Control Flow Analysis as an Instance for Modular Analysis

Joonhyup Lee

December 12, 2023

1 0CFA for the Simple Module Language

$$
\begin{array}{rrcll}
\text{Identifiers} & x, d & \in & \text{Var} & \\
\text{Expression} & e & \to & x \mid \lambda x.e \mid e\ e & \text{untyped } \lambda\text{-calculus} \\
& & \mid & m \rtimes e & \text{linked expression} \\
\text{Module} & m & \to & \varepsilon & \text{empty module} \\
& & \mid & d & \text{module identifier} \\
& & \mid & \texttt{val}\ x\ e\ m & \text{value binding} \\
& & \mid & \texttt{mod}\ d\ m\ m & \text{module binding}
\end{array}
$$

Figure 1: Abstract syntax of the simple module language.

1.1 Instrumented Operational Semantics

$$
\begin{array}{rrcll}
\text{Timestamp} & t & \in & \mathbb{T} & \\
\text{Address} & \ell & \in & \text{Addr} \triangleq \mathbb{N} \times \mathbb{T} & \\
\text{Environment/Context} & \sigma & \in & \text{Ctx} \triangleq \text{Var} \xrightarrow{\text{fin}} \text{Addr} + \text{Ctx} & \\
\text{Value of expressions} & v & \in & \text{Val} \triangleq \text{Var} \times \text{Expr} \times \text{Ctx} & \\
\text{Heap} & h & \in & \text{Heap} \triangleq \text{Addr} \xrightarrow{\text{fin}} \text{Val} & \\
\text{State} & s & \in & \text{State} \triangleq \text{Ctx} \times \text{Heap} & \\
\text{Context} & \sigma & \to & \bullet & \text{empty stack} \\
& & \mid & (x, \ell) :: \sigma & \text{value binding} \\
& & \mid & (d, \sigma) :: \sigma & \text{module binding} \\
\text{Value of expressions} & v & \to & \langle \lambda x.e, \sigma \rangle & \text{closure}
\end{array}
$$

Figure 2: Definition of the semantic domains.

$$\boxed{(e, s) \Downarrow (v, h) \text{ and } (m, s) \Downarrow (\sigma, h)}$$

$$
[\textsc{ExprID}]\ \frac{\begin{array}{c}\ell = \sigma(x) \\ v = h(\ell)\end{array}}{(x, \sigma, h) \Downarrow (v, h)}
\qquad
[\textsc{Fn}]\ \frac{}{(\lambda x.e, \sigma, h) \Downarrow (\langle \lambda x.e, \sigma \rangle, h)}
\qquad
[\textsc{App}]\ \frac{\begin{array}{c}(e_1, \sigma, h) \Downarrow (\langle \lambda x.e, \sigma_\lambda \rangle, h_\lambda) \\ (e_2, \sigma, h_\lambda) \Downarrow (v, h_a) \\ (e, (x, \ell) :: \sigma_\lambda, h_a[\ell \mapsto v]) \Downarrow (v', h')\end{array} \quad \begin{array}{c} n \in \mathsf{fresh}(h_a) \\ t = \mathsf{tick}(x) \\ \ell = (n, t)\end{array}}{(e_1\ e_2, \sigma, h) \Downarrow (v', h')}
$$

$$
[\textsc{Link}]\ \frac{\begin{array}{c}(m_1, \sigma, h) \Downarrow (\sigma', h') \\ (e_2, \sigma', h') \Downarrow (v', h'')\end{array}}{(m_1 \rtimes e_2, \sigma, h) \Downarrow (v', h'')}
\qquad
[\textsc{Empty}]\ \frac{}{(\varepsilon, \sigma, h) \Downarrow (\bullet, h)}
\qquad
[\textsc{ModID}]\ \frac{\sigma' = \sigma(d)}{(d, \sigma, h) \Downarrow (\sigma', h)}
$$

$$
[\textsc{LetE}]\ \frac{\begin{array}{c}(e_1, \sigma, h) \Downarrow (v, h') \\ (m_2, (x, \ell) :: \sigma, h'[\ell \mapsto v]) \Downarrow (\sigma', h'')\end{array} \quad \begin{array}{c} n \in \mathsf{fresh}(h') \\ t = \mathsf{tick}(x) \\ \ell = (n, t)\end{array}}{(\texttt{val}\ x\ e_1\ m_2, \sigma, h) \Downarrow ((x, \ell) :: \sigma', h'')}
\qquad
[\textsc{LetM}]\ \frac{\begin{array}{c}(m_1, \sigma, h) \Downarrow (\sigma', h') \\ (m_2, (d, \sigma') :: \sigma, h') \Downarrow (\sigma'', h'')\end{array}}{(\texttt{mod}\ d\ m_1\ m_2, \sigma, h) \Downarrow ((d, \sigma') :: \sigma'', h'')}
$$

Figure 3: The big-step operational semantics.

