

x	$::=$	<code>ALPHANUMERIC</code> <code>★ALPHANUMERIC</code>	<i>variables</i>
$\overset{\star}{\ell}$	$::=$	$\ell \mid *$	<i>general labels</i>
ℓ			<i>labels</i>
T	$::=$	$[t, \dots]$	<i>stack</i>
t	$::=$	$\langle \eta, \overset{\star}{\ell}, S \rangle$	<i>stack frames</i>
S	$::=$	$[s, \dots]$	<i>programs</i>
s	$::=$	$\ell : \overset{\star}{\ell} : d$	<i>clauses</i>
d	$::=$	$x = e \mid \text{return } x \mid \text{goto } \ell \mid \text{goto } \ell \text{ if not } x$ $\mid \text{raise } x \mid \text{catch } x \mid \text{pass}$	<i>directives</i>
B	$::=$	$\{x \mapsto m, \dots\}$	<i>bindings</i>
H	$::=$	$\{m \mapsto v, \dots\}$	<i>heap</i>
v	$::=$	$\mathbb{Z} \mid \mathbb{S} \mid [m, \dots] \mid (m, \dots) \mid B \mid F \mid M \mid *$	<i>values</i>
e	$::=$	$\mathbb{Z} \mid \mathbb{S} \mid \text{None} \mid x \mid \text{def } x(x, \dots) = \{S\} \mid x(x, \dots) \mid x.x \mid [x, \dots] \mid (x, \dots)$	<i>expressions</i>
Y	$::=$	$[y, \dots]$	<i>microcode stack</i>
Z	$::=$	$[z, \dots]$	<i>microcode literal stack</i>
y	$::=$	<code>STORE</code> <code>WRAP</code> <code>BIND</code> <code>LOOKUP</code> <code>LIST</code> n <code>TUPLE</code> n <code>ADVANCE</code> <code>POP</code> <code>PUSH</code> S <code>RAISE</code> <code>GOTO</code> ℓ <code>GOTOIFN</code> ℓ <code>CALL</code> n <code>CONVERT</code> n <code>RETRIEVE</code> <code>ALLOCNAMEERROR</code> <code>ALLOCTYPEERROR</code> <code>ALLOCATTRERROR</code>	<i>microcode instructions</i>
z	$::=$	$x \mid m \mid v$	<i>microcode literals</i>
$\overset{\star}{m}$	$::=$	$m \mid \eta \mid *$	<i>general memory locations</i>
η, m	$::=$	<code><ADDRESS></code>	<i>memory locations</i>
F	$::=$	$\langle \eta, \text{def } (x, \dots) \rightarrow S \rangle \mid \mathfrak{F} \mid \mathfrak{M}$	<i>general functions</i>
M	$::=$	$\langle m, F \rangle$	<i>general methods</i>
$\mathfrak{F}, \mathfrak{M}$	$::=$	<code>CallFunc</code> <code>Type</code>	<i>magic functions</i>
n			<i>integers</i>

Figure 1: Expression Grammar

Definition 0.1. *Initialization*

$$\begin{aligned}
H_{\text{INIT}} &= \{\eta_{\text{INIT}} \mapsto B_{\text{INIT}}, m_{\text{None}} \mapsto *, m_{\text{AttrError}} \mapsto \{\}, m_{\text{FuncType}} \mapsto \{\}\} \\
B_{\text{INIT}} &= \{\text{ATTRIBUTEERROR} \mapsto m_{\text{AttrError}}, \text{FUNCTIONTYPE} \mapsto m_{\text{FuncType}}\} \\
t_{\text{INIT}} &= \langle \ell_{\text{INIT}}, S_{\text{INIT}} \rangle \\
T_{\text{INIT}} &= [t_{\text{INIT}}] \\
\text{GETATTRIBUTE} &= \langle \eta, \text{def } (o, a) \rightarrow \$1 = o.a; \rangle \\
\text{GETCALL} &= \langle \eta, \text{def } (f) \rightarrow \rangle
\end{aligned}$$

(*"if \$1 == None then return getattribute(o.__class__,a) else return \$1" - TC*)
(*todo: add builtin mappings $m \mapsto F$ - TC*)

