x	$::=$	$\textsc{alphanumeric} \mid \star\textsc{alphanumeric}$	*variables*
$\overset{\star}{\ell}$	$::=$	$\ell \mid *$	*general labels*
ℓ			*labels*
T	$::=$	$[t, \ldots]$	*stack*
t	$::=$	$\langle \eta, \overset{\star}{\ell}, S \rangle$	*stack frames*
S	$::=$	$[s, \ldots]$	*programs*
s	$::=$	$\ell : \overset{\star}{\ell} : d$	*clauses*
d	$::=$	$x = e \mid \texttt{return}\ x \mid \texttt{goto}\ \ell \mid \texttt{goto}\ \ell\ \texttt{if not}\ x$	*directives*
		$\mid \texttt{raise}\ x \mid \texttt{catch}\ x \mid \texttt{pass}$	
B	$::=$	$\{x \mapsto m, \ldots\}$	*bindings*
H	$::=$	$\{m \mapsto v, \ldots\}$	*heap*
v	$::=$	$\mathbb{Z} \mid [m, \ldots] \mid (m, \ldots) \mid B \mid F \mid M \mid *$	*values*
e	$::=$	$\mathbb{Z} \mid \texttt{None} \mid x \mid \texttt{def}\ x(x, \ldots) = \{S\} \mid x(x, \ldots) \mid x.x \mid [x, \ldots] \mid (x, \ldots)$	*expressions*
Y	$::=$	$[y, \ldots]$	*microcode stack*
Z	$::=$	$[z, \ldots]$	*microcode literal stack*
y	$::=$	$\textsc{Store} \mid \textsc{Wrap} \mid \textsc{Bind} \mid \textsc{LookUp} \mid \textsc{List}\ n \mid \textsc{Tuple}\ n$	*microcode instructions*
		$\mid \textsc{Advance} \mid \textsc{Pop} \mid \textsc{Push}\ S \mid \textsc{Raise} \mid \textsc{Goto}\ \ell \mid \textsc{GotoIfN}\ \ell$	
		$\mid \textsc{Call}\ n \mid \textsc{GetCall}\ n \mid \textsc{Convert}\ n \mid \textsc{Retrieve}$	
		$\mid \textsc{AllocNameError} \mid \textsc{AllocTypeError} \mid \textsc{AllocAttrError}$	
z	$::=$	$x \mid m \mid v$	*microcode literals*
P	$::=$	$m \mapsto m$	*parental map*
$\overset{\star}{m}$	$::=$	$m \mid \eta \mid *$	*general memory locations*
η, m	$::=$	$<\textsc{address}>$	*memory locations*
F	$::=$	$\langle \eta,\ \texttt{def}\ (x, \ldots) \to S \rangle \mid \mathfrak{F} \mid \mathfrak{M}$	*general functions*
M	$::=$	$\langle m, F \rangle$	*general methods*
$\mathfrak{F}, \mathfrak{M}$	$::=$	$\mathfrak{GetAttribute}$	*magic functions*
n			*integers*

Figure 1: Expression Grammar

Definition 0.1. *Initialization*

$$H_{\textsc{Init}} = \{\eta_{\textsc{Init}} \mapsto B_{\textsc{Init}}, m_{\texttt{None}} \mapsto *, m_{\texttt{AttrError}} \mapsto \{\ \}, m_{\texttt{FunType}} \mapsto \{\ \}\}$$
$$B_{\textsc{Init}} = \{\textsc{AttributeError} \mapsto m_{\texttt{AttrError}}, \textsc{FunctionType} \mapsto m_{\texttt{FunType}}\}$$
$$t_{\textsc{Init}} = \langle \ell_{\textsc{Init}}, S_{\textsc{Init}} \rangle$$
$$T_{\textsc{Init}} = [t_{\textsc{Init}}]$$
$$P_{\textsc{Init}} = \{\ \}$$

(todo: add builtin mappings $m \mapsto F$ – TC)

Store v

$$\frac{m \notin H \qquad H' = H[m \mapsto v]}{P, Z \,||\, [v, \textsc{Store}] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [m] \,||\, Y, T, H'}$$

Wrap m

$$\frac{v = \textsc{GetObj}(H, m)}{P, Z \,||\, [m, \textsc{Wrap}] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [v] \,||\, Y, T, H}$$

Bind m to x

$$\frac{B = H[\eta] \qquad B' = B[x \mapsto m] \qquad H' = H[\eta \mapsto B']}{P, Z \,||\, [m, x, \textsc{Bind}] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, Y, T, H'}$$

Advance

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : d \qquad \ell \overset{s}{\blacktriangleleft} \overset{\star''}{\ell}}{P, Z \,||\, [\textsc{Advance}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 P, Z \,||\, Y, [\langle \eta, \overset{\star''}{\ell}, S \rangle] \,||\, T, H}$$

Pop

$$\frac{}{P, Z \,||\, [\textsc{Pop}] \,||\, Y, t \,||\, T, H \longrightarrow^1 P, Z \,||\, Y, T, H}$$

Push S

$$\frac{P' = P[\eta' \mapsto \eta], \eta' \notin P \qquad S = [\ell : \overset{\star'}{\ell} : d, \ldots]}{P, Z \,||\, [\eta, \textsc{Push}\ S] \,||\, Y, T, H \longrightarrow^1 P', Z \,||\, Y, [\langle \eta', \ell, S \rangle] \,||\, T, H}$$

Look up x (bound)

$$\frac{\textsc{Lookup}(P, H, \eta, x) = m}{P, Z \,||\, [x, \textsc{LookUp}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 P, Z \,||\, [m] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

