

Modular Analysis

Joonhyup Lee

April 20, 2024

1 Syntax and Semantics

1.1 Abstract Syntax

Identifiers	x	\in	Var	
Expression	e	\rightarrow	$x \mid \lambda x.e \mid e e$	λ -calculus
			$ e \rtimes e$	linked expression
			$ \varepsilon$	empty module
			$ x = e ; e$	(recursive) binding

Figure 1: Abstract syntax of the language.

1.2 Operational Semantics

Environment	σ	\in	$\text{Env} \triangleq \text{Var} \times (\text{Loc} + \text{WVal}) \times \text{Env} + \{\bullet\}$	
Location	ℓ	\in	Loc	
Value	v	\in	$\text{Val} \triangleq \text{Env} + \text{Var} \times \text{Expr} \times \text{Env}$	
Weak Value	w	\in	$\text{WVal} \triangleq \text{Val} + \underline{\text{Val}}$	
Environment	σ	\rightarrow	\bullet	empty stack
			$ (x, w) :: \sigma$	weak value binding
			$ (x, \ell) :: \sigma$	free location binding
Value	v	\rightarrow	σ	exported environment
			$ \langle \lambda x.e, \sigma \rangle$	closure
Weak Value	w	\rightarrow	v	value
			$ \mu\ell.v$	recursive value

Figure 2: Definition of the semantic domains.

$\frac{\text{ID}}{\sigma(x) = v} \quad \frac{\text{RECID}}{\sigma(x) = \mu\ell.v} \quad \frac{\text{FN}}{\sigma \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle}$	$\frac{\text{APP}}{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad \sigma \vdash e_2 \Downarrow v_2 \quad (x, v_2) :: \sigma_1 \vdash e \Downarrow v} \quad \sigma \vdash e_1 e_2 \Downarrow v$
$\frac{\text{LINK}}{\sigma \vdash e_1 \Downarrow \sigma_1 \quad \sigma_1 \vdash e_2 \Downarrow v} \quad \sigma \vdash e_1 \rtimes e_2 \Downarrow v$	$\frac{\text{EMPTY}}{\sigma \vdash \varepsilon \Downarrow \bullet} \quad \frac{\text{BIND}}{\ell \notin \text{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu\ell.v_1) :: \sigma \vdash e_2 \Downarrow \sigma_2} \quad \sigma \vdash x = e_1; e_2 \Downarrow (x, \mu\ell.v_1) :: \sigma_2$

Figure 3: The big-step operational semantics.

The big-step operational semantics is *deterministic* up to α -equivalence.

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

Figure 4: Definition of the semantic domains with memory.

$\sigma, m, L \vdash e \Downarrow v, m', L'$

$$\begin{array}{c}
\text{ID} \\
\frac{\sigma(x) = \ell \quad m(\ell) = v \quad \text{FN}}{\sigma, m, L \vdash x \Downarrow v, m, L} \quad \frac{}{\sigma, m, L \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle, m, L}
\end{array}$$

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

$$\begin{array}{c}
\text{LINK} \\
\frac{\sigma, m, L \vdash e_1 \Downarrow \sigma_1, m_1, L_1 \quad \sigma_1, m_1, L_1 \vdash e_2 \Downarrow v, m', L'}{\sigma, m, L \vdash e_1 \bowtie e_2 \Downarrow v, m', L'} \quad \frac{\text{EMPTY}}{\sigma, m, L \vdash \varepsilon \Downarrow \bullet, m, L}
\end{array}$$

