Modular Analysis

Joonhyup Lee

March 22, 2024

1 Syntax and Semantics

1.1 Abstract Syntax

$$
\begin{array}{rccll}
\text{Identifiers} & x & \in & \text{Var} & \\
\text{Expression} & e & \to & x \mid \lambda x.e \mid e\ e & \lambda\text{-calculus} \\
 & & \mid & e \rtimes e & \text{linked expression} \\
 & & \mid & \varepsilon & \text{empty module} \\
 & & \mid & x = e\ ;\ e & \text{(recursive) binding}
\end{array}
$$

Figure 1: Abstract syntax of the language.

1.2 Operational Semantics

$$
\begin{array}{rccll}
\text{Environment} & \sigma & \in & \text{Env} & \\
\text{Location} & \ell & \in & \text{Loc} & \\
\text{de Bruijn Index} & n & \in & \mathbb{N} & \\
\text{Value} & v & \in & \text{Val} \triangleq \text{Env} + \text{Var} \times \text{Expr} \times \text{Env} & \\
\text{Weak Value} & w & \in & \text{WVal} \triangleq \text{Val} + \underline{\text{Val}} & \\
\text{Environment} & \sigma & \to & \bullet & \text{empty stack} \\
 & & \mid & (x, w) :: \sigma & \text{weak value binding} \\
 & & \mid & (x, \ell) :: \sigma & \text{free location binding} \\
 & & \mid & (x, n) :: \sigma & \text{bound location binding} \\
\text{Value} & v & \to & \sigma & \text{exported environment} \\
 & & \mid & \langle \lambda x.e, \sigma \rangle & \text{closure} \\
\text{Weak Value} & w & \to & v & \text{value} \\
 & & \mid & \mu.v & \text{recursive value}
\end{array}
$$

Figure 2: Definition of the semantic domains.

$$\boxed{\sigma \vdash e \Downarrow v}$$

$$
\dfrac{\sigma(x) = v}{\sigma \vdash x \Downarrow v}\ \textsc{Id}
\qquad
\dfrac{\sigma(x) = \mu.v}{\sigma \vdash x \Downarrow \{0 \to \mu.v\}v}\ \textsc{RecId}
\qquad
\dfrac{}{\sigma \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle}\ \textsc{Fn}
\qquad
\dfrac{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad \sigma \vdash e_2 \Downarrow v_2 \quad (x, v_2) :: \sigma_1 \vdash e \Downarrow v}{\sigma \vdash e_1\ e_2 \Downarrow v}\ \textsc{App}
$$

$$
\dfrac{\sigma \vdash e_1 \Downarrow \sigma_1 \quad \sigma_1 \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \rtimes e_2 \Downarrow v}\ \textsc{Link}
\qquad
\dfrac{}{\sigma \vdash \varepsilon \Downarrow \bullet}\ \textsc{Empty}
\qquad
\dfrac{\ell \notin \text{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma \vdash e_1 \Downarrow \sigma_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma_2}\ \textsc{Bind}
$$

Figure 3: The big-step operational semantics.

We use the locally nameless representation, and enforce that all values be *locally closed*. As a consequence, the big-step operational semantics will be *deterministic*, no matter what ℓ is chosen in the Bind rule.

1.3 Reconciling with Conventional Backpatching

$$
\begin{array}{rccll}
\text{Environment} & \sigma & \in & \text{MEnv} \triangleq \text{Var} \xrightarrow{\text{fin}} \text{Loc} & \\
\text{Memory} & m & \in & \text{Mem} \triangleq \text{Loc} \xrightarrow{\text{fin}} \text{MVal} & \\
\text{Allocated set} & L & \subseteq & \text{Loc} & \\
\text{Value} & v & \in & \text{MVal} \triangleq \text{MEnv} + \text{Var} \times \text{Expr} \times \text{MEnv} & \\
\text{Environment} & \sigma & \to & \bullet & \text{empty stack} \\
& & \mid & (x, \ell) :: \sigma & \text{location binding} \\
\text{Value} & v & \to & \sigma & \text{exported environment} \\
& & \mid & \langle \lambda x.e, \sigma \rangle & \text{closure}
\end{array}
$$

Figure 4: Definition of the semantic domains with memory.