$$\begin{array}{c}
\text{STORE } v \\
\frac{m \notin H \quad H' = H[m \mapsto v]}{Z \parallel [v, \text{STORE}] \parallel Y, T, H \longrightarrow^1 Z \parallel [m] \parallel Y, T, H'} \\
\\
\text{WRAP } m \\
\frac{v = \text{GETOBJ}(H, m)}{Z \parallel [m, \text{WRAP}] \parallel Y, T, H \longrightarrow^1 Z \parallel [v] \parallel Y, T, H} \\
\\
\text{BIND } m \text{ TO } x \\
\frac{B = H[\eta] \quad B' = B[x \mapsto m] \quad H' = H[\eta \mapsto B']}{Z \parallel [m, x, \text{BIND}] \parallel Y, T, H \longrightarrow^1 Z \parallel Y, T, H'} \\
\\
\text{ADVANCE} \\
\frac{S(\ell) = \ell : \ell^{\star'} : d \quad \ell \stackrel{s}{\blacktriangleleft} \ell^{\star''}}{Z \parallel [\text{ADVANCE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Z \parallel Y, [\langle \eta, \ell^{\star''}, S \rangle] \parallel T, H} \\
\\
\text{POP} \\
\frac{}{Z \parallel [\text{POP}] \parallel Y, t \parallel T, H \longrightarrow^1 Z \parallel Y, T, H} \\
\\
\text{PUSH } S \\
\frac{H' = H[\eta' \mapsto \{\star x_{\text{parent}} \mapsto \eta\}], \eta' \notin H \quad S = [\ell : \ell^{\star'} : d, \dots]}{Z \parallel [\eta, \text{PUSH } S] \parallel Y, T, H \longrightarrow^1 Z \parallel Y, [\langle \eta', \ell, S \rangle] \parallel T, H'} \\
\\
\text{LOOK UP } x \text{ (BOUND)} \\
\frac{\text{LOOKUP}(H, \eta, x) = m}{Z \parallel [x, \text{LOOKUP}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Z \parallel [m] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{LOOK UP } x \text{ (NAMEERROR)} \\
\frac{\text{LOOKUP}(H, \eta, x) = *}{Z \parallel [x, \text{LOOKUP}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 [\text{ALLOCNAMEERROR}, \text{RAISE}], [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{MAKE LIST} \\
\frac{v = [m_1, \dots, m_n]}{Z \parallel [m_1, \dots, m_n, \text{LIST } n] \parallel Y, T, H \longrightarrow^1 Z \parallel [v] \parallel Y, T, H} \\
\\
\text{MAKE TUPLE} \\
\frac{v = (m_1, \dots, m_n)}{Z \parallel [m_1, \dots, m_n, \text{TUPLE } n] \parallel Y, T, H \longrightarrow^1 Z \parallel [v] \parallel Y, T, H}
\end{array}$$

Figure 2: Microcommands

$$\begin{array}{c}
\text{RAISE (NO EXCEPTION LABEL)} \\
\frac{S(\ell) = \ell : * : d}{Z \parallel [\text{RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Z \parallel [\text{POP, RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{RAISE (CAUGHT)} \\
\frac{S(\ell) = \ell : \ell_0 : d \quad S(\ell_0) = \ell_0 : \ell_1 : \text{catch } x \quad Y' = [x, \text{BIND, ADVANCE}]}{Z \parallel [\text{RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Z \parallel Y' \parallel Y, [\langle \eta, \ell_0, S \rangle] \parallel T, H} \\
\\
\text{GOTO } \ell \\
\frac{S(\ell) = \ell : \ell^* : d}{Z \parallel [\text{GOTO } \ell] \parallel Y, [\langle \eta, \ell', S \rangle] \parallel T, H \longrightarrow^1 Z \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTOIFN } \ell \text{ (SUCCESS)} \\
\frac{H[m] = \text{FALSE} \quad S(\ell) = \ell : \ell^* : d}{Z \parallel [m, \text{GOTOIFN } \ell] \parallel Y, T, H \longrightarrow^1 Z \parallel [\text{GOTO}] \parallel Y, T, H} \\
\\
\text{GOTOIFN } \ell \text{ (FAILURE)} \\
\frac{H[m] = \text{TRUE}}{Z \parallel [m, \text{GOTOIFN } \ell] \parallel Y, T, H \longrightarrow^1 Z \parallel [\text{ADVANCE}] \parallel Y, T, H} \\
\\
\text{CALL FUNCTION } m \\
\frac{v = \langle \eta, \text{def } (x_1, \dots, x_n) \rightarrow S \rangle \quad Y' = [\eta, \text{PUSH } S, m_1, x_1, \text{BIND}, \dots, m_n, x_n, \text{BIND}]}{Z \parallel [v, m_1, \dots, m_n, \text{CALL } n] \parallel Y, T, H \longrightarrow^1 Z \parallel Y' \parallel Y, T, H} \\
\\
\text{CALL FUNCTION (WRONG ARGS)} \\
\frac{v = \langle \eta, \text{def } (x_1, \dots, x_q) \rightarrow S \rangle \quad q \neq n}{Z \parallel [v, m_1, \dots, m_n, \text{CALL } n] \parallel Y, T, H \longrightarrow^1 [\text{ALLOCTYPEERROR, RAISE}], T, H}
\end{array}$$

Figure 3: Microcommands (cont.)

