

# Modular Analysis

Joonhyup Lee

March 22, 2024

## 1 Syntax and Semantics

### 1.1 Abstract Syntax

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

Figure 1: Abstract syntax of the language.

### 1.2 Operational Semantics

Environment	$\sigma$	$\in$	Env	
Location	$\ell$	$\in$	Loc	
de Bruijn Index	$n$	$\in$	$\mathbb{N}$	
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	$\sigma$	$\rightarrow$	$\bullet$	empty stack
			$  (x, w) :: \sigma$	weak value binding
			$  (x, \ell) :: \sigma$	free location binding
			$  (x, n) :: \sigma$	bound location binding
Value	$v$	$\rightarrow$	$\sigma$	exported environment
			$  \langle \lambda x.e, \sigma \rangle$	closure
Weak Value	$w$	$\rightarrow$	$v$	value
			$  \mu.v$	recursive value

Figure 2: Definition of the semantic domains.

<div style="border: 1px solid black; padding: 2px; display: inline-block;"><math>\sigma \vdash e \Downarrow v</math></div>			
<b>ID</b> $\frac{\sigma(x) = v}{\sigma \vdash x \Downarrow v}$	<b>RECID</b> $\frac{\sigma(x) = \mu.v}{\sigma \vdash x \Downarrow \{0 \rightarrow \mu.v\}v}$	<b>FN</b> $\frac{}{\sigma \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle}$	<b>APP</b> $\frac{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad \sigma \vdash e_2 \Downarrow v_2 \quad (x, v_2) :: \sigma_1 \vdash e \Downarrow v}{\sigma \vdash e_1 e_2 \Downarrow v}$
<b>LINK</b> $\frac{\sigma \vdash e_1 \Downarrow \sigma_1 \quad \sigma_1 \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \bowtie e_2 \Downarrow v}$	<b>EMPTY</b> $\frac{}{\sigma \vdash \varepsilon \Downarrow \bullet}$	<b>BIND</b> $\frac{\ell \notin \text{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma \vdash e_1 \Downarrow \sigma_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma_2}$	

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  $\ell$  is chosen in the Bind rule.

### 1.3 Reconciling with Conventional Backpatching

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

Figure 4: Definition of the semantic domains with memory.

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

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

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

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

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

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

$w \sim_f v, m$

$$\begin{array}{c}
 \text{EQ-NIL} \\
 \frac{}{\bullet \sim_f \bullet} \quad \text{EQ-CONSFREE} \\
 \frac{\ell \notin \text{dom}(f) \quad \ell \notin \text{dom}(m) \quad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell) :: \sigma'} \quad \text{EQ-CONSBIND} \\
 \frac{f(\ell) = \ell' \quad \ell' \in \text{dom}(m) \quad \sigma \sim_f \sigma'}{(x, \ell) :: \sigma \sim_f (x, \ell') :: \sigma'}
 \end{array}$$

$$\begin{array}{c}
 \text{EQ-CONSWVAL} \\
 \frac{m(\ell') = v' \quad w \sim_f v' \quad \sigma \sim_f \sigma'}{(x, w) :: \sigma \sim_f (x, \ell') :: \sigma'} \quad \text{EQ-CLOS} \\
 \frac{\sigma \sim_f \sigma'}{\langle \lambda x.e, \sigma \rangle \sim_f \langle \lambda x.e, \sigma' \rangle} \quad \text{EQ-REC} \\
 \frac{L \text{ finite} \quad m(\ell') = v' \quad \forall \ell \notin L, \{0 \rightarrow \ell\} v \sim_{f[\ell \mapsto \ell']} v'}{\mu.v \sim_f v'}
 \end{array}$$

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.

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:

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

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:

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

The 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).