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

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

$w \sim_f v, m$

$$\begin{array}{c}
\text{EQ-NIL} \quad \frac{}{\bullet \sim_f \bullet} \quad \text{EQ-CONSFREE} \quad \frac{\ell \notin \text{dom}(f) \quad \ell \notin \text{dom}(m) \quad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell) :: \sigma'} \quad \text{EQ-CONSBIND} \quad \frac{f(\ell) = \ell' \quad \ell' \in \text{dom}(m) \quad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell') :: \sigma'} \\
\\
\text{EQ-CONSWVAL} \quad \frac{m(\ell') = v' \quad w \sim_f v' \quad \sigma \sim_f \sigma'}{(x, w) :: \sigma \sim_f (x, \ell') :: \sigma'} \quad \text{EQ-CLOS} \quad \frac{\sigma \sim_f \sigma'}{\langle \lambda x.e, \sigma \rangle \sim_f \langle \lambda x.e, \sigma' \rangle} \quad \text{EQ-REC} \quad \frac{L \text{ finite} \quad m(\ell') = v' \quad \forall \nu \notin L, v[\nu/\ell] \sim_{f[\nu \mapsto \ell']} v'}{\mu \ell.v \sim_f v'}
\end{array}$$

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. m is omitted for brevity.

1.3 Reconciling with Conventional Backpatching

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:

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

The following theorem holds:

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:

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

The definition for $w \sim (\sigma, m, L)$ is:

$$w \sim_{\perp} (\sigma, m) \text{ and } \text{FLoc}(w) \subseteq L$$

where the definition for \sim_f 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).

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

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

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

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).

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

2 Generating and Resolving Events

Now we formulate the semantics for generating events.

Event	E	\rightarrow	Init	initial environment
		$ $	Read(E, x)	read event
		$ $	Call(E, v)	call event
Environment	σ	\rightarrow	\dots	
		$ $	$[E]$	answer to an event
Value	v	\rightarrow	\dots	
		$ $	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.

$$\begin{aligned} \bullet(x) &\triangleq \perp & ((x', \ell) :: \sigma)(x) &\triangleq (x = x'?\ell : \sigma(x)) \\ [E](x) &\triangleq \text{Read}(E, x) & ((x', w) :: \sigma)(x) &\triangleq (x = x'?w : \sigma(x)) \end{aligned}$$

Then we need to add only three rules to the semantics in Figure 3 for the semantics to incorporate events.

$$\begin{array}{c} \text{LINKEVENT} \\ \frac{\sigma \vdash e_1 \Downarrow E \quad [E] \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \times e_2 \Downarrow v} \end{array} \quad \begin{array}{c} \text{APPEVENT} \\ \frac{\sigma \vdash e_1 \Downarrow E \quad \sigma \vdash e_2 \Downarrow v}{\sigma \vdash e_1 e_2 \Downarrow \text{Call}(E, v)} \end{array} \quad \begin{array}{c} \text{BINDEVENT} \\ \frac{\ell \notin \text{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu\ell.v_1) :: \sigma \vdash e_2 \Downarrow E_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu\ell.v_1) :: [E_2]} \end{array}$$