$$\begin{array}{c}
\text{CONVERT FUNCTION } v \\
\frac{v = F}{Z \parallel [v, m_1, \dots, m_n, \text{CONVERT } n] \parallel Y, T, H \longrightarrow^1 Z \parallel [v, m_1, \dots, m_n, \text{CALL } n] \parallel Y, T, H} \\
\\
\text{CONVERT METHOD } v \\
\frac{v = \langle m_0, F \rangle \quad v' = F}{Z \parallel [v, m_1, \dots, m_n, \text{CONVERT } n] \parallel Y, T, H \longrightarrow^1 Z \parallel [v', m_0, m_1, \dots, m_n, \text{CALL } n + 1] \parallel Y, T, H} \\
\\
\begin{array}{c}
\text{RETRIEVE } x \\
\frac{v = H[m][x] \quad Y' = [v, \text{STORE}]}{Z \parallel [m, x, \text{RETRIEVE}] \parallel Y, T, H \longrightarrow^1 Z \parallel Y' \parallel Y, T, H}
\end{array} \\
\\
\begin{array}{c}
\text{RETRIEVE } x \text{ (ATTRIBUTEERROR)} \\
\frac{* = H[m][x]}{Z \parallel [m, x, \text{RETRIEVE}] \parallel Y, T, H \longrightarrow^1 [\text{ALLOCATTRERROR}, \text{RAISE}], T, H}
\end{array}
\end{array}$$

Figure 4: Microcommands (cont.)

Figure 5: Magic Functions

NONE ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = \mathbf{None} \quad Y = [m_{\mathbf{None}}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

LITERAL ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = e, \ e \text{ is of form } \mathbb{Z}, \mathbb{S} \quad Y = [v, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

(TODO: make literal category (ints, str, bool, None) – TC)

NAME ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x_1 = x_2 \quad Y = [x_2, \text{LOOKUP}, x_1, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

LIST ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = [x_1, \dots, x_n] \quad Y = [(x_1, \text{LOOKUP}), \dots, (x_n, \text{LOOKUP}), \text{LIST } n, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

(Parentheses in Y group instructions together for convenience of reading. – TC)

TUPLE ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = [x_1, \dots, x_n] \quad Y = [(x_1, \text{LOOKUP}), \dots, (x_n, \text{LOOKUP}), \text{TUPLE } n, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

FUNCTIONDEF ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell' : x = \mathbf{def} (x_1, \dots, x_n) = \{S'\} \quad v = \langle \eta, \mathbf{def} (x_1, \dots, x_n) \rightarrow S' \rangle \quad Y = [v, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

ATTRIBUTE ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell' : x = x_1.x_2 \quad Y = [x_1, \text{LOOKUP}, x_2, \text{RETRIEVE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

CALL ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell' : x = x_0(x_1, \dots, x_n) \quad Y = [x_0, \text{LOOKUP}, \dots, x_n, \text{LOOKUP}, \text{CONVERT } n]}{[\], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

Figure 6: Operational Semantics: Assignment

$$\begin{array}{c}
\text{PASS} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{pass} \quad Y = [\text{ADVANCE}]}{[], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{RETURN} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{return } x \quad T = [\langle \eta', \ell'', S' \rangle] \parallel T' \quad S(\ell'') = \ell'' : \ell^{\star'''} : x' = e \quad Y = [x, \text{LOOKUP}, \text{POP}, x', \text{BIND}, \text{ADVANCE}]}{[], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTO} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{goto } \ell'' \quad Y = [\text{GOTO } \ell'']}{[], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTOIFNOT} \\
\frac{S(\ell) = \ell : \ell' : \text{goto } \ell'' \text{ if not } x \quad Y = [x, \text{LOOKUP}, \text{GOTOIFN } \ell'']}{[], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{END OF FUNCTION} \\
\frac{t = \langle \eta', \ell, S' \rangle \quad S(\ell) = \ell : \ell^{\star'} : x = e \quad Y = [\text{POP}, m_{\text{None}}, x, \text{BIND}, \text{ADVANCE}]}{[], [\langle \eta, *, S \rangle, t] \parallel T, H \longrightarrow^1 Y, [\langle \eta, *, S \rangle, t] \parallel T, H} \\
\\
(m_{\text{None}} \text{ is a memory location reserved for None. - TC}) \\
\\
\text{END OF PROGRAM} \\
\frac{T = [\langle \eta, *, S \rangle] \quad Y = [\text{POP}]}{[], T, H \longrightarrow^1 Y, T, H}
\end{array}$$

Figure 7: Operational Semantics: Flow

Definition 0.2.

$$\text{LOOKUP}(H, \eta, x) = (\text{todo} - \text{TC})$$

Definition 0.3.

$$H[m] = v, B_{obj} = \{\star x_{value} \mapsto v, \text{--getattribute--} \mapsto \text{Getattribute}\}$$

$$\text{GETOBJ}(H, m) = \begin{cases} B, & \text{if } v = B \\ B = B_{obj}[\text{--class--} \mapsto \star \text{FUNTYPE}], & \text{if } v = F \end{cases} \quad (1)$$

(getobj takes (H, m, memory location of getattribute) - TC)

Figure 8: Helper Functions