

Assignment 5

Recursive programs and recursive types

15-814: Types and Programming Languages
Jan Hoffmann & C.B. Aberlé

Due Friday, October 11, 2024
75 points

1 Least fixed-points

In lecture we noted that the fixed-point computation of **PCF** picks out the *most undefined* computation that satisfies the desired fixed-point equation. In this section we explore some key properties of partial orders that are used in the *denotational semantics* of **PCF**, a perspective that furnishes a mathematical account of the meaning of programs that makes the above perspective precise.

Complete partial orders. Recall that a partial order is a pair (P, \sqsubseteq) consisting of a set P together with a binary relation \sqsubseteq on P that is

reflexive: $x \sqsubseteq x$ for all $x \in P$ transitive: $x \sqsubseteq y$ and $y \sqsubseteq z \implies x \sqsubseteq z$

antisymmetric: $x \sqsubseteq y$ and $y \sqsubseteq x \implies x = y$

Fix a partial order (P, \sqsubseteq) . A chain in P is sequence of elements of P of the following form:

$$d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$$

More precisely, a chain is a function $d : \mathbb{N} \rightarrow P$ such that $d_i \sqsubseteq d_{i+1}$ for all i . The least upper bound of a chain d is an element $x \in P$ such that $d_i \sqsubseteq x$ for all i and for any other element $y \in P$ such that $d_i \sqsubseteq y$ for all i , we have $x \sqsubseteq y$. If every chain in P has a least upper bound, then P is an ω -complete partial order, written as ω CPO. In an ω CPO, we write $\bigsqcup_i d_i$ for the least upper bound of a chain d .

Continuous maps of ω CPOs. A function $f : P \rightarrow Q$ between partial orders (P, \sqsubseteq_P) , (Q, \sqsubseteq_Q) is monotone when it preserves the ordering relation, i.e.

$$x \sqsubseteq_P y \implies f(x) \sqsubseteq_Q f(y)$$

If (P, \sqsubseteq_P) and (Q, \sqsubseteq_Q) are both ω CPOs, then a monotone $f : P \rightarrow Q$ is ω -continuous if it preserves least upper bounds of chains, i.e.

$$f\left(\bigsqcup_i d_i\right) = \bigsqcup_i f(d_i)$$

for all chains $d : \mathbb{N} \rightarrow P$.

Domains. An ω CPO (P, \sqsubseteq) is a *domain* if there is an element $\perp \in P$ such that $\perp \sqsubseteq x$ for all $x \in P$.

In a domain model of **PCF**, we interpret types as domains and functions as continuous maps between domains. Intuitively, the ordering relation of domains reflects the *information*, or *definedness* order between programs, where the \perp element of each type represents a *totally undefined* program (e.g. an infinite loop with no effects).

In order to define this interpretation of **PCF**, we inductively define a partial order on semantic values of each type, in a manner that should by now be familiar to you from your prior work with logical relations. In the case of the function type $\sigma \rightarrow \tau$, assuming inductively that σ and τ have already been assigned domains $(\llbracket \sigma \rrbracket, \sqsubseteq_\sigma)$ and $(\llbracket \tau \rrbracket, \sqsubseteq_\tau)$, respectively, we define the interpretation of $\sigma \rightarrow \tau$ as the set $\llbracket \sigma \rrbracket \Rightarrow \llbracket \tau \rrbracket$ of continuous functions $\llbracket \sigma \rrbracket \rightarrow \llbracket \tau \rrbracket$, equipped with the partial order $\sqsubseteq_{\sigma \rightarrow \tau}$ such that $f \sqsubseteq_{\sigma \rightarrow \tau} g$ if for all $s \in \llbracket \sigma \rrbracket$, we have $f(s) \sqsubseteq_\tau g(s)$.

That the fixed-point computation of **PCF** chooses the *most undefined* function (i.e. the least function on the information order) satisfying the fixed-point equation is then corroborated by the fact that in a domain model of **PCF**, fixed-points are interpreted by the *least* fixed-point of the corresponding domain map.

Least fixed-points.

Consider a function $\text{fun } f(x : \sigma) : \tau \text{ is } e$ of type $\sigma \rightarrow \tau$ in call-by-value **PCF**. Assuming, inductively, that e is interpreted as a continuous map $\llbracket e \rrbracket : (\llbracket \sigma \rrbracket \Rightarrow \llbracket \tau \rrbracket) \rightarrow (\llbracket \sigma \rrbracket \Rightarrow \llbracket \tau \rrbracket)$, we then compute the interpretation of $\text{fun } f(x : \sigma) : \tau \text{ is } e$ as the least fixed point of $\llbracket e \rrbracket$.

Task 1 (10pts) Let (D, \sqsubseteq) be a domain and $f : D \rightarrow D$ a continuous map. Prove that the sequence

$$i \in \mathbb{N} \mapsto f^{(i)}(\perp)$$

where

$$f^{(0)}(\perp) = \perp \quad \text{and} \quad f^{(i+1)}(\perp) = f(f^{(i)}(\perp))$$

is a chain in \sqsubseteq .

Task 2 (25pts) Let (D, \sqsubseteq) and f be as in **task 1**. Prove that the least fixed-point of f is given by $\bigsqcup_i f^{(i)}(\perp)$.