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

$$\begin{array}{l} \boxed{\infty \in \text{Env} \rightarrow \text{Event} \rightarrow \mathcal{P}(\text{Val})} \\ \sigma_0 \times \text{Init} \triangleq \{\sigma_0\} \\ \sigma_0 \times \text{Read}(E, x) \triangleq \{v_+ | \sigma_+ \in \sigma_0 \times E, \sigma_+(x) = v_+\} \\ \quad \cup \{v_+ | \mu\ell.v_+/\ell | \sigma_+ \in \sigma_0 \times E, \sigma_+(x) = \mu\ell.v_+\} \\ \sigma_0 \times \text{Call}(E, v) \triangleq \{v'_+ | \langle \lambda x.e, \sigma_+ \rangle \in \sigma_0 \times E, v_+ \in \sigma_0 \times v, (x, v_+) :: \sigma_+ \vdash e \Downarrow v'_+\} \\ \quad \cup \{\text{Call}(E_+, v_+) | E_+ \in \sigma_0 \times E, v_+ \in \sigma_0 \times v\} \\ \boxed{\infty \in \text{Env} \rightarrow \text{Env} \rightarrow \mathcal{P}(\text{Env})} \\ \sigma_0 \times \bullet \triangleq \{\bullet\} \\ \sigma_0 \times (x, \ell) :: \sigma \triangleq \{(x, \ell) :: \sigma_+ | \sigma_+ \in \sigma_0 \times \sigma\} \\ \sigma_0 \times (x, w) :: \sigma \triangleq \{(x, w_+) :: \sigma_+ | w_+ \in \sigma_0 \times w, \sigma_+ \in \sigma_0 \times \sigma\} \\ \sigma_0 \times [E] \triangleq \{\sigma_+ | \sigma_+ \in \sigma_0 \times E\} \cup \{[E_+] | E_+ \in \sigma_0 \times E\} \\ \boxed{\infty \in \text{Env} \rightarrow \text{Val} \rightarrow \mathcal{P}(\text{Val})} \\ \sigma_0 \times \langle \lambda x.e, \sigma \rangle \triangleq \{\langle \lambda x.e, \sigma_+ \rangle | \sigma_+ \in \sigma_0 \times \sigma\} \\ \boxed{\infty \in \text{Env} \rightarrow \text{WVal} \rightarrow \mathcal{P}(\text{WVal})} \\ \sigma_0 \times \mu\ell.v \triangleq \{\mu\ell'.v_+ | \ell' \notin \text{FLoc}(v) \cup \text{FLoc}(\sigma_0), v_+ \in \sigma_0 \times v[\ell'/\ell]\} \end{array}$$

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

$$\text{eval}(e, \sigma) \triangleq \{v | \sigma \vdash e \Downarrow v\} \quad \text{eval}(e, \Sigma) \triangleq \bigcup_{\sigma \in \Sigma} \text{eval}(e, \sigma) \quad \Sigma_0 \times W \triangleq \bigcup_{\substack{\sigma_0 \in \Sigma_0 \\ w \in W}} (\sigma_0 \times w)$$

Then the following holds:

Theorem 2.1 (Advance). Given $e \in \text{Expr}$, $\Sigma_0, \Sigma \subseteq \text{Env}$,

$$\text{eval}(e, \Sigma_0 \times \Sigma) \subseteq \Sigma_0 \times \text{eval}(e, \Sigma)$$

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 weak values w and u . For all $\ell \notin \text{FLoc}(\sigma_0)$, we have:

$$\forall w_+, u_+ : w_+ \in \sigma_0 \times w \wedge u_+ \in \sigma_0 \times u \Rightarrow w_+[u_+/\ell] \in \sigma_0 \times w[u/\ell]$$

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

$$\text{unfold}(\mu\ell.v) \triangleq v[\mu\ell.v/\ell] \quad \text{unfold}(v) \triangleq v$$

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

$$\exists w_+ \in \text{WVal} : \sigma_+(x) = w_+ \wedge \text{unfold}(w_+) \in \sigma_0 \times \text{unfold}(w)$$

Now we can formulate modular analysis. A modular analysis consists of two requirements: an abstraction for the semantics with events and an abstraction for the semantic linking operator.

Theorem 2.2 (Modular analysis). Assume:

1. An abstract domain $\text{WVal}^\#$ that is concretized by a monotonic $\gamma \in \mathcal{P}(\text{WVal}) \rightarrow \text{WVal}^\#$
2. A sound $\text{eval}^\#$: $\Sigma_0 \subseteq \gamma(\sigma_0^\#) \Rightarrow \text{eval}(e, \Sigma_0) \subseteq \gamma(\text{eval}^\#(e, \sigma_0^\#))$
3. A sound $\times^\#$: $\Sigma_0 \subseteq \gamma(\sigma_0^\#)$ and $W \subseteq \gamma(w^\#) \Rightarrow \Sigma_0 \times W \subseteq \gamma(\sigma_0^\# \times^\# w^\#)$

then we have:

$$\Sigma_0 \subseteq \gamma(\sigma_0^\#) \text{ and } \Sigma \subseteq \gamma(\sigma^\#) \Rightarrow \text{eval}(e, \Sigma_0 \times \Sigma) \subseteq \gamma(\sigma_0^\# \times^\# \text{eval}^\#(e, \sigma^\#))$$

