

1 Small-step 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** (re-assignable) or **const** (non-reassignable).
 - S : Statement or expression currently being executed.

Values

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

Unary Operations

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

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

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

Binary Operations

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

$$\frac{\langle \rho, \sigma, \mu, e_1 \rangle \rightarrow \langle \rho', \sigma', \mu', e'_1 \rangle}{\langle \rho, \sigma, \mu, (e_1 op_b e_2) \rangle \rightarrow \langle \rho', \sigma', \mu', (e'_1 op_b e_2) \rangle} \quad (\text{E-BINARYOPLEFT})$$

$$\frac{\langle \rho, \sigma, \mu, e_2 \rangle \rightarrow \langle \rho', \sigma', \mu', e'_2 \rangle}{\langle \rho, \sigma, \mu, (v_1 op_b e_2) \rangle \rightarrow \langle \rho', \sigma', \mu', (v_1 op_b e'_2) \rangle} \quad (\text{E-BINARYOPRIGHT})$$

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

Call

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, e(e_1, e_2, \dots, e_n) \rangle \rightarrow \langle \rho', \sigma', \mu', e'(e_1, e_2, \dots, e_n) \rangle} \quad (\text{E-CALLSTEP})$$

$$\frac{\langle \rho, \sigma, \mu, e_i \rangle \rightarrow \langle \rho', \sigma', \mu', e'_i \rangle}{\langle \rho, \sigma, \mu, f(v_1, \dots, v_{i-1}, e_i, \dots, e_n) \rangle \rightarrow \langle \rho', \sigma', \mu', f(v_1, \dots, v_{i-1}, e'_i, \dots, e_n) \rangle} \quad (\text{E-CALLARGS})$$

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

Subscript

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

Variable Definitions

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \text{let } x = e \rangle \rightarrow \langle \rho', \sigma', \mu', \text{let } x = e' \rangle} \quad (\text{ST-LETSTEP})$$

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

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \text{const } x = e \rangle \rightarrow \langle \rho', \sigma', \mu', \text{const } x = e' \rangle} \quad (\text{ST-CONSTSTEP})$$

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

Assignment

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, x = e \rangle \rightarrow \langle \rho', \sigma', \mu', x = e' \rangle} \quad (\text{ST-ASSIGNSTEP})$$

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

Scope

$$\frac{\langle \rho, \sigma, \mu, S \rangle \rightarrow \langle \rho', \sigma', \mu', S' \rangle}{\langle \rho, \sigma, \mu, \text{do } S \text{ end} \rangle \rightarrow \langle \rho', \sigma', \mu', \text{do } S' \text{ end} \rangle} \quad (\text{ST-SCOPE})$$

$$\overline{\langle \rho, \sigma, \mu, \text{do skip end} ; S \rangle \rightarrow \langle \rho, \sigma, \mu, S \rangle} \quad (\text{ST-SCOPEEXIT})$$

Function Definition

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \mathbf{fn} \ f = e \rangle \rightarrow \langle \rho', \sigma', \mu', \mathbf{fn} \ f = e' \rangle} \quad (\text{ST-FNDEFSTEP})$$

$$\frac{\ell \notin \text{dom}(\sigma) \quad v = \text{closure}(\text{params}, \text{captures}, T_{\text{ret}}, S_{\text{body}}, \rho_c)}{\langle \rho, \sigma, \mu, \mathbf{fn} \ f(\text{params})[\text{captures}] \rightarrow T_{\text{ret}} \ \mathbf{do} \ S_{\text{body}} \ \mathbf{end} ; S \rangle \rightarrow \langle \rho[f \mapsto \ell], \sigma[\ell \mapsto v], \mu[\ell \mapsto \mathbf{const}], S \rangle} \quad (\text{ST-FNDEF})$$

If

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \mathbf{if} \ e \ \mathbf{do} \ S_1 \ \mathbf{else} \ S_2 \ \mathbf{end} \rangle \rightarrow \langle \rho', \sigma', \mu', \mathbf{if} \ e' \ \mathbf{do} \ S_1 \ \mathbf{else} \ S_2 \ \mathbf{end} \rangle} \quad (\text{ST-IF})$$

$$\overline{\langle \rho, \sigma, \mu, \mathbf{if} \ \mathbf{true} \ \mathbf{do} \ S_1 \ \mathbf{else} \ S_2 \ \mathbf{end} \rangle \rightarrow \langle \rho, \sigma, \mu, S_1 \rangle} \quad (\text{ST-IFTRUE})$$

$$\overline{\langle \rho, \sigma, \mu, \mathbf{if} \ \mathbf{false} \ \mathbf{do} \ S_1 \ \mathbf{else} \ S_2 \ \mathbf{end} \rangle \rightarrow \langle \rho, \sigma, \mu, S_2 \rangle} \quad (\text{ST-IFFALSE})$$

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

$$\overline{\langle \rho, \sigma, \mu, \mathbf{if} \ \mathbf{true} \ \mathbf{then} \ e_1 \ \mathbf{else} \ e_2 \rangle \rightarrow \langle \rho, \sigma, \mu, e_1 \rangle} \quad (\text{E-IFTRUE})$$

$$\overline{\langle \rho, \sigma, \mu, \mathbf{if} \ \mathbf{false} \ \mathbf{then} \ e_1 \ \mathbf{else} \ e_2 \rangle \rightarrow \langle \rho, \sigma, \mu, e_2 \rangle} \quad (\text{E-IFFALSE})$$