1.2 Adding Intermediate Transitions

$$\boxed{(e \text{ or } m, s) \hookrightarrow (e \text{ or } m, s)}$$

$$[\textsc{AppL}]\ \frac{}{(e_1\, e_2, \sigma, h) \hookrightarrow (e_1, \sigma, h)} \qquad [\textsc{AppR}]\ \frac{(e_1, \sigma, h) \Downarrow (\langle \lambda x.e, \sigma_\lambda \rangle, h_\lambda)}{(e_1\, e_2, \sigma, h) \hookrightarrow (e_2, \sigma, h_\lambda)}$$

$$[\textsc{AppBody}]\ \frac{\begin{array}{c}(e_1, \sigma, h) \Downarrow (\langle \lambda x.e, \sigma_\lambda \rangle, h_\lambda) \\ (e_2, \sigma, h_\lambda) \Downarrow (v, h_a)\end{array} \quad \begin{array}{c} n \in \mathsf{fresh}(h_a) \\ t = \mathsf{tick}(x) \\ \ell = (n, t)\end{array}}{(e_1\, e_2, \sigma, h) \hookrightarrow (e, (x, \ell) :: \sigma_\lambda, h_a[\ell \mapsto v])}$$

$$[\textsc{LinkL}]\ \frac{}{(m_1 \rtimes e_2, \sigma, h) \hookrightarrow (m_1, \sigma, h)} \qquad [\textsc{LinkR}]\ \frac{(m_1, \sigma, h) \Downarrow (\sigma', h')}{(m_1 \rtimes e_2, \sigma, h) \hookrightarrow (e_2, \sigma', h')}$$

$$[\textsc{LetEL}]\ \frac{}{(\texttt{val}\ x\ e_1\ m_2, \sigma, h) \hookrightarrow (e_1, \sigma, h)} \qquad [\textsc{LetER}]\ \frac{(e_1, \sigma, h) \Downarrow (v, h') \quad \begin{array}{c} n \in \mathsf{fresh}(h') \\ t = \mathsf{tick}(x) \\ \ell = (n, t)\end{array}}{(\texttt{val}\ x\ e_1\ m_2, \sigma, h) \hookrightarrow (m_2, (x, \ell) :: \sigma, h'[\ell \mapsto v])}$$

$$[\textsc{LetML}]\ \frac{}{(\texttt{mod}\ d\ m_1\ m_2, \sigma, h) \hookrightarrow (m_1, \sigma, h)} \qquad [\textsc{LetMR}]\ \frac{(m_1, \sigma, h) \Downarrow (\sigma', h')}{(\texttt{mod}\ d\ m_1\ m_2, \sigma, h) \hookrightarrow (m_2, (d, \sigma') :: \sigma, h')}$$

Figure 4: The transition relation for collecting all intermediate configurations.

1.3 Collecting Semantics

$$\begin{array}{rccl}
\text{Configuration} & c & \in & \text{Config} \triangleq (\text{Expr} + \text{Module}) \times \text{State} \\
\text{Result} & r & \in & \text{Result} \triangleq (\text{Val} + \text{Ctx}) \times \text{Heap} \\
\text{Judgements} & J & \subseteq & \text{Judge} \triangleq \text{Config} + \Downarrow
\end{array}$$

Figure 5: Definition of additional semantic domains for collecting semantics.

$$\begin{aligned}
\mathsf{Eval}(J) &\triangleq \left\{ c \Downarrow r \middle| \frac{P}{c \Downarrow r} \text{ and } P \subseteq J \text{ and } c \in J \right\} \\
\mathsf{Step}(J) &\triangleq \left\{ c' \middle| \frac{P}{c \hookrightarrow c'} \text{ and } P \subseteq J \text{ and } c \in J \right\} \\
[\![e]\!]S &\triangleq \mathsf{lfp}(\lambda X.\mathsf{Step}(X) \cup \mathsf{Eval}(X) \cup \{(e, s) | s \in S\})
\end{aligned}$$