Corollary 2.1 (Modular analysis of linked program).

$$\Sigma_0 \subseteq \gamma(\sigma_0^\#) \text{ and } [\text{Init}] \in \gamma(\text{Init}^\#) \Rightarrow \text{eval}(e_1 \times e_2, \Sigma_0) \subseteq \gamma(\text{eval}^\#(e_1, \sigma_0^\#) \times^\# \text{eval}^\#(e_2, \text{Init}^\#))$$

3 CFA

3.1 Collecting semantics

Program point	p	\in	$\mathbb{P} \triangleq \{\text{finite set of program points}\}$
Labelled expression	pe	\in	$\mathbb{P} \times \text{Expr}$
Labelled location	ℓ^p	\in	$\mathbb{P} \times \text{Loc}$
Collecting semantics	t	\in	$\mathbb{T} \triangleq \mathbb{P} \rightarrow \mathcal{P}(\text{Env} + \text{Env} \times \text{Val})$
Labelled expression	pe	\rightarrow	$\{p : e\}$
Expression	e	\rightarrow	$x \mid \lambda x. pe \mid pe \ pe \mid pe \rtimes pe \mid \varepsilon \mid x = pe; pe$

Step : $\mathbb{T} \rightarrow \mathbb{T}$

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

step : $(\mathbb{T} \times \mathbb{P}) \rightarrow \mathbb{T}$

$$\text{step}(t, p) \triangleq [p \mapsto \{(\sigma, v) \mid \sigma \in t(p) \text{ and } \sigma(x) = v\}] \quad \text{when } \{p : x\}$$

$$\cup [p \mapsto \{(\sigma, v[\mu\ell^{p'} . v / \ell^{p'}]) \mid \sigma \in t(p) \text{ and } \sigma(x) = \mu\ell^{p'} . v\}]$$

$$\text{step}(t, p) \triangleq [p \mapsto \{(\sigma, \langle \lambda x. p', \sigma \rangle) \mid \sigma \in t(p)\}] \quad \text{when } \{p : \lambda x. p'\}$$

$$\text{step}(t, p) \triangleq [p_1 \mapsto \{\sigma \in \text{Env} \mid \sigma \in t(p)\}] \quad \text{when } \{p : p_1 \ p_2\}$$

$$\cup [p_2 \mapsto \{\sigma \in \text{Env} \mid \sigma \in t(p)\}]$$

$$\cup \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \langle \lambda x. p', \sigma_1 \rangle) \in t(p_1)} [p' \mapsto \{(x, v_2) :: \sigma_1 \mid (\sigma, v_2) \in t(p_2)\}]$$

$$\cup [p \mapsto \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \langle \lambda x. p', \sigma_1 \rangle) \in t(p_1)} \bigcup_{(\sigma, v_2) \in t(p_2)} \{(\sigma, v) \mid ((x, v_2) :: \sigma_1, v) \in t(p')\}]$$

$$\cup [p \mapsto \bigcup_{\sigma \in t(p)} \{(\sigma, \text{Call}(E_1, v_2)) \mid (\sigma, E_1) \in t(p_1) \text{ and } (\sigma, v_2) \in t(p_2)\}]$$

$$\text{step}(t, p) \triangleq [p_1 \mapsto \{\sigma \mid \sigma \in t(p)\}] \quad \text{when } \{p : p_1 \rtimes p_2\}$$

$$\cup [p_2 \mapsto \bigcup_{\sigma \in t(p)} \{\sigma_1 \mid (\sigma, \sigma_1) \in t(p_1)\}]$$

$$\cup [p \mapsto \bigcup_{\sigma \in t(p)} \bigcup_{(\sigma, \sigma_1) \in t(p_1)} \{(\sigma, v_2) \mid (\sigma_1, v_2) \in t(p_2)\}]$$