$$\boxed{\sigma, m, L \vdash e \Downarrow v, m', L'}$$

$$
\frac{\sigma(x) = \ell \qquad m(\ell) = v}{\sigma, m, L \vdash x \Downarrow v, m, L}\ \textsc{Id}
\qquad
\frac{}{\sigma, m, L \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle, m, L}\ \textsc{Fn}
$$

$$
\frac{\begin{array}{c}\sigma, m, L \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle, m_1, L_1 \qquad \sigma, m_1, L_1 \vdash e_2 \Downarrow v_2, m_2, L_2 \qquad \ell \notin \text{dom}(m_2) \cup L_2 \\ (x, \ell) :: \sigma_1, m_2[\ell \mapsto v_2], L_2 \vdash e \Downarrow v, m', L'\end{array}}{\sigma, m, L \vdash e_1\ e_2 \Downarrow v, m', L'}\ \textsc{App}
$$

$$
\frac{\sigma, m, L \vdash e_1 \Downarrow \sigma_1, m_1, L_1 \qquad \sigma_1, m_1, L_1 \vdash e_2 \Downarrow v, m', L'}{\sigma, m, L \vdash e_1 \rtimes e_2 \Downarrow v, m', L'}\ \textsc{Link}
\qquad
\frac{}{\sigma, m, L \vdash \varepsilon \Downarrow \bullet, m, L}\ \textsc{Empty}
$$

$$
\frac{\begin{array}{c}\ell \notin \text{dom}(m) \cup L \qquad (x, \ell) :: \sigma, m, L \cup \{\ell\} \vdash e_1 \Downarrow v_1, m_1, L_1 \\ (x, \ell) :: \sigma, m_1[\ell \mapsto v_1], L_1 \vdash e_2 \Downarrow \sigma_2, m', L'\end{array}}{\sigma, m, L \vdash x = e_1; e_2 \Downarrow (x, \ell) :: \sigma_2, m', L'}\ \textsc{Bind}
$$

Figure 5: The big-step operational semantics with memory.

$$\boxed{w \sim_f v, m}$$

$$
\frac{}{\bullet \sim_f \bullet}\ \textsc{Eq-Nil}
\qquad
\frac{\ell \notin \text{dom}(f) \qquad \ell \notin \text{dom}(m) \qquad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell) :: \sigma'}\ \textsc{Eq-ConsFree}
\qquad
\frac{f(\ell) = \ell' \qquad \ell' \in \text{dom}(m) \qquad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell') :: \sigma'}\ \textsc{Eq-ConsBound}
$$

$$
\frac{m(\ell') = v' \qquad w \sim_f v' \qquad \sigma \sim_f \sigma'}{(x, w) :: \sigma \sim_f (x, \ell') :: \sigma'}\ \textsc{Eq-ConsWVal}
\qquad
\frac{\sigma \sim_f \sigma'}{\langle \lambda x.e, \sigma \rangle \sim_f \langle \lambda x.e, \sigma' \rangle}\ \textsc{Eq-Clos}
\qquad
\frac{L \text{ finite} \qquad m(\ell') = v' \qquad \forall \ell \notin L,\ \{0 \to \ell\} v \sim_{f[\ell \mapsto \ell']} v'}{\mu.v \sim_f v'}\ \textsc{Eq-Rec}
$$

Figure 6: The equivalence relation between weak values in the original semantics and values in the semantics with memory. $f \in \text{Loc} \xrightarrow{\text{fin}} \text{Loc}$ tells what the free locations in w that were *opened* should be mapped to in memory.

The semantics in Figure 3 makes sense due to similarity with a conventional backpatching semantics as presented in Figure 5. We have defined a relation $\sim$ that satisfies:

$$\sim \subseteq \text{WVal} \times (\text{MVal} \times \text{Mem} \times \mathcal{P}(\text{Loc})) \qquad \bullet \sim (\bullet, \emptyset, \emptyset)$$

and the following theorem:

Theorem 1.1 (Equivalence of semantics)**.** For all $\sigma \in \text{Env}, \sigma' \in \text{MEnv} \times \text{Mem} \times \mathcal{P}(\text{Loc}), v \in \text{Val}, v' \in \text{MVal} \times \text{Mem} \times \mathcal{P}(\text{Loc})$, we have:

$$\sigma \sim \sigma' \text{ and } \sigma \vdash e \Downarrow v \Rightarrow \exists v' : v \sim v' \text{ and } \sigma' \vdash e \Downarrow v'$$
$$\sigma \sim \sigma' \text{ and } \sigma' \vdash e \Downarrow v' \Rightarrow \exists v : v \sim v' \text{ and } \sigma \vdash e \Downarrow v$$

The actual definition for $\sim$ is given in Figure 6.

