1 Evaluation Rules

Notations

- ℓ : A location in memory.
- $\langle \rho, \sigma, \mu, S \rangle$: Machine configuration
  - ρ : Environment table that maps identifiers to locations (ℓ).
  - σ : Storage table that maps locations (ℓ) to values (v).
  - μ : Reassignability table that maps locations (ℓ) to either `var` (reassignable) or `const` (non-reassignable).
  - S : Statement or expression currently being executed.

Values

$$v \in \mathcal{V} ::= i \in \mathbb{Z} \mid n \in \mathbb{R} \mid b \in \{\texttt{true}, \texttt{false}\} \mid \ldots$$

Unary Operations

$$op_u \in \{+,\ -,\ !,\ \sim,\ \&\}$$

$$\frac{\langle \rho, \sigma, \mu, e \rangle \to \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, (op_u\ e) \rangle \to \langle \rho', \sigma', \mu', (op_u\ e') \rangle} \quad \text{(E-UnaryOpStep)}$$

$$\frac{v_2 = op_u\ v_1}{\langle \rho, \sigma, \mu, (op_u\ v_1) \rangle \to \langle \rho, \sigma, \mu, v_2 \rangle} \quad \text{(E-UnaryOp)}$$

Binary Operations

$op_b \in \{+,\ -,\ *,\ /,\ \%,\ ==,\ !=,\ >,\ <,\ >=,\ <=,\ \&\&,\ ||,\ \&,\ |,\ \wedge,\ <<,\ >>\}$

$$\frac{\langle \rho, \sigma, \mu, e_1 \rangle \to \langle \rho', \sigma', \mu', e_1' \rangle}{\langle \rho, \sigma, \mu, (e_1\ op_b\ e_2) \rangle \to \langle \rho', \sigma', \mu', (e_1'\ op_b\ e_2) \rangle} \quad \text{(E-BinaryOpLeft)}$$

$$\frac{\langle \rho, \sigma, \mu, e_2 \rangle \to \langle \rho', \sigma', \mu', e_2' \rangle}{\langle \rho, \sigma, \mu, (v_1\ op_b\ e_2) \rangle \to \langle \rho', \sigma', \mu', (v_1\ op_b\ e_2') \rangle} \quad \text{(E-BinaryOpRight)}$$

$$\frac{v_3 = v_1\ op_b\ v_2}{\langle \rho, \sigma, \mu, (v_1\ op_b\ v_2) \rangle \to \langle \rho, \sigma, \mu, v_3 \rangle} \quad \text{(E-BinaryOp)}$$

Application

$$\frac{\langle \rho, \sigma, \mu, e \rangle \to \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, e(e_1, e_2, \ldots, e_n) \rangle \to \langle \rho', \sigma', \mu', e'(e_1, e_2, \ldots, e_n) \rangle} \quad \text{(E-AppStep)}$$

$$\frac{\langle \rho, \sigma, \mu, e_i \rangle \to \langle \rho', \sigma', \mu', e_i' \rangle}{\begin{array}{l}\langle \rho, \sigma, \mu, f(v_1, \ldots, v_{i-1}, e_i, \ldots, e_n) \rangle \to \\ \langle \rho', \sigma', \mu', f(v_1, \ldots, v_{i-1}, e_i', \ldots, e_n') \rangle\end{array}} \quad \text{(E-AppArgs)}$$

$$\frac{f = \text{closure}(\text{params}, \text{captures}, T_{\text{ret}}, S_{\text{body}}, \rho_c) \quad |params| = n}{\langle \rho, \sigma, \mu, f(v_1, \ldots, v_n) \rangle \to \langle \rho_c[params \mapsto (v_1 \ldots v_n)], \sigma, \mu, S_{\text{body}} \rangle} \quad \text{(E-App)}$$