$$\text{step}(t, p) \triangleq [p \mapsto \{(\sigma, \bullet) \mid \sigma \in t(p)\}] \quad \text{when } \{p : \varepsilon\}$$

$$\text{step}(t, p) \triangleq [p_1 \mapsto \bigcup_{\sigma \in t(p)} \{(x, \ell^{p_1}) :: \sigma \mid \ell \notin \text{FLoc}(\sigma)\}] \quad \text{when } \{p : x = p_1; p_2\}$$

$$\cup [p_2 \mapsto \bigcup_{\sigma \in t(p)} \{(x, \mu\ell^{p_1} . v_1) :: \sigma \mid ((x, \ell^{p_1}) :: \sigma, v_1) \in t(p_1)\}]$$

$$\cup [p \mapsto \bigcup_{\sigma \in t(p)} \bigcup_{((x, \ell^{p_1}) :: \sigma, v_1) \in t(p_1)} \{(\sigma, (x, \mu\ell^{p_1} . v_1) :: \sigma_2) \mid ((x, \mu\ell^{p_1} . v_1) :: \sigma, \sigma_2) \in t(p_2)\}]$$

The collecting semantics $\llbracket p_0 \rrbracket \Sigma_0$ computed by

$$\llbracket p_0 \rrbracket \Sigma_0 \triangleq \text{lfp}(\lambda t. \text{Step}(t) \cup t_{\text{init}}) \quad \text{where } t_{\text{init}} = [p_0 \mapsto \Sigma_0]$$

contains all derivations of the form $\sigma_0 \vdash p_0 \Downarrow v_0$ for some $\sigma_0 \in \Sigma_0$ and v_0 . That is, (σ, v) is contained in $\llbracket p_0 \rrbracket \Sigma_0(p)$ if and only if $\sigma \vdash p \Downarrow v$ is contained in some derivation for the judgment $\sigma_0 \vdash p_0 \Downarrow v_0$.

3.2 Abstract semantics

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

$$\sigma \leq (\sigma^\#, t^\#)$$

$$\begin{array}{c} \text{CONC-NIL} \quad \text{CONC-ENIL} \quad \text{CONC-CONSLoc} \quad \text{CONC-CONSWVAL} \\ \hline \bullet \leq \sigma^\# \quad \frac{E \leq (\sigma^\#, \emptyset)}{[E] \leq \sigma^\#} \quad \frac{p \in \sigma^\#.1(x) \quad \sigma \leq \sigma^\#}{(x, \ell^p) :: \sigma \leq \sigma^\#} \quad \frac{p \in \sigma^\#.1(x) \quad w \leq t^\#(p).2 \quad \sigma \leq \sigma^\#}{(x, w) :: \sigma \leq \sigma^\#} \end{array}$$

$$w \leq (v^\#, t^\#)$$

$$\begin{array}{c} \text{CONC-CLOS} \quad \text{CONC-REC} \\ \hline \frac{\langle \lambda x.p, p' \rangle \in v^\#.2 \quad \sigma \leq t^\#(p').1}{\langle \lambda x.p, \sigma \rangle \leq v^\#} \quad \frac{v \leq t^\#(p).2 \quad v \leq v^\#}{\mu \ell^p.v \leq v^\#} \end{array}$$

$$\begin{array}{c} \text{CONC-INIT} \quad \text{CONC-READ} \quad \text{CONC-CALL} \\ \hline \frac{\text{Init}^\# \in v^\#.1.2}{\text{Init} \leq v^\#} \quad \frac{\text{Read}^\#(p, x) \in v^\#.1.2 \quad [E] \leq t^\#(p).1}{\text{Read}(E, x) \leq v^\#} \quad \frac{\text{Call}^\#(p_1, p_2) \in v^\#.1.2 \quad E \leq t^\#(p_1).2 \quad v \leq t^\#(p_2).2}{\text{Call}(E, v) \leq v^\#} \end{array}$$

Figure 8: The concretization relation between weak values and abstract values. $t^\#$ is omitted.