The proof of Theorem 1.1 uses some useful lemmas, such as:

Lemma 1.1 (Free locations not in f are free in memory)**.**

$$w \sim_f v', m \Rightarrow m|_{\text{FLoc}(w)-\text{dom}(f)} = \bot$$

Lemma 1.2 (Equivalence is preserved by extension of memory)**.**

$$w \sim_f v', m \text{ and } m \sqsubseteq m' \text{ and } m'|_{\text{FLoc}(w)-\text{dom}(f)} = \bot \Rightarrow w \sim_f v', m$$

Lemma 1.3 (Equivalence only cares about f on free locations)**.**

$$w \sim_f v', m \text{ and } f|_{\text{FLoc}(w)} = f|_{\text{FLoc}(w)} \Rightarrow w \sim_{f'} v', m$$

Lemma 1.4 (Extending equivalence on free locations)**.**

$$w \sim_f v', m \text{ and } \ell \notin \text{dom}(f) \text{ and } \ell \notin \text{dom}(m) \Rightarrow \forall u', w \sim_{f[\ell \mapsto \ell]} v', m[\ell \mapsto u']$$

Lemma 1.5 (Substitution of values)**.**

$$w \sim_f v', m \text{ and } f(\ell) = \ell' \text{ and } m(\ell') = u' \text{ and } u \sim_{f-\ell} u', m \Rightarrow w[u/\ell] \sim_{f-\ell} v', m$$

Lemma 1.6 (Substitution of locations)**.**

$$w \sim_f v', m \text{ and } \ell \in \text{dom}(f) \text{ and } \nu \notin \text{FLoc}(w) \Rightarrow w[\nu/\ell] \sim_{f \circ (\nu \leftrightarrow \ell)} v', m$$

2 Generating and Resolving Events

Now we formulate the semantics for generating events.

Event	E	$\rightarrow$	Init	initial environment
		$\mid$	$\mathsf{Read}(E, x)$	read event
		$\mid$	$\mathsf{Call}(E, v)$	call event
Environment	σ	$\rightarrow$	$\cdots$	
		$\mid$	$[E]$	answer to an event
Value	v	$\rightarrow$	$\cdots$	
		$\mid$	E	answer to an event

Figure 7: Definition of the semantic domains with events. All other semantic domains are equal to Figure 2.

We extend how to read weak values given an environment.

$$\bullet(x) \triangleq \bot \qquad ((x', \ell) :: \sigma)(x) \triangleq (x = x' ? \ell : \sigma(x))$$
$$[E](x) \triangleq \mathsf{Read}(E, x) \qquad ((x', w) :: \sigma)(x) \triangleq (x = x' ? w : \sigma(x))$$

Then we need to add only one rule to the semantics in Figure 3 for the semantics to incorporate events.

$$\textsc{AppEvent}\ \frac{\sigma \vdash e_1 \Downarrow E \qquad \sigma \vdash e_2 \Downarrow v}{\sigma \vdash e_1\ e_2 \Downarrow \mathsf{Call}(E, v)}$$

Now we need to formulate the *concrete linking* rules. The concrete linking rule $\sigma_0 \propto w$, given an answer σ_0 to the Init event, resolves all events within w to obtain a set of final results.

Concrete linking makes sense because of the following theorem. First define:

$$\text{eval}(e, \sigma) \triangleq \{v | \sigma \vdash e \Downarrow v\} \qquad \text{eval}(e, \Sigma) \triangleq \bigcup_{\sigma \in \Sigma} \text{eval}(e, \sigma) \qquad \sigma_0 \propto W \triangleq \bigcup_{w \in W} (\sigma_0 \propto w)$$

Then the following holds:

$$\boxed{\infty \in \text{Env} \to \text{Event} \to \mathcal{P}(\text{Val})}$$

$$\begin{aligned}
\sigma_0 \infty \mathsf{Init} &\triangleq \{\sigma_0\} \\
\sigma_0 \infty \mathsf{Read}(E, x) &\triangleq \{v_+ | \sigma_+ \in \sigma_0 \infty E \land \sigma_+(x) = v_+\} \\
&\cup \{\{0 \to \mu.v_+\}v_+ | \sigma_+ \in \sigma_0 \infty E \land \sigma_+(x) = \mu.v_+\} \\
\sigma_0 \infty \mathsf{Call}(E, v) &\triangleq \{v'_+ | \langle \lambda x.e, \sigma_+\rangle \in \sigma_0 \infty E \land v_+ \in \sigma_0 \infty v \land (x, v_+) :: \sigma_+ \vdash e \Downarrow v'_+\} \\
&\cup \{\mathsf{Call}(E_+, v_+) | E_+ \in \sigma_0 \infty E \land v_+ \in \sigma_0 \infty v\}
\end{aligned}$$