We can extend this to an interpretation of **PCF** expressions as continuous maps between domains, per the following theorem (you will have a chance to develop this interpretation in detail yourself in one of the final projects for this course):

Theorem 1 There is a function $\llbracket - \rrbracket$ that assigns each **PCF** expression $e : \tau$ to an element $\llbracket e \rrbracket \in \llbracket \tau \rrbracket$, where $\llbracket \sigma \rightarrow \tau \rrbracket = \llbracket \sigma \rrbracket \Rightarrow \llbracket \tau \rrbracket$, as above, and $\llbracket \text{nat} \rrbracket = \{\perp\} \cup \mathbb{N}$ ordered by the flat ordering \sqsubseteq_{nat} such that

$$x \sqsubseteq_{\text{nat}} y \iff x = \perp \text{ or } x = y$$

such that every nonterminating expression $e : \text{nat}$ is mapped to $\llbracket e \rrbracket = \perp$, and for every terminating expression $e : \text{nat}$, we have $\llbracket e \rrbracket = n$ if and only if $e \mapsto^* \bar{n}$.

Task 3 (5pts) Using the above theorem, show that **PCF** cannot distinguish terminating and nonterminating computations of type nat , i.e. there is no expression $f : \text{nat} \rightarrow \text{nat}$ in **PCF** such that $f(e) \mapsto^* 0$ if e is nonterminating, and $f(e) \mapsto^* 1$ otherwise.

2 Strictly Positive Recursive Types & Initial Algebras

In this section, we work in System FPC with call-by-value dynamics.

Let t be a type variable. The set of *strictly positive* type expressions in t , written $\mathbb{P}[t]$ is defined to be the least set of types closed under the following rules:

$$\frac{}{t \in \mathbb{P}[t]} \quad \frac{}{0 \in \mathbb{P}[t]} \quad \frac{}{1 \in \mathbb{P}[t]} \quad \frac{\tau_1 \in \mathbb{P}[t] \quad \tau_2 \in \mathbb{P}[t]}{\tau_1 + \tau_2 \in \mathbb{P}[t]} \quad \frac{\tau_1 \in \mathbb{P}[t] \quad \tau_2 \in \mathbb{P}[t]}{\tau_1 \times \tau_2 \in \mathbb{P}[t]}$$

Recursive types of the form $\text{rec}(t.\tau)$ where $\tau \in \mathbb{P}[t]$ are called *strictly positive recursive types*, and form an especially well-behaved class of recursive types containing most familiar examples from programming. For instance, we have already seen in lectures how the natural numbers can be represented as the type $\text{rec}(t.1 + t)$, and the expression $1 + t$ is indeed strictly positive in t .

In this section, we will see how strictly positive recursive types can be seen as arising from a similar kind of *least fixed point* as in the previous section. In order to do so, we generalize the notion of *least fixed point* to be defined only *up to isomorphism*, leading to the notion of an *initial algebra*.

Task 4 (15pts) Show by induction on $\tau \in \mathbb{P}[t]$ that if there is an FPC expression $e_f : \sigma_1 \rightarrow \sigma_2$ for closed types σ_1, σ_2 , then there is an expression $\tau[e_f] : [\sigma_1/t]\tau \rightarrow [\sigma_2/t]\tau$.

A τ -algebra, for $\tau \in \mathbb{P}[t]$, is a pair (σ, s) consisting of a closed type σ together with an expression s of type $[\sigma/t]\tau \rightarrow \sigma$.

Example For $\tau = t \times t$, one possible τ -algebra is the pair $(\text{nat}, \text{plus})$, where $\text{plus} : \text{nat} \times \text{nat} \rightarrow \text{nat}$ is the usual addition of natural numbers. A different τ -algebra with the same carrier type is $(\text{nat}, \text{times})$, where $\text{times} : \text{nat} \times \text{nat} \rightarrow \text{nat}$ is the usual multiplication of natural numbers.

Task 5 (5pts) Show that for all $\tau \in \mathbb{P}[t]$, the type $\text{rec}(t.\tau)$ carries the structure of a τ -algebra, i.e. there is a function $\text{alg}_\tau : ([\text{rec}(t.\tau)/t]\tau \rightarrow \text{rec}(t.\tau))$. (Hint: you do not need to use induction.)

Given τ -algebras (σ_1, s_1) , (σ_2, s_2) an expression e_f of type $\sigma_1 \rightarrow \sigma_2$ is a *homomorphism* from (σ_1, s_1) to (σ_2, s_2) if for all values $v : [\sigma_1/t]\tau$,

$$e_f(s_1(v)) = s_2(\tau[e_f](v))$$

Example For $\tau = t \times t$ as in the above example, the function $\text{exp} : \text{nat} \rightarrow \text{nat}$ where $\text{exp}(\bar{n}) = 2^{\bar{n}}$, is a homomorphism from the τ -algebra $(\text{nat}, \text{plus})$ to $(\text{nat}, \text{times})$, since for all $m, n \in \mathbb{N}$

$$\text{exp}(\text{plus}(\bar{m}, \bar{n})) = \text{times}(\text{exp}(\bar{m}), \text{exp}(\bar{n}))$$

A τ -algebra (σ, s) is called *initial* if for every τ -algebra (σ', s') there is a unique homomorphism $f_{s'} : \sigma \rightarrow \sigma'$ from (σ, s) to (σ', s')

Task 6 (10pts) Show that for all $\tau \in \mathbb{P}[t]$, the algebra $(\text{rec}(t.\tau), \text{alg}_\tau)$ defined in **task 5** is initial. You do not need to show that the homomorphisms you construct are unique, but you should prove that they are homomorphisms (hint: use recursion; you still do not need induction.)

Task 7 (5pts) Show that initial algebras for a given $\tau \in \mathbb{P}[t]$ are unique up to isomorphism, i.e. if (σ_1, s_1) and (σ_2, s_2) are both initial τ -algebras, then $\sigma_1 \cong \sigma_2$ (you may use the informal definition of isomorphism given in lecture).

Compare the notion of initial algebras we have developed here with the notion of least fixed points given previously. In what way can this be regarded as a notion of least fixed point "up to isomorphism"?