The concretization function γ that sends an element of $\mathbb{T}^\#$ to \mathbb{T} is defined as:

$$\gamma(t^\#) \triangleq \lambda p. \{ \sigma \mid \sigma \leq (t^\#(p).1, t^\#) \} \cup \{ (\sigma, v) \mid v \leq (t^\#(p).2, t^\#) \}$$

where \leq is the concretization relation that is inductively defined in Figure 8.

Now the abstract semantic function can be given.

$$\text{Step}^\# : \mathbb{T}^\# \rightarrow \mathbb{T}^\#$$

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

$$\text{step}^\# : (\mathbb{T}^\# \times \mathbb{P}) \rightarrow \mathbb{T}^\#$$

$$\begin{array}{ll} \text{step}^\#(t^\#, p) \triangleq [p \mapsto \bigsqcup_{p' \in t^\#(p).1.1(x)} (\perp, t^\#(p').2)] & \text{when } \{p : x\} \\ \sqcup [p \mapsto (\perp, ((\perp, \{\text{Read}^\#(p, x)\}), \emptyset))] & \text{if } t^\#(p).1.2 \neq \emptyset \\ \text{step}^\#(t^\#, p) \triangleq [p \mapsto (\perp, (\perp, \{\langle \lambda x.p', p' \rangle\}))] & \text{when } \{p : \lambda x.p'\} \\ \text{step}^\#(t^\#, p) \triangleq [p_1 \mapsto (t^\#(p).1, \perp)] & \text{when } \{p : p_1 p_2\} \\ \sqcup [p_2 \mapsto (t^\#(p).1, \perp)] & \\ \sqcup \bigsqcup_{\langle \lambda x.p', p'' \rangle \in t^\#(p_1).2.2} [p' \mapsto (t^\#(p'').1 \sqcup ([x \mapsto \{p_2\}], \emptyset), \perp)] & \\ \sqcup [p \mapsto \bigsqcup_{\langle \lambda x.p', _ \rangle \in t^\#(p_1).2.2} (\perp, t^\#(p').2)] & \\ \sqcup [p \mapsto (\perp, ((\perp, \{\text{Call}^\#(p_1, p_2)\}), \emptyset))] & \text{if } t^\#(p_1).2.1.2 \neq \emptyset \\ \text{step}^\#(t^\#, p) \triangleq [p_1 \mapsto (t^\#(p).1, \perp)] & \text{when } \{p : p_1 \times p_2\} \\ \sqcup [p_2 \mapsto (t^\#(p_1).2.1, \perp)] & \\ \sqcup [p \mapsto (\perp, t^\#(p_2).2)] & \\ \text{step}^\#(t^\#, p) \triangleq \perp & \text{when } \{p : \varepsilon\} \\ \text{step}^\#(t^\#, p) \triangleq [p_1 \mapsto (t^\#(p).1 \sqcup ([x \mapsto \{p_1\}], \emptyset), \perp)] & \text{when } \{p : x = p_1; p_2\} \\ \sqcup [p_2 \mapsto (t^\#(p).1 \sqcup ([x \mapsto \{p_1\}], \emptyset), \perp)] & \\ \sqcup [p \mapsto (\perp, (t^\#(p_2).2.1 \sqcup ([x \mapsto \{p_1\}], \emptyset), \emptyset))] & \end{array}$$

The abstract semantics $t^\#$ computed by

$$\llbracket p_0 \rrbracket^\#(\sigma_0^\#, t_0^\#) \triangleq \text{lfp}(\lambda t^\#. \text{Step}^\#(t^\#) \sqcup t_{\text{init}}^\#) \quad \text{where } t_{\text{init}} = t_0^\# \sqcup [p_0 \mapsto (\sigma_0^\#, \perp)]$$

is a sound abstraction of $\llbracket p_0 \rrbracket \Sigma_0$ when $\Sigma_0 \subseteq \gamma(\sigma_0^\#, t_0^\#)$.