$$\boxed{\infty \in \text{Env} \to \text{Env} \to \mathcal{P}(\text{Env})}$$

$$\begin{aligned}
\sigma_0 \infty \bullet &\triangleq \{\bullet\} \\
\sigma_0 \infty (x, \ell) :: \sigma &\triangleq \{(x, \ell) :: \sigma_+ | \sigma_+ \in \sigma_0 \infty \sigma\} \\
\sigma_0 \infty (x, w) :: \sigma &\triangleq \{(x, w_+) :: \sigma_+ | w_+ \in \sigma_0 \infty w \land \sigma_+ \in \sigma_0 \infty \sigma\} \\
\sigma_0 \infty [E] &\triangleq \{\sigma_+ | \sigma_+ \in \sigma_0 \infty E\} \cup \{[E_+] | E_+ \in \sigma_0 \infty E\}
\end{aligned}$$

$$\boxed{\infty \in \text{Env} \to \text{Val} \to \mathcal{P}(\text{Val})}$$

$$\sigma_0 \infty \langle \lambda x.e, \sigma\rangle \triangleq \{\langle \lambda x.e, \sigma_+\rangle | \sigma_+ \in \sigma_0 \infty \sigma\}$$

$$\boxed{\infty \in \text{Env} \to \text{WVal} \to \mathcal{P}(\text{WVal})}$$

$$\sigma_0 \infty \mu.v \triangleq \{\mu.\{0 \leftarrow \ell\}v_+ | \ell \notin \text{FLoc}(v) \cup \text{FLoc}(\sigma_0) \land v_+ \in \sigma_0 \infty \{0 \to \ell\}v\}$$

Figure 8: Definition for concrete linking.

Theorem 2.1 (Soundness of concrete linking)**.** Given $e \in \text{Expr}, \sigma \in \text{Env}, v \in \text{Val}$,

$$\forall \sigma_0 \in \text{Env} : \text{eval}(e, \sigma_0 \infty \sigma) \subseteq \sigma_0 \infty \text{eval}(e, \sigma)$$

The proof of Theorem 2.1 uses some useful lemmas, such as:

Lemma 2.1 (Linking distributes under substitution)**.** Let σ_0 be the external environment that is linked with locally closed weak values w and u. For all $\ell \notin \text{FLoc}(\sigma_0)$, we have:

$$\forall w_+, u_+ : w_+ \in \sigma_0 \infty w \land u_+ \in \sigma_0 \infty u \Rightarrow \{u_+ \leftarrow \ell\}w_+ \in \sigma_0 \infty \{u \leftarrow \ell\}w$$

Lemma 2.2 (Linking is compatible with reads)**.** Let σ_0 be the external environment that is linked with some environment σ. Let v be the value obtained from reading x from σ. Let unfold : WVal $\to$ Val be defined as:

$$\text{unfold}(\mu.v) \triangleq \{0 \to \mu.v\}v \qquad \text{unfold}(v) \triangleq v$$

Then for all $\sigma_+ \in \sigma_0 \infty \sigma$, we have:

$$\exists w_+ \in \text{WVal} : \sigma_+(x) = w_+ \land \text{unfold}(w_+) \in \sigma_0 \infty v$$

3 CFA

$$
\begin{array}{rccl}
\text{Program point} & p & \in & \mathbb{P} \triangleq \{\text{finite set of program points}\} \\
\text{Labelled expression} & e & \in & \text{PExpr} \\
\text{Program graph} & g & \in & \mathbb{G} \triangleq \mathbb{P} \to \text{PExpr} \\
\text{Proof tree} & t & \in & \mathbb{T} \triangleq \mathbb{P} \to \mathcal{P}(\text{PEnv} + \text{PEnv} \times \text{PVal}) \\
 & e & \to & x \mid \lambda x.p \mid p\ p \mid p \rtimes p \mid \varepsilon \mid x = p; p
\end{array}
$$