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

**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).

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

## 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	$\sigma$	$\rightarrow$	...	
			[ $E$ ]	answer to an event
Value	$v$	$\rightarrow$	...	
			$E$	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 one rule to the semantics in Figure 3 for the semantics to incorporate events.

$$\frac{\text{APPEVENT} \quad \sigma \vdash e_1 \Downarrow E \quad \sigma \vdash e_2 \Downarrow v}{\sigma \vdash e_1 e_2 \Downarrow \text{Call}(E, v)}$$

Now we need to formulate the *concrete linking* rules. The concrete linking rule  $\sigma_0 \times w$ , given an answer  $\sigma_0$  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:

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

Then the following holds:

$$\begin{array}{c}
\boxed{\infty \in \text{Env} \rightarrow \text{Event} \rightarrow \mathcal{P}(\text{Val})} \\
\sigma_0 \times \text{Init} \triangleq \{\sigma_0\} \\
\sigma_0 \times \text{Read}(E, x) \triangleq \{v_+ | \sigma_+ \in \sigma_0 \times E \wedge \sigma_+(x) = v_+\} \\
\quad \cup \{\{0 \rightarrow \mu.v_+\}v_+ | \sigma_+ \in \sigma_0 \times E \wedge \sigma_+(x) = \mu.v_+\} \\
\sigma_0 \times \text{Call}(E, v) \triangleq \{v'_+ | \langle \lambda x.e, \sigma_+ \rangle \in \sigma_0 \times E \wedge v_+ \in \sigma_0 \times v \wedge (x, v_+) :: \sigma_+ \vdash e \Downarrow v'_+\} \\
\quad \cup \{\text{Call}(E_+, v_+) | E_+ \in \sigma_0 \times E \wedge v_+ \in \sigma_0 \times v\} \\
\boxed{\infty \in \text{Env} \rightarrow \text{Env} \rightarrow \mathcal{P}(\text{Env})} \\
\sigma_0 \times \bullet \triangleq \{\bullet\} \\
\sigma_0 \times (x, \ell) :: \sigma \triangleq \{(x, \ell) :: \sigma_+ | \sigma_+ \in \sigma_0 \times \sigma\} \\
\sigma_0 \times (x, w) :: \sigma \triangleq \{(x, w_+) :: \sigma_+ | w_+ \in \sigma_0 \times w \wedge \sigma_+ \in \sigma_0 \times \sigma\} \\
\sigma_0 \times [E] \triangleq \{\sigma_+ | \sigma_+ \in \sigma_0 \times E\} \cup \{[E_+] | E_+ \in \sigma_0 \times E\} \\
\boxed{\infty \in \text{Env} \rightarrow \text{Val} \rightarrow \mathcal{P}(\text{Val})} \\
\sigma_0 \times \langle \lambda x.e, \sigma \rangle \triangleq \{\langle \lambda x.e, \sigma_+ \rangle | \sigma_+ \in \sigma_0 \times \sigma\} \\
\boxed{\infty \in \text{Env} \rightarrow \text{WVal} \rightarrow \mathcal{P}(\text{WVal})} \\
\sigma_0 \times \mu.v \triangleq \{\mu.\{0 \leftarrow \ell\}v_+ | \ell \notin \text{FLoc}(v) \cup \text{FLoc}(\sigma_0) \wedge v_+ \in \sigma_0 \times \{0 \rightarrow \ell\}v\}
\end{array}$$

Figure 8: Definition for concrete linking.

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

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

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

**Lemma 2.1** (Linking distributes under substitution). Let  $\sigma_0$  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:

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

**Lemma 2.2** (Linking is compatible with reads). Let  $\sigma_0$  be the external environment that is linked with some environment  $\sigma$ . Let  $v$  be the value obtained from reading  $x$  from  $\sigma$ . Let  $\text{unfold} : \text{WVal} \rightarrow \text{Val}$  be defined as:

$$\text{unfold}(\mu.v) \triangleq \{0 \rightarrow \mu.v\}v \quad \text{unfold}(v) \triangleq v$$