Subscript

$$e_1[e_2] \stackrel{\text{def}}{=} \text{subscript}(e_1, e_2)$$

Variable Definitions

$$\frac{\langle \rho, \sigma, \mu, e \rangle \to \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \texttt{let } x = e \rangle \to \langle \rho', \sigma', \mu', \texttt{let } x = e' \rangle} \quad \text{(St-LetStep)}$$

$$\frac{\ell \notin \text{dom}(\sigma) \quad v \in \mathcal{V}}{\langle \rho, \sigma, \mu, \texttt{let } x = v \ ; \ S \rangle \to \langle \rho[x \mapsto \ell], \sigma[\ell \mapsto v], \mu[\ell \mapsto \texttt{var}], S \rangle} \quad \text{(St-Let)}$$

$$\frac{\langle \rho, \sigma, \mu, e \rangle \to \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \texttt{const } x = e \rangle \to \langle \rho', \sigma', \mu', \texttt{const } x = e' \rangle} \quad \text{(St-ConstStep)}$$

$$\frac{\ell \notin \text{dom}(\sigma) \quad v \in \mathcal{V}}{\langle \rho, \sigma, \mu, \texttt{const } x = v \ ; \ S \rangle \to \langle \rho[x \mapsto \ell], \sigma[\ell \mapsto v], \mu[\ell \mapsto \texttt{const}], S \rangle} \quad \text{(St-Const)}$$

Assignment

$$\frac{\langle \rho, \sigma, \mu, e \rangle \to \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, x = e \rangle \to \langle \rho', \sigma', \mu', x = e' \rangle} \quad \text{(St-AssignStep)}$$

$$\frac{\rho(x) = \ell \quad \mu(\ell) = \texttt{var} \quad v \in \mathcal{V}}{\langle \rho, \sigma, \mu, x = v \ ; \ S \rangle \to \langle \rho, \sigma[\ell \mapsto v], \mu, S \rangle} \quad \text{(St-Assign)}$$

Scope

$$\frac{\langle \rho, \sigma, \mu, S \rangle \to \langle \rho', \sigma', \mu', S' \rangle}{\langle \rho, \sigma, \mu, \texttt{do } S \texttt{ end} \rangle \to \langle \rho', \sigma', \mu', \texttt{do } S' \texttt{ end} \rangle} \quad \text{(St-Scope)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{do skip end} \ ; \ S \rangle \to \langle \rho, \sigma, \mu, S \rangle} \quad \text{(St-ScopeExit)}$$

Function Definition

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{fn } f = e\rangle \to \langle \rho', \sigma', \mu', \texttt{fn } f = e'\rangle} \quad \text{(St-FnDefStep)}$$

$$\frac{\ell \notin \mathrm{dom}(\sigma) \quad v = \mathrm{closure}(\mathrm{params}, \mathrm{captures}, T_{\mathrm{ret}}, S_{\mathrm{body}}, \rho_c)}{\begin{array}{l}\langle \rho, \sigma, \mu, \texttt{fn } f(\mathrm{params})[\mathrm{captures}] \to T_{\mathrm{ret}} \texttt{ do } S_{\mathrm{body}} \texttt{ end ; } S\rangle \to \\ \langle \rho[f \mapsto \ell], \sigma[\ell \mapsto v], \mu[\ell \mapsto \texttt{const}], S\rangle\end{array}} \quad \text{(St-FnDef)}$$

If

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{if } e \texttt{ do } S_1 \texttt{ else } S_2 \texttt{ end}\rangle \to \langle \rho', \sigma', \mu', \texttt{if } e' \texttt{ do } S_1 \texttt{ else } S_2 \texttt{ end}\rangle} \quad \text{(St-IfStep)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{if true do } S_1 \texttt{ else } S_2 \texttt{ end}\rangle \to \langle \rho, \sigma, \mu, S_1\rangle} \quad \text{(St-IfTrue)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{if false do } S_1 \texttt{ else } S_2 \texttt{ end}\rangle \to \langle \rho, \sigma, \mu, S_2\rangle} \quad \text{(St-IfFalse)}$$

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2\rangle \to \langle \rho', \sigma', \mu', \texttt{if } e' \texttt{ then } e_1 \texttt{ else } e_2\rangle} \quad \text{(E-IfStep)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{if true then } e_1 \texttt{ else } e_2\rangle \to \langle \rho, \sigma, \mu, e_1\rangle} \quad \text{(E-IfTrue)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{if false then } e_1 \texttt{ else } e_2\rangle \to \langle \rho, \sigma, \mu, e_2\rangle} \quad \text{(E-IfFalse)}$$