$$\boxed{\text{Step} : \mathbb{T} \to \mathbb{T}}$$

$$\text{Step}(t) \triangleq \bigcup_{p \in \mathbb{P}} \text{step}(t, p)$$

$$\boxed{\text{step} : (\mathbb{T} \times \mathbb{P}) \to \mathbb{T}}$$

$$
\begin{array}{rll}
\text{step}(t,p) \triangleq & [p \mapsto \{(\sigma, v) | \sigma \in t(p) \text{ and } \sigma(x) = v\}] & \text{when } g(p) = x \\
& \cup [p \mapsto \{(\sigma, \{0 \to \mu.v\}v) | \sigma \in t(p) \text{ and } \sigma(v) = \mu.v\}] & \\
\text{step}(t,p) \triangleq & [p \mapsto \{(\sigma, \langle \lambda x.p', \sigma \rangle) | \sigma \in t(p)\}] & \text{when } g(p) = \lambda x.p' \\
\text{step}(t,p) \triangleq & [p_1 \mapsto \{\sigma | \sigma \in t(p)\}] & \text{when } g(p) = p_1\, p_2 \\
& \cup [p_2 \mapsto \{\sigma | \sigma \in t(p)\}] & \\
& \cup \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \langle \lambda x.p', \sigma_1 \rangle) \in t(p_1)} [p' \mapsto \{(x, v_2) :: \sigma_1 | (\sigma, v_2) \in t(p_2)\}] & \\
& \cup [p \mapsto \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \langle \lambda x.p', \sigma_1 \rangle) \in t(p_1)} \bigcup_{(\sigma, v_2) \in t(p_2)} \{(\sigma, v) | ((x, v_2) :: \sigma_1, v) \in t(p')\}] & \\
& \cup [p \mapsto \bigcup_{\sigma \in t(p)} \{(\sigma, \mathsf{Call}(E_1, v_2)) | (\sigma, E_1) \in t(p_1) \text{ and } (\sigma, v_2) \in t(p_2)\}] & \\
\text{step}(t,p) \triangleq & [p_1 \mapsto \{\sigma | \sigma \in t(p)\}] & \text{when } g(p) = p_1 \rtimes p_2 \\
& \cup [p_2 \mapsto \bigcup_{\sigma \in t(p)} \{\sigma_1 | (\sigma, \sigma_1) \in t(p_1)\}] & \\
& \cup [p \mapsto \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \sigma_1) \in t(p_1)} \{(\sigma, v_2) | (\sigma_1, v_2) \in t(p_2)\}] & \\
\text{step}(t,p) \triangleq & [p \mapsto \{(\sigma, \bullet) | \sigma \in t(p)\}] & \text{when } g(p) = \varepsilon \\
\text{step}(t,p) \triangleq & [p_1 \mapsto \{(x, \ell) :: \sigma | \sigma \in t(p) \text{ and } \ell \notin \text{FLoc}(\sigma)\}] & \text{when } g(p) = x = p_1; p_2 \\
& \cup [p_2 \mapsto \{(x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma | \sigma \in t(p) \text{ and } ((x, \ell) :: \sigma, v_1) \in t(p_1)\}] & \\
& \cup [p \mapsto \{(\sigma, (x, w_1) :: \sigma_2) | \sigma \in t(p) \text{ and } ((x, w_1) :: \sigma, \sigma_2) \in t(p_2)\}] &
\end{array}
$$

The proof tree t_0 computed by

$$t_0 \triangleq \text{lfp}(\lambda t.\text{Step}(t) \cup [p_0 \mapsto \{\sigma_0\}])$$

contains all derivations of the form $\sigma_0 \vdash p_0 \Downarrow v_0$ for some v_0. That is, (σ, v) is contained in $t_0(p)$ if and only if $\sigma \vdash p \Downarrow v$ must be contained in a valid derivation for the judgment $\sigma_0 \vdash p_0 \Downarrow v_0$.

$$
\begin{array}{rccl}
\text{Abstract event} & E^\# & \in & \text{Event}^\# \\
\text{Abstract environment} & \sigma^\# & \in & \text{Env}^\# \triangleq (\text{Var} \xrightarrow{\text{fin}} \mathcal{P}(\mathbb{P})) \times \mathcal{P}(\text{Event}^\#) \\
\text{Abstract closure} & \langle \lambda x.p, p' \rangle & \in & \text{Clos}^\# \triangleq \text{Var} \times \mathbb{P} \times \mathbb{P} \\
\text{Abstract value} & v^\# & \in & \text{Val}^\# \triangleq \text{Env}^\# \times \mathcal{P}(\text{Clos}^\#) \\
\text{Abstract proof tree} & t^\# & \in & \mathbb{T}^\# \triangleq \mathbb{P} \to \text{Env}^\# \times \text{Val}^\# \\
 & E^\# & \to & \mathsf{Init}^\# \mid \mathsf{Read}^\#(p, x) \mid \mathsf{Call}^\#(p, p)
\end{array}
$$