3.3 Abstract linking

Now we define a sound linking operator that abstracts \bowtie . Assume we have

$$\sigma_0 \leq (\sigma_0^\#, t_0^\#) \quad t \subseteq \gamma(t^\#)$$

we define:

$$\sigma_0 \bowtie t \triangleq \lambda p. \bigcup_{\sigma \in t(p)} (\sigma_0 \bowtie \sigma) \cup \bigcup_{(\sigma, v) \in t(p)} \{(\sigma_+, v_+) \mid \sigma_+ \in \sigma_0 \bowtie \sigma \text{ and } v_+ \in \sigma_0 \bowtie v\}$$

We want to define $\bowtie^\#$ so that the following holds:

$$\sigma_0 \bowtie t \subseteq \gamma((\sigma_0^\#, t_0^\#) \bowtie^\# t^\#)$$

This is equivalent to saying that the linked result $t_+^\# = (\sigma_0^\#, t_0^\#) \bowtie^\# t^\#$ satisfies:

$$\sigma_0 \leq (\sigma_0^\#, t_0^\#) \text{ and } w \leq (v^\#, t^\#) \Rightarrow w_+ \leq (v_+^\#, t_+^\#)$$

for each $w_+ \in \sigma_0 \bowtie w$ and $p \in \mathbb{P}$, where $[v^\#, v_+^\#] \in \{[(t^\#(p).1, \emptyset), (t_+^\#(p).1, \emptyset)], [t^\#(p).2, t_+^\#(p).2]\}$.

The condition for $t_+^\#$ can be deduced by attempting the proof of the above in advance.

We proceed by induction on the derivation for

$$w_+ \in \sigma_0 \bowtie w$$

and inversion on $w \leq (v^\#, t^\#)$.

When:	$w = \text{Init},$	
Have:	$\text{Init}^\# \in v^\#.1.2$	
Need:	$v_+^\# \sqsupseteq \sigma_0^\#$ $t_+^\# \sqsupseteq t_0^\#$	
When:	$w = \text{Read}(E, x),$	
Have:	$\text{Read}^\#(p', x) \in v^\#.1.2$ and $[E] \leq t^\#(p').1$	
Need:	$v_+^\# \sqsupseteq t_+^\#(p'').2$ $v_+^\# \sqsupseteq ((\square, \{\text{Read}^\#(p', x)\}), \emptyset)$	for $p'' \in t_+^\#(p').1.1(x)$ if $t_+^\#(p').1.2 \neq \emptyset$
When:	$w = \text{Call}(E, v),$	
Have:	$\text{Call}^\#(p_1, p_2) \in v^\#.1.2$ and $E \leq t^\#(p_1).2$ and $v \leq t^\#(p_2).2$	
Need:	$v_+^\# \sqsupseteq t_+^\#(p').2$ $v_+^\# \sqsupseteq ((\square, \{\text{Call}^\#(p_1, p_2)\}), \emptyset)$ $t_+^\#(p') \sqsupseteq (t_+^\#(p'').1 \sqcup ([x \mapsto \{p_2\}], \emptyset), \emptyset)$ $t_+^\# \sqsupseteq \text{Step}^\#(t_+^\#)$	for $\langle \lambda x.p', p'' \rangle \in t_+^\#(p_1).2.2$ if $t_+^\#(p_1).2.1.2 \neq \emptyset$ for $\langle \lambda x.p', p'' \rangle \in t_+^\#(p_1).2.2$
When:	$w = (x, \ell^{p'}) :: \sigma,$	
Have:	$p' \in v^\#.1.1(x)$ and $\sigma \leq v^\#$	
Need:	$v_+^\#.1.1(x) \ni p'$	
When:	$w = (x, w') :: \sigma,$	
Have:	$p' \in v^\#.1.1(x)$ and $w' \in t^\#(p').1$ and $\sigma \leq v^\#$	
Need:	$v_+^\#.1.1(x) \ni p'$	
When:	$w = \langle \lambda x.p', \sigma \rangle,$	
Have:	$\langle \lambda x.p', p'' \rangle \in v^\#.2$ and $\sigma \leq t^\#(p'').1$	
Need:	$v_+^\#.2 \ni \langle \lambda x.p', p'' \rangle$	