1.4 Abstract Operational Semantics

$$\begin{array}{rcll}
\text{Abstract Context} & \dot\sigma & \in & \dot{\text{Ctx}} \triangleq \text{Var} \xrightarrow{\text{fin}} \mathbb{T} + \dot{\text{Ctx}} & \\
\text{Abstract Value} & \dot v & \in & \dot{\text{Val}} \triangleq \text{Var} \times \text{Expr} \times \dot{\text{Ctx}} & \\
\text{Abstract Heap} & \dot h & \in & \dot{\text{Heap}} \triangleq \mathbb{T} \xrightarrow{\text{fin}} \mathcal{P}(\dot{\text{Val}}) & \\
\text{Abstract State} & \dot s & \in & \dot{\text{State}} \triangleq \dot{\text{Ctx}} \times \dot{\text{Heap}} & \\
\text{Abstract Context} & \dot\sigma & \rightarrow & \bullet & \text{empty stack} \\
 & & | & (x, t) :: \dot\sigma & \text{value binding} \\
 & & | & (d, \dot\sigma) :: \dot\sigma & \text{module binding} \\
\text{Abstract Value} & \dot v & \rightarrow & \langle \lambda x.e, \dot\sigma \rangle & \text{abstract closure}
\end{array}$$

Figure 6: Definition of the abstract semantic domains.

$$\boxed{(e, \dot{s}) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h}) \text{ and } (m, \dot{s}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}, \dot{h})}$$

$$[\textsc{ExprID}]\ \frac{\begin{array}{c} t = \dot{\sigma}(x) \\ \dot{v} \in \dot{h}(t) \end{array}}{(x, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h})} \qquad [\textsc{Fn}]\ \frac{}{(\lambda x.e, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\langle \lambda x.e, \dot{\sigma} \rangle, \dot{h})}$$

$$[\textsc{App}]\ \frac{\begin{array}{c} (e_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\langle \lambda x.e, \dot{\sigma}_\lambda \rangle, \dot{h}_\lambda) \\ (e_2, \dot{\sigma}, \dot{h}_\lambda) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h}_a) \qquad t = \mathsf{tick}(x) \\ (e, (x,t) :: \dot{\sigma}_\lambda, \dot{h}_a[t \mathrel{\dot{\mapsto}} \dot{v}]) \mathrel{\dot{\Downarrow}} (\dot{v}', \dot{h}') \end{array}}{(e_1\ e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{v}', \dot{h}')}$$

$$[\textsc{Link}]\ \frac{\begin{array}{c} (m_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h}') \\ (e_2, \dot{\sigma}', \dot{h}') \mathrel{\dot{\Downarrow}} (\dot{v}', \dot{h}'') \end{array}}{(m_1 \rtimes e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{v}', \dot{h}'')} \qquad [\textsc{Empty}]\ \frac{}{(\varepsilon, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\bullet, \dot{h})} \qquad [\textsc{ModID}]\ \frac{\dot{\sigma}' = \dot{\sigma}(d)}{(d, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h})}$$

$$[\textsc{LetE}]\ \frac{\begin{array}{c} (e_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h}') \qquad t = \mathsf{tick}(x) \\ (m_2, (x,t) :: \dot{\sigma}, \dot{h}'[t \mathrel{\dot{\mapsto}} \dot{v}]) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h}'') \end{array}}{(\texttt{val}\ x\ e_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} ((x,t) :: \dot{\sigma}', \dot{h}'')} \qquad [\textsc{LetM}]\ \frac{\begin{array}{c} (m_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h}') \\ (m_2, (d, \dot{\sigma}') :: \dot{\sigma}, \dot{h}') \mathrel{\dot{\Downarrow}} (\dot{\sigma}'', \dot{h}'') \end{array}}{(\texttt{mod}\ d\ m_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} ((d, \dot{\sigma}') :: \dot{\sigma}'', \dot{h}'')}$$

Figure 7: The big-step abstract operational semantics.