Match Expression

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{match } e \texttt{ as } x \textit{ cases } \texttt{end}\rangle \to \langle \rho', \sigma', \mu', \texttt{match } e' \texttt{ as } x \textit{ cases } \texttt{end}\rangle} \quad \text{(E-MatchStep)}$$

$$\frac{v \in \mathcal{V} \quad v \text{ matches } p}{\langle \rho, \sigma, \mu, \texttt{match } v \texttt{ as } x \texttt{ case } p \Rightarrow e \textit{ cases } \texttt{end}\rangle \to \langle \rho[x \mapsto v], \sigma, \mu, e\rangle} \quad \text{(E-MatchCase)}$$

$$\frac{\langle \rho[x \mapsto v], \sigma, \mu, e_{cond}\rangle \to \langle \rho', \sigma', \mu', e'_{cond}\rangle}{\begin{array}{l}\langle \rho, \sigma, \mu, \texttt{match } v \texttt{ as } x \texttt{ case if } e_{cond} \Rightarrow e \textit{ cases } \texttt{end}\rangle \to \\ \langle \rho', \sigma', \mu', \texttt{ match } v \texttt{ as } x \texttt{ case if } e'_{cond} \Rightarrow e \textit{ cases } \texttt{end}\rangle\end{array}} \quad \text{(E-MatchIfStep)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{match } v \texttt{ as } x \texttt{ case if true} \Rightarrow e \textit{ cases } \texttt{end}\rangle \to \langle \rho[x \mapsto v], \sigma, \mu, e\rangle} \quad \text{(E-MatchIfTrue)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case if false} \Rightarrow e\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(E-MatchIfFalse)}$$

$$\frac{v \in \mathcal{V} \quad v \text{ does not match } p}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case}\ p \Rightarrow e\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(E-MatchNext)}$$

$$\frac{v \in \mathcal{V} \quad \text{no match for } v}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \ldots\ \texttt{end}\rangle \to \text{error}(\rho, \sigma, \mu, \texttt{MatchFailure})} \quad \text{(E-MatchExhausted)}$$

Match Statement

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{match}\ e\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho', \sigma', \mu', \texttt{match}\ e'\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(St-MatchStep)}$$

$$\frac{v \in \mathcal{V} \quad v \text{ matches } p}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case}\ p\ \texttt{do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho[x \mapsto v], \sigma, \mu, S\rangle} \quad \text{(St-MatchCase)}$$

$$\frac{\langle \rho[x \mapsto v], \sigma, \mu, e_{cond}\rangle \to \langle \rho', \sigma', \mu', e'_{cond}\rangle}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case if}\ e_{cond}\ \texttt{do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho', \sigma', \mu', \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case if}\ e'_{cond}\ \texttt{do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(St-MatchIfStep)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case if true do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho[x \mapsto v], \sigma, \mu, S\rangle} \quad \text{(St-MatchIfTrue)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case if false do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(St-MatchIfFalse)}$$

$$\frac{v \in \mathcal{V} \quad v \text{ does not match } p}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \texttt{case}\ p\ \texttt{do}\ S\ \texttt{end}\ \mathit{cases}\ \texttt{end}\rangle \to \langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\rangle} \quad \text{(St-MatchNext)}$$

$$\frac{v \in \mathcal{V} \quad \text{no match for } v}{\langle \rho, \sigma, \mu, \texttt{match}\ v\ \texttt{as}\ x\ \mathit{cases}\ \texttt{end}\ ;\ S\rangle \to \langle \rho, \sigma, \mu, S\rangle} \quad \text{(St-MatchExhausted)}$$

While

$$\texttt{while}\ e\ \texttt{do}\ S\ \texttt{end} \stackrel{\text{def}}{=} \texttt{if}\ e\ \texttt{do}\ (S\ ;\ \texttt{while}\ e\ \texttt{do}\ S\ \texttt{end})\ \texttt{else skip}$$

For

$$\texttt{for}\ x\ \texttt{in}\ e\ \texttt{do}\ S\ \texttt{end} \stackrel{\text{def}}{=} \texttt{do}\ \texttt{const}\ it = \text{iterator}(e);\ \texttt{while}\ \text{hasNext}(it)\ \texttt{do}\ \texttt{const}\ x = \text{next}(it);\ S\ \texttt{end}\ \texttt{end}$$