The above conditions can be summarized by saying $t_+^\#$ is a post-fixed point of:

$$\lambda t_+^\#. \text{Step}^\#(t_+^\#) \sqcup \text{Link}^\#(\sigma_0^\#, t^\#, t_+^\#) \sqcup t_0^\#$$

where $\text{Link}^\#(\sigma_0^\#, t^\#, t_+^\#)$ is the least function that satisfies:

Let $\text{link}^\# = \text{Link}^\#(\sigma_0^\#, t^\#, t_+^\#)$ in For each $p \in \mathbb{P}$, when $v^\#, v_+^\# = (t^\#(p).1, \emptyset)$, $(\text{link}^\#(p).1, \emptyset)$ or when $v^\#, v_+^\# = t^\#(p).2, \text{link}^\#.2$	
If:	$\text{Init}^\# \in v^\#.1.2$
Then:	$v_+^\# \sqsupseteq \sigma_0^\#$
If:	$\text{Read}^\#(p', x) \in v^\#.1.2$
Then:	$v_+^\# \sqsupseteq t_+^\#(p'').2$ for $p'' \in t_+^\#(p').1.1(x)$ $v_+^\# \sqsupseteq ((\square, \{\text{Read}^\#(p', x)\}), \emptyset)$ if $t_+^\#(p').1.2 \neq \emptyset$
If:	$\text{Call}^\#(p_1, p_2) \in v^\#.1.2$
Then:	$v_+^\# \sqsupseteq t_+^\#(p').2$ for $\langle \lambda x.p', p'' \rangle \in t_+^\#(p_1).2.2$ $v_+^\# \sqsupseteq ((\square, \{\text{Call}^\#(p_1, p_2)\}), \emptyset)$ if $t_+^\#(p_1).2.1.2 \neq \emptyset$ $\text{link}^\#(p') \sqsupseteq (t_+^\#(p'').1 \sqcup ([x \mapsto \{p_2\}], \emptyset), \emptyset)$ for $\langle \lambda x.p', p'' \rangle \in t_+^\#(p_1).2.2$
If:	$p' \in v^\#.1.1(x)$
Then:	$v_+^\#.1.1(x) \ni p'$
If:	$p' \in v^\#.1.1(x)$
Then:	$v_+^\#.1.1(x) \ni p'$
If:	$\langle \lambda x.p', p'' \rangle \in v^\#.2$
Then:	$v_+^\#.2 \ni \langle \lambda x.p', p'' \rangle$

Note that the left-hand side contains only $\text{link}^\#$ and the right-hand side does not depend on the value of $\text{link}^\#$.

Some auxiliary lemmas:

Lemma 3.1 (Substitution of values).

$$w \leq (v^\#, t^\#) \text{ and } u \leq (t^\#(p).2, t^\#) \Rightarrow w[u/\ell^p] \leq (v^\#, t^\#)$$

Lemma 3.2 (Sound step[#]).

$$\forall p, t, t^\# : t \subseteq \gamma(t^\#) \Rightarrow \text{step}(t, p) \cup t \subseteq \gamma(\text{step}^\#(t^\#, p) \sqcup t^\#)$$

Lemma 3.3 (Sound Step[#]).

$$\forall t_{\text{init}}, t^\# : t_{\text{init}} \subseteq \gamma(t^\#) \text{ and } \text{Step}^\#(t^\#) \sqsubseteq t^\# \Rightarrow \text{lfp}(\lambda t. \text{Step}(t) \cup t_{\text{init}}) \subseteq \gamma(t^\#)$$