. So, $\mathcal{A}_{\mathrm{PCF}}[\![fact]\!]$ is the function from $\mathcal{N}_\bot$ to $\mathcal{N}_\bot$, mapping $\bot$ to $\bot$, and any other $y \in \mathcal{N}$ to $y!$.

Reference

[Mit96] Mitchell J.C. Foundations for Programming Languages. MIT Press, 1996.