Modular Analysis

Joonhyup Lee

April 26, 2025

1 Denotational Semantics for Untyped λ-Calculus

1.1 Semantic Domains

$$
\begin{array}{rcll}
\text{Program variable} & x & \in & \mathsf{Var} \\
\text{Expression} & e & \in & \mathsf{Expr} ::= x \mid \lambda x.e \mid e\ e \\
\text{Shadow} & S & \in & \mathsf{Shadow} ::= \mathsf{Rd}(x) \mid \mathsf{Ap}(S, v) \\
\text{Environment} & \sigma & \in & \mathsf{Env} ::= \mathsf{Init} \mid (x, v) :: \sigma \\
\text{Value} & v & \in & \mathsf{Val} ::= S \mid \langle x, t, \sigma \rangle \\
\text{Trace (Coinductive)} & t & \in & \mathsf{Trace} ::= \bot \mid \sigma \rightarrow t \mid v
\end{array}
$$

1.2 Linking

Linking is defined by mixed induction-coinduction. $\sigma_0 \propto t \mathbin{⨾} k$ appeals to t coinductively, while for other domains the arguments are recursed upon inductively.

$$\boxed{\cdot \propto \cdot \mathbin{⨾} \cdot \in \mathsf{Env} \rightarrow \mathsf{Shadow} \rightarrow (\mathsf{Val} \rightarrow \mathsf{Trace}) \rightarrow \mathsf{Trace}}$$

$$
\begin{aligned}
\sigma_0 \propto \mathsf{Rd}(x) \mathbin{⨾} k &\triangleq \mathsf{rd}(\sigma_0, x; k) \\
\sigma_0 \propto \mathsf{Ap}(S, v) \mathbin{⨾} k &\triangleq \sigma_0 \propto S \mathbin{⨾} \lambda f.\sigma_0 \propto v \mathbin{⨾} \lambda a.\mathsf{ap}(f, a; k)
\end{aligned}
$$

$$\boxed{\cdot \propto \cdot \mathbin{⨾} \cdot \in \mathsf{Env} \rightarrow \mathsf{Env} \rightarrow (\mathsf{Env} \rightarrow \mathsf{Trace}) \rightarrow \mathsf{Trace}}$$

$$
\begin{aligned}
\sigma_0 \propto \mathsf{Init} \mathbin{⨾} k &\triangleq k(\sigma_0) \\
\sigma_0 \propto (x, v) :: \sigma \mathbin{⨾} k &\triangleq \sigma_0 \propto v \mathbin{⨾} \lambda v'.\sigma_0 \propto \sigma \mathbin{⨾} \lambda \sigma'.k((x, v') :: \sigma')
\end{aligned}
$$

$$\boxed{\cdot \propto \cdot \mathbin{⨾} \cdot \in \mathsf{Env} \rightarrow \mathsf{Val} \rightarrow (\mathsf{Val} \rightarrow \mathsf{Trace}) \rightarrow \mathsf{Trace}}$$

$$
\sigma_0 \propto \langle x, t, \sigma \rangle \mathbin{⨾} k \triangleq \sigma_0 \propto \sigma \mathbin{⨾} \lambda \sigma'.k(\langle x, t, \sigma' \rangle)
$$

$$\boxed{\cdot \infty \cdot \fatsemi \cdot \in \mathsf{Env} \to \mathsf{Trace} \to (\mathsf{Val} \to \mathsf{Trace}) \to \mathsf{Trace}}$$

$$\begin{aligned} \sigma_0 \infty \quad\quad \bot \fatsemi k &\triangleq \bot \\ \sigma_0 \infty \quad \sigma \to t \fatsemi k &\triangleq \sigma_0 \infty \sigma \fatsemi \lambda \sigma'.\sigma' \to \sigma_0 \infty t \fatsemi k \end{aligned}$$

$\mathsf{rd}(\sigma, x; k)$ and $\mathsf{ap}(f, a; k)$ are defined by

$$\mathsf{rd}(\sigma, x; k) \triangleq \begin{cases} \bot & \text{when } \sigma(x) = \bot \\ k(v) & \text{when } \sigma(x) = v \end{cases}$$

$$\mathsf{ap}(f, a; k) \triangleq \begin{cases} (x, a) :: \sigma \infty t \fatsemi k & \text{when } f = \langle x, t, \sigma \rangle \\ k(\mathsf{Ap}(S, a)) & \text{when } f = S \end{cases}$$

1.3 Denotational Semantics

Denotational semantics is defined by

$$\boxed{[\![e]\!] \in (\mathsf{Val} \to \mathsf{Trace}) \to \mathsf{Trace}}$$

$$\begin{aligned} [\![x]\!]k &\triangleq \mathsf{Init} \to k(\mathsf{Rd}(x)) \\ [\![\lambda x.e]\!]k &\triangleq \mathsf{Init} \to k(\langle x, [\![e]\!]\mathsf{Ret}, \mathsf{Init} \rangle) \\ [\![e_1\ e_2]\!]k &\triangleq \mathsf{Init} \to [\![e_1]\!]\lambda f.[\![e_2]\!]\lambda a.\mathsf{ap}(f, a; k) \end{aligned}$$

Why do we need the $\mathsf{Init} \to$ prefix?

$$(\lambda \omega.(\lambda x.\lambda y.y)(\omega\ \omega))(\lambda x.x\ x)$$

Preserve non-termination in call-by-value semantics

1.4 Proving Equivalence with Standard Semantics

Environment	$\underline{\sigma}$	$\in$	$\underline{\mathsf{Env}} \triangleq \mathsf{Var} \xrightarrow{\text{fin}} \underline{\mathsf{Val}}$
Value	$\underline{v}$	$\in$	$\underline{\mathsf{Val}} \triangleq \mathsf{Var} \times \mathsf{Expr} \times \underline{\mathsf{Env}}$

We can "lower" a shadow-free environment/value/trace into a function from some initial environment $\underline{\sigma_0} \in \underline{\mathsf{Env}}$ to its output environment/value/value. For example,

$$\boxed{\downarrow \sigma \in \underline{\mathsf{Env}} \rightharpoonup \underline{\mathsf{Env}}}$$

$$\begin{aligned} \downarrow \mathsf{Init} &\triangleq \lambda \underline{\sigma_0}.\underline{\sigma_0} \\ \downarrow (x, v) :: \sigma &\triangleq (\downarrow \sigma)[x \mapsto \downarrow v] \end{aligned}$$

Note that the function is a partial function; it might return bottom if some intermediate output is bottom. Such a case might occur when either there is an infinite computation or an unresolved computation (a shadow).

We want to prove that $\underline{\sigma} \vdash e \Downarrow \underline{v}$ if and only if there exists a σ such that $(\downarrow \sigma)[] = \underline{\sigma}$ and $(\downarrow (\sigma \infty [\![e]\!]\mathsf{Ret} \fatsemi \mathsf{Ret}))[] = \underline{v}$.