$$\boxed{(e \text{ or } m, \dot{s}) \mathrel{\dot{\hookrightarrow}} (e \text{ or } m, \dot{s})}$$

$$[\textsc{AppL}]\ \frac{}{(e_1\ e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (e_1, \dot{\sigma}, \dot{h})} \qquad [\textsc{AppR}]\ \frac{(e_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\langle \lambda x.e, \dot{\sigma}_\lambda \rangle, \dot{h}_\lambda)}{(e_1\ e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (e_2, \dot{\sigma}, \dot{h}_\lambda)}$$

$$[\textsc{AppBody}]\ \frac{\begin{array}{c} (e_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\langle \lambda x.e, \dot{\sigma}_\lambda \rangle, \dot{h}_\lambda) \qquad t = \mathsf{tick}(x) \\ (e_2, \dot{\sigma}, \dot{h}_\lambda) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h}_a) \end{array}}{(e_1\ e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (e, (x,t) :: \dot{\sigma}_\lambda, \dot{h}_a[t \mathrel{\dot{\mapsto}} \dot{v}])}$$

$$[\textsc{LinkL}]\ \frac{}{(m_1 \rtimes e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (m_1, \dot{\sigma}, \dot{h})} \qquad [\textsc{LinkR}]\ \frac{(m_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h}')}{(m_1 \rtimes e_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (e_2, \dot{\sigma}', \dot{h}')}$$

$$[\textsc{LetEL}]\ \frac{}{(\texttt{val}\ x\ e_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (e_1, \dot{\sigma}, \dot{h})} \qquad [\textsc{LetER}]\ \frac{(e_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{v}, \dot{h}') \qquad t = \mathsf{tick}(x)}{(\texttt{val}\ x\ e_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (m_2, (x,t) :: \dot{\sigma}, \dot{h}'[t \mathrel{\dot{\mapsto}} \dot{v}])}$$

$$[\textsc{LetML}]\ \frac{}{(\texttt{mod}\ d\ m_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (m_1, \dot{\sigma}, \dot{h})} \qquad [\textsc{LetMR}]\ \frac{(m_1, \dot{\sigma}, \dot{h}) \mathrel{\dot{\Downarrow}} (\dot{\sigma}', \dot{h}')}{(\texttt{mod}\ d\ m_1\ m_2, \dot{\sigma}, \dot{h}) \mathrel{\dot{\hookrightarrow}} (m_2, (d, \dot{\sigma}') :: \dot{\sigma}, \dot{h}')}$$

Figure 8: The abstract transition relation for collecting all intermediate configurations.

1.5 Abstract Semantics

$$\begin{array}{rcll} \text{Abstract Configuration} & \dot{c} & \in & \dot{\text{Config}} \triangleq (\text{Expr} + \text{Module}) \times \dot{\text{State}} \\ \text{Abstract Result} & \dot{r} & \in & \dot{\text{Result}} \triangleq (\dot{\text{Val}} + \dot{\text{Ctx}}) \times \dot{\text{Heap}} \\ \text{Abstract Judgements} & J^{\#} & \subseteq & \dot{\text{Judge}} \triangleq \dot{\text{Config}} + \dot{\Downarrow} \end{array}$$

Figure 9: Definition of additional semantic domains for collecting semantics.

$$\mathsf{Eval}^\#(J^\#) \triangleq \left\{ \dot{c} \Downarrow \dot{r} \,\middle|\, \frac{P^\#}{\dot{c} \Downarrow \dot{r}} \text{ and } P^\# \subseteq J^\# \text{ and } \dot{c} \in J^\# \right\}$$

$$\mathsf{Step}^\#(J^\#) \triangleq \left\{ \dot{c'} \,\middle|\, \frac{P^\#}{\dot{c} \hookrightarrow \dot{c'}} \text{ and } P^\# \subseteq J^\# \text{ and } \dot{c} \in J^\# \right\}$$

$$\llbracket e \rrbracket^\# S^\# \triangleq \mathsf{lfp}(\lambda X^\#.\mathsf{Step}^\#(X^\#) \cup \mathsf{Eval}^\#(X^\#) \cup \{(e, \dot{s}) | \dot{s} \in S^\#\})$$