Match Expression

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \mathbf{match} \ e \ \mathbf{as} \ x \ \mathbf{cases} \ \mathbf{end} \rangle \rightarrow \langle \rho', \sigma', \mu', \mathbf{match} \ e' \ \mathbf{as} \ x \ \mathbf{cases} \ \mathbf{end} \rangle} \quad (\text{E-MATCHSTEP})$$

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

$$\frac{\langle \rho[x \mapsto v], \sigma, \mu, e_{\text{cond}} \rangle \rightarrow \langle \rho', \sigma', \mu', e'_{\text{cond}} \rangle}{\langle \rho, \sigma, \mu, \mathbf{match} \ v \ \mathbf{as} \ x \ \mathbf{case} \ \mathbf{if} \ e_{\text{cond}} \Rightarrow e \ \mathbf{cases} \ \mathbf{end} \rangle \rightarrow \langle \rho', \sigma', \mu', \mathbf{match} \ v \ \mathbf{as} \ x \ \mathbf{case} \ \mathbf{if} \ e'_{\text{cond}} \Rightarrow e \ \mathbf{cases} \ \mathbf{end} \rangle} \quad (\text{E-MATCHIFSTEP})$$

$$\overline{\langle \rho, \sigma, \mu, \mathbf{match} \ v \ \mathbf{as} \ x \ \mathbf{case} \ \mathbf{if} \ \mathbf{true} \Rightarrow e \ \mathbf{cases} \ \mathbf{end} \rangle \rightarrow \langle \rho[x \mapsto v], \sigma, \mu, e \rangle} \quad (\text{E-MATCHIFTRUE})$$

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

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

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

Match Statement

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \text{match } e \text{ as } x \text{ cases end} \rangle \rightarrow \langle \rho', \sigma', \mu', \text{match } e' \text{ as } x \text{ cases end} \rangle} \quad (\text{ST-MATCHSTEP})$$

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

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

$$\frac{}{\langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ case if true do } S \text{ end cases end} \rangle \rightarrow \langle \rho[x \mapsto v], \sigma, \mu, S \rangle} \quad (\text{ST-MATCHIFTRUE})$$

$$\frac{}{\langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ case if false do } S \text{ end cases end} \rangle \rightarrow \langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ cases end} \rangle} \quad (\text{ST-MATCHIFFALSE})$$

$$\frac{v \in \mathcal{V} \quad v \text{ does not match } p}{\langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ case } p \text{ do } S \text{ end cases end} \rangle \rightarrow \langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ cases end} \rangle} \quad (\text{ST-MATCHNEXT})$$

$$\frac{v \in \mathcal{V} \quad \text{no match for } v}{\langle \rho, \sigma, \mu, \text{match } v \text{ as } x \text{ cases end} ; S \rangle \rightarrow \langle \rho, \sigma, \mu, S \rangle} \quad (\text{ST-MATCHEXHAUSTED})$$

While

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

For

```

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

```

Throw

$$\frac{\langle \rho, \sigma, \mu, e \rangle \rightarrow \langle \rho', \sigma', \mu', e' \rangle}{\langle \rho, \sigma, \mu, \text{throw } e \rangle \rightarrow \langle \rho', \sigma', \mu', \text{throw } e' \rangle} \quad (\text{ST-THROWSTEP})$$

$$\frac{v \in \mathcal{V}}{\langle \rho, \sigma, \mu, \text{throw } v \rangle \rightarrow \text{error}(\rho, \sigma, \mu, v)} \quad (\text{ST-THROW})$$

Try Else

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

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

$$\frac{}{\text{error}(\rho, \sigma, \mu, \text{try } \dots \text{ else } e_{\text{def}}) \rightarrow \langle \rho, \sigma, \mu, e_{\text{def}} \rangle} \quad (\text{E-TRYELSETHROW})$$

Try Catch

$$\frac{\langle \rho, \sigma, \mu, S \rangle \rightarrow \langle \rho', \sigma', \mu', S' \rangle}{\langle \rho, \sigma, \mu, \text{try do } S \text{ end catch } T \text{ as } x \text{ do } S_c \text{ end} \rangle \rightarrow \langle \rho', \sigma', \mu', \text{try do } S' \text{ end catch } T \text{ as } x \text{ do } S_c \text{ end} \rangle} \quad (\text{ST-TRYCATCH})$$

$$\frac{}{\langle \rho, \sigma, \mu, \text{try do skip end catch } T \text{ as } x \text{ do } S_c \text{ end} ; S \rangle \rightarrow \langle \rho, \sigma, \mu, S \rangle} \quad (\text{ST-TRYCATCHNOTHROW})$$

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

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

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

Misc

$$\frac{\langle \rho, \sigma, \mu, S_1 \rangle \rightarrow \langle \rho', \sigma', \mu', S'_1 \rangle}{\langle \rho, \sigma, \mu, S_1 ; S_2 \rangle \rightarrow \langle \rho', \sigma', \mu', S'_1 ; S_2 \rangle} \quad \text{ST-SEQSTEP}$$

$$\frac{}{\langle \rho, \sigma, \mu, \text{skip} ; S \rangle \rightarrow \langle \rho', \sigma', \mu', S \rangle} \quad \text{ST-SEQSKIP}$$