Throw

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{throw}\ e\rangle \to \langle \rho', \sigma', \mu', \texttt{throw}\ e'\rangle} \quad \text{(StE-ThrowStep)}$$

$$\frac{v \in \mathcal{V}}{\langle \rho, \sigma, \mu, \texttt{throw}\ v\rangle \to \text{error}(\rho, \sigma, \mu, v)} \quad \text{(StE-Throw)}$$

Try Else

$$\frac{\langle \rho, \sigma, \mu, e\rangle \to \langle \rho', \sigma', \mu', e'\rangle}{\langle \rho, \sigma, \mu, \texttt{try}\ e\ \texttt{else}\ e_{\text{def}}\rangle \to \langle \rho', \sigma', \mu', \texttt{try}\ e'\ \texttt{else}\ e_{\text{def}}\rangle} \quad \text{(E-TryElseStep)}$$

$$\frac{v \in \mathcal{V}}{\langle \rho, \sigma, \mu, \texttt{try}\ v\ \texttt{else}\ e_{\text{def}}\rangle \to \langle \rho, \sigma, \mu, v\rangle} \quad \text{(E-TryElseNoThrow)}$$

$$\frac{}{\text{error}(\rho, \sigma, \mu, \texttt{try}\ \ldots\ \texttt{else} e_{\text{def}}) \to \langle \rho, \sigma, \mu, e_{\text{def}}\rangle} \quad \text{(E-TryElseThrow)}$$

Try Catch

$$\frac{\langle \rho, \sigma, \mu, S\rangle \to \langle \rho', \sigma', \mu', S'\rangle}{\langle \rho, \sigma, \mu, \texttt{try do}\ S\ \texttt{end catch}\ T\ \texttt{as}\ x\ \texttt{do}\ S_c\ \texttt{end}\rangle \to \langle \rho', \sigma', \mu', \texttt{try do}\ S'\ \texttt{end catch}\ T\ \texttt{as}\ x\ \texttt{do}\ S_c\ \texttt{end}\rangle} \quad \text{(St-TryCatch)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{try do skip end catch}\ T\ \texttt{as}\ x\ \texttt{do}\ S_c\ \texttt{end}\ ;\ S\rangle \to \langle \rho, \sigma, \mu, S\rangle} \quad \text{(St-TryCatchNoThrow)}$$

$$\frac{\texttt{typeof}(v) = T}{\langle \rho, \sigma, \mu, \texttt{try do throw}\ v\ \texttt{end catch}\ T\ \texttt{as}\ x\ \texttt{do}\ S_c\ \texttt{end}\rangle \to \langle \rho[x \mapsto v], \sigma, \mu, S_c\rangle} \quad \text{(St-TryCatchThrow)}$$

$$\frac{\text{typeof}(v) \neq T}{\langle \rho, \sigma, \mu, \texttt{try do throw } v \texttt{ end catch } T \texttt{ as } x \texttt{ do } S_c \texttt{ end } \mathit{catches} \rangle \rightarrow \langle \rho, \sigma, \mu, \texttt{try do throw } v \texttt{ end } \mathit{catches} \rangle} \quad \text{(St-TryCatchNext)}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{try do throw } v \texttt{ end} \rangle \rightarrow \langle \rho, \sigma, \mu, \texttt{throw } v \rangle} \quad \text{(St-TryCatchNotCaught)}$$

Misc

$$\frac{\langle \rho, \sigma, \mu, S_1 \rangle \rightarrow \langle \rho', \sigma', \mu', S_1' \rangle}{\langle \rho, \sigma, \mu, S_1 \ ; \ S_2 \rangle \rightarrow \langle \rho', \sigma', \mu', S_1' \ ; \ S_2 \rangle} \quad \text{St-SeqStep}$$

$$\frac{}{\langle \rho, \sigma, \mu, \texttt{skip} \ ; \ S \rangle \rightarrow \langle \rho', \sigma', \mu', S \rangle} \quad \text{St-SeqSkip}$$