1.6 Galois Connection

$$|\bullet| \triangleq \bullet \qquad |(x, (\_, t)) :: \sigma| \triangleq (x, t) :: |\sigma|$$

$$|(d, \sigma) :: \sigma'| \triangleq (d, |\sigma|) :: |\sigma'| \qquad |\langle \lambda x.e, \sigma \rangle| \triangleq \langle \lambda x.e, |\sigma| \rangle$$

$$|h| \triangleq \lambda t.\{|h(n, t)| | n \in \mathbb{N}\} \qquad |(\_, \_)| \triangleq (|\_|, |\_|)$$

Lemma 1.1 (Galois Connection).

$$\mathcal{P}(\text{Judge}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathcal{P}(\text{Ju}\dot{\text{d}}\text{ge})$$

where

$$\gamma(D^\#) \triangleq \{c | |c| \in D^\#\} \cup \{c \Downarrow r | |c| \dot{\Downarrow} |r| \in D^\#\}$$

$$\alpha(D) \triangleq \{|c| | c \in D\} \cup \{|c| \dot{\Downarrow} |r| | c \Downarrow r \in D\}$$

1.7 Soundness

Lemma 1.2 (Operational Soundness).

$$c \Downarrow r \Rightarrow |c| \dot{\Downarrow} |r|$$

Lemma 1.3 (Soundness of Step, Eval).

$$\mathsf{Step} \circ \gamma \subseteq \gamma \circ \mathsf{Step}^\# \qquad \mathsf{Eval} \circ \gamma \subseteq \gamma \circ \mathsf{Eval}^\#$$

Theorem 1.1 (Soundness).

$$\llbracket e \rrbracket \circ \gamma \subseteq \gamma \circ \llbracket e \rrbracket^\#$$

1.8 0CFA

Instantiating

$$\mathsf{tick}(x) \triangleq x$$

leads to 0CFA.

2 Punching Holes

Punch holes in the abstract semantics and define injection/linking operators such that:

$$\llbracket e \rrbracket^\# (S^\# \rhd S_H) \sqsubseteq S^\# \propto \llbracket e \rrbracket_H S_H$$

The goal is to define:

$\llbracket \cdot \rrbracket_H$	semantics with holes
S_H	states with holes
$\propto$	semantic linking
$\rhd$	semantic injection

2.1 States with Holes

First define:

$$\sigma_H \in \mathrm{Ctx}_H$$

by:

$$\begin{array}{rcll} \sigma_H & \to & [] & \text{hole} \\ & | & \bullet & \text{empty stack} \\ & | & (x,t) :: \sigma_H & \text{value binding} \\ & | & (d,\sigma_H) :: \sigma_H & \text{module binding} \end{array}$$

and define:

$$\mathrm{Val}_H \qquad \mathrm{Heap}_H \qquad \mathrm{State}_H \qquad \mathrm{Config}_H \qquad \mathrm{Result}_H \qquad \mathrm{Judge}_H$$

accordingly.

2.2 Semantics with Holes

Define:

$$\Downarrow_H \subseteq \mathrm{Config}_H \times \mathrm{Result}_H \qquad \hookrightarrow_H \subseteq \mathrm{Config}_H \times \mathrm{Config}_H$$

by using:

$$\mathsf{tick}_H \in \mathrm{Var} \to \mathbb{T}_H$$

where $\mathbb{T}_H$ is some set chosen by the analysis designer.

Define:

$$\mathsf{Step}_H \in \mathcal{P}(\mathrm{Judge}_H) \to \mathcal{P}(\mathrm{Judge}_H) \qquad \mathsf{Eval}_H \in \mathcal{P}(\mathrm{Judge}_H) \to \mathcal{P}(\mathrm{Judge}_H)$$

Define:

$$[\![e]\!]_H S_H \triangleq \mathsf{lfp}(\lambda X_H.\mathsf{Step}_H(X) \cup \mathsf{Eval}_H(X) \cup \{(e,s_H)|s_H \in S_H\})$$

2.3 Injection

Define injection into a context with holes:

$$\cdot[\cdot] : \mathrm{Ctx}_H \to \dot{\mathrm{Ctx}} \to \dot{\mathrm{Ctx}}_+$$

where

$$\dot{\mathrm{Ctx}}_+ \triangleq \mathrm{Var} \xrightarrow{\text{fin}} \mathbb{T} + \mathbb{T}_H$$

by:

$$[][\dot\sigma] \triangleq \dot\sigma \qquad \bullet[\dot\sigma] \triangleq \bullet$$

$$((x,t) :: \dot\sigma_H)[\dot\sigma] \triangleq (x,t) :: \dot\sigma_H[\dot\sigma] \qquad ((d,\dot\sigma_H) :: \dot\sigma_H')[\dot\sigma] \triangleq (d, \dot\sigma_H[\dot\sigma]) :: \dot\sigma_H'[\dot\sigma]$$