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

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

### 3 CFA

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

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

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

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

$$\begin{aligned}
\text{step}(t, p) &\triangleq [p \mapsto \{(\sigma, v) \mid \sigma \in t(p) \text{ and } \sigma(x) = v\}] && \text{when } g(p) = x \\
&\cup [p \mapsto \{(\sigma, \{0 \rightarrow \mu.v\}v) \mid \sigma \in t(p) \text{ and } \sigma(v) = \mu.v\}] \\
\text{step}(t, p) &\triangleq [p \mapsto \{(\sigma, \langle \lambda x.p', \sigma \rangle) \mid \sigma \in t(p)\}] && \text{when } g(p) = \lambda x.p' \\
\text{step}(t, p) &\triangleq [p_1 \mapsto \{\sigma \mid \sigma \in t(p)\}] && \text{when } g(p) = p_1 p_2 \\
&\cup [p_2 \mapsto \{\sigma \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)\}] && \text{when } g(p) = p_1 \times 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)\}] && \text{when } g(p) = \varepsilon \\
\text{step}(t, p) &\triangleq [p_1 \mapsto \{(x, \ell) :: \sigma \mid \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 \mid \sigma \in t(p) \text{ and } ((x, \ell) :: \sigma, v_1) \in t(p_1)\}] \\
&\cup [p \mapsto \{(\sigma, (x, w_1) :: \sigma_2) \mid \sigma \in t(p) \text{ and } ((x, w_1) :: \sigma, \sigma_2) \in t(p_2)\}]
\end{aligned}$$

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,  $(\sigma, v)$  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  $\sigma_0 \vdash p_0 \Downarrow v_0$ .

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 proof tree	$t^\#$	$\in$	$\mathbb{T}^\# \triangleq \mathbb{P} \rightarrow \text{Env}^\# \times \text{Val}^\#$
	$E^\#$	$\rightarrow$	$\text{Init}^\# \mid \text{Read}^\#(p, x) \mid \text{Call}^\#(p, p)$

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

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

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

$$\begin{aligned} \text{step}^\#(t^\#, p) &\triangleq [p \mapsto \bigsqcup_{p' \in t^\#(p).1.1(x)} (t^\#(p).1, t^\#(p').2)] && \text{when } g(p) = x \\ &\sqcup [p \mapsto \bigsqcup_{t^\#(p).1.2 \neq \emptyset} (t^\#(p).1, (([], \{\text{Read}^\#(p, x)\}), \emptyset))] \\ \text{step}^\#(t^\#, p) &\triangleq [p \mapsto (t^\#(p).1, (\perp, \{\langle \lambda x.p', p \rangle\}))] && \text{when } g(p) = \lambda x.p' \\ \text{step}^\#(t^\#, p) &\triangleq [p_1 \mapsto (t^\#(p).1, \perp)] && \text{when } g(p) = p_1 p_2 \\ &\sqcup [p_2 \mapsto (t^\#(p).1, \perp)] \\ &\sqcup \bigsqcup_{\langle \lambda x.p', p'' \rangle \in t^\#(p_1).2.2} [p' \mapsto (t^\#(p'').1 \sqcup ([x \mapsto \{p_2\}], \emptyset), \perp)] \\ &\sqcup [p \mapsto \bigsqcup_{\langle \lambda x.p', \_ \rangle \in t^\#(p_1).2.2} (t^\#(p).1, t^\#(p').2)] \\ &\sqcup [p \mapsto \bigsqcup_{t^\#(p_1).2.1.2 \neq \emptyset} (t^\#(p).1, (([], \{\text{Call}^\#(p_1, p_2)\}), \emptyset))] \\ \text{step}^\#(t^\#, p) &\triangleq [p_1 \mapsto (t^\#(p).1, \perp)] && \text{when } g(p) = p_1 \times p_2 \\ &\sqcup [p_2 \mapsto (t^\#(p_1).2.1, \perp)] \\ &\sqcup [p \mapsto (t^\#(p).1, t^\#(p_2).2)] \\ \text{step}^\#(t^\#, p) &\triangleq [p \mapsto (t^\#(p).1, \perp)] && \text{when } g(p) = \varepsilon \\ \text{step}^\#(t^\#, p) &\triangleq [p_1 \mapsto \{(x, \ell) :: \sigma \mid \sigma \in t(p) \text{ and } \ell \notin \text{FLoc}(\sigma)\}] && \text{when } g(p) = x = p_1; p_2 \\ &\sqcup [p_2 \mapsto \{(x, \mu.\{0 \leftarrow \ell\}v_1) :: \sigma \mid \sigma \in t(p) \text{ and } ((x, \ell) :: \sigma, v_1) \in t(p_1)\}] \\ &\sqcup [p \mapsto \{(\sigma, (x, w_1) :: \sigma_2) \mid \sigma \in t(p) \text{ and } ((x, w_1) :: \sigma, \sigma_2) \in t(p_2)\}] \end{aligned}$$