Define injection into other domains where

$$\dot{\mathrm{Val}}_+ \qquad \dot{\mathrm{Heap}}_+ \qquad \dot{\mathrm{State}}_+ \qquad \dot{\mathrm{Config}}_+ \qquad \dot{\mathrm{Result}}_+ \qquad \dot{\mathrm{Judge}}_+$$

are defined accordingly as domains that use $\mathbb{T} + \mathbb{T}_H$ for timestamps and tick_H for tick by:

$$\langle \lambda x.e, \sigma_H\rangle[\dot\sigma] \triangleq \langle \lambda x.e, \sigma_H[\dot\sigma]\rangle \qquad h_H[\dot\sigma] \triangleq \lambda t_H.\{v_H[\dot\sigma]|v_H \in h_H(t_H)\}$$

$$(\sigma_H, h_H)[(\dot\sigma, \dot h)] \triangleq (\sigma_H[\dot\sigma], \dot h \cup h_H[\dot\sigma]) \qquad (v_H, h_H)[(\dot\sigma, \dot h)] \triangleq (v_H[\dot\sigma], \dot h \cup h_H[\dot\sigma])$$

$$(e, s_H)[\dot s] \triangleq (e, s_H[\dot s]) \qquad (m, s_H)[\dot s] \triangleq (m, s_H[\dot s])$$

Then we have:

Lemma 2.1 (Preservation After Injection)**.**

$$c_H \Downarrow_H r_H \Rightarrow c_H[\dot s] \,\dot\Downarrow\, r_H[\dot s] \qquad c_H \hookrightarrow_H c_H' \Rightarrow c_H[\dot s] \,\dot\hookrightarrow\, c_H'[\dot s]$$

Define injection into set of states:

$$\rhd : \mathcal{P}(\dot{\mathrm{State}}) \to \mathcal{P}(\mathrm{State}_H) \to \mathcal{P}(\dot{\mathrm{State}}_+)$$

by:

$$S^\# \rhd S_H \triangleq \{\sigma_H[\dot\sigma]|\dot\sigma \in S^\#, \sigma_H \in S_H\}$$

2.4 Linking

Define injection into set of judgements:

$$\triangleright : \mathcal{P}(\dot{\text{State}}) \to \mathcal{P}(\text{Judge}_H) \to \mathcal{P}(\dot{\text{Judge}}_+)$$

by:

$$S^\# \triangleright J_H \triangleq \{c_H[\dot{s}] | \dot{s} \in S^\#, c_H \in J_H\} \cup \{c_H[\dot{s}] \dot{\Downarrow} r_H[\dot{s}] | \dot{s} \in S^\#, c_H \Downarrow_H r_H \in J_H\}$$

Define linking:

$$\propto : \mathcal{P}(\dot{\text{State}}) \to \mathcal{P}(\text{Judge}_H) \to \mathcal{P}(\dot{\text{Judge}}_+)$$

by:

$$S^\# \propto J_H \triangleq \mathsf{lfp}(\lambda X_+^\#.\mathsf{Step}_+^\#(X_+^\#) \cup \mathsf{Eval}_+^\#(X_+^\#) \cup (S^\# \triangleright J_H))$$

Lemma 2.2 (Advance)**.**

$$[\![e]\!]^\#(S^\# \triangleright S_H) = S^\# \propto [\![e]\!]_H S_H$$

2.5 Separate Modular Analysis

Theorem 2.1 (Separate Analysis)**.** Let $\mathsf{emp} \triangleq \{(\bullet, \emptyset)\} \subseteq \dot{\text{State}}$. Then:

$$([\![m \rtimes e]\!]^\# \mathsf{emp})|_e \cong^\# (S^\# \propto [\![e]\!]_H S_H)|_e$$

when

$$([\![m]\!]^\# \mathsf{emp})|_m \cong^\# S^\# \triangleright S_H$$