Look up x (NameError)

$$\frac{\textsc{Lookup}(P, H, \eta, x) = *}{P, Z \,||\, [x, \textsc{LookUp}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 P, [\textsc{AllocNameError}, \textsc{Raise}], [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

Make List

$$\frac{v = [m_1, \ldots, m_n]}{P, Z \,||\, [m_1, \ldots, m_n, \textsc{List}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [v] \,||\, Y, T, H}$$

Make Tuple

$$\frac{v = (m_1, \ldots, m_n)}{P, Z \,||\, [m_1, \ldots, m_n, \textsc{Tuple}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [v] \,||\, Y, T, H}$$

Figure 2: Microcommands

Raise (no exception label)

$$\frac{S(\ell) = \ell : * : d}{P, Z \,||\, [\textsc{Raise}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 P, Z \,||\, [\textsc{Pop}, \textsc{Raise}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

Raise (caught)

$$\frac{S(\ell) = \ell : \ell_0 : d \qquad S(\ell_0) = \ell_0 : \ell_1 : \ \texttt{catch}\, x \qquad Y' = [x, \textsc{Bind}, \textsc{Advance}]}{P, Z \,||\, [\textsc{Raise}] \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 P, Z \,||\, Y' \,||\, Y, [\langle \eta, \ell_0, S \rangle] \,||\, T, H}$$

Goto ℓ

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : d}{P, Z \,||\, [\textsc{Goto}\ \ell] \,||\, Y, [\langle \eta, \ell', S \rangle] \,||\, T, H \longrightarrow^1 P, Z \,||\, Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

GotoIfN ℓ (success)

$$\frac{H[m] = \textsc{False} \qquad S(\ell) = \ell : \overset{\star'}{\ell} : d}{P, Z \,||\, [m, \textsc{GotoIfN}\ \ell] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [\textsc{Goto}] \,||\, Y, T, H}$$

GotoIfN ℓ (failure)

$$\frac{H[m] = \textsc{True}}{P, Z \,||\, [m, \textsc{GotoIfN}\ \ell] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [\textsc{Advance}] \,||\, Y, T, H}$$

Call function m

$$\frac{\begin{array}{c} v = \langle \eta,\ \texttt{def}\ (x_1, \ldots, x_n) \to S \rangle \\ Y' = [\eta, \textsc{Push}\ S, m_1, x_1, \textsc{Bind}, \ldots, m_n, x_n, \textsc{Bind}] \end{array}}{P, Z \,||\, [v, m_1, \ldots, m_n, \textsc{Call}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, Y' \,||\, Y, T, H}$$

Call function (wrong args)

$$\frac{v = \langle \eta,\ \texttt{def}\ (x_1, \ldots, x_q) \to S \rangle q \neq n}{P, Z \,||\, [v, m_1, \ldots, m_n, \textsc{Call}\ n] \,||\, Y, T, H \longrightarrow^1 P, [\textsc{AllocTypeError}, \textsc{Raise}], T, H}$$

Get Call m

$$\frac{v = H[m_0],\ v \text{ is of form } F \text{ or } M \qquad Y' = [v, m_1, \ldots, m_n]}{P, Z \,||\, [m_0, \ldots, m_n, \textsc{GetCall}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, Y' \,||\, Y, T, H}$$

Get Call (TypeError)

$$\frac{v = H[m_0],\ v \text{ is not of form } F \text{ or } M}{P, Z \,||\, [m_0, \ldots, m_n, \textsc{GetCall}\ n] \,||\, Y, T, H \longrightarrow^1 P, [\textsc{AllocTypeError}, \textsc{Raise}], T, H}$$

Figure 3: Microcommands (cont.)

Convert Function v

$$\frac{v = F}{P, Z \,||\, [v, m_1, \ldots, m_n, \textsc{Convert}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [v, m_1, \ldots, m_n, \textsc{Call}\ n] \,||\, Y, T, H}$$

Convert Method v

$$\frac{v = \langle m_0, F\rangle \qquad v' = F}{P, Z \,||\, [v, m_1, \ldots, m_n, \textsc{Convert}\ n] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, [v', m_0, m_1, \ldots, m_n, \textsc{Call}\ n+1] \,||\, Y, T, H}$$

Retrieve x

$$\frac{v = H[m] \qquad Y' = [v, \textsc{Store}]}{P, Z \,||\, [m, x, \textsc{Retrieve}] \,||\, Y, T, H \longrightarrow^1 P, Z \,||\, Y' \,||\, Y, T, H}$$

Retrieve x (AttributeError)

$$\frac{* = H[m]}{P, Z \,||\, [m, x, \textsc{Retrieve}] \,||\, Y, T, H \longrightarrow^1 P, [\textsc{AllocAttrError}, \textsc{Raise}], T, H}$$

Figure 4: Microcommands (cont.)

None Assignment

$$\frac{S(\ell) = \ell : \overset{\star\prime}{\ell} : x = \texttt{None} \qquad Y = [m_{\texttt{None}}, x, \textsc{Bind}, \textsc{Advance}]}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Literal Assignment

$$\frac{S(\ell) = \ell : \overset{\star\prime}{\ell} : x = \mathbb{Z} \qquad Y = [v, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}]}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

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

Name Assignment

$$\frac{S(\ell) = \ell : \overset{\star\prime}{\ell} : x_1 = x_2 \qquad Y = [x_2, \textsc{LookUp}, x_1, \textsc{Bind}, \textsc{Advance}]}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

List Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \overset{\star\prime}{\ell} : x = [x_1, \ldots, x_n] \\ Y = [(x_1, \textsc{LookUp}), \ldots, (x_n, \textsc{LookUp}), \textsc{List}\ n, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

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

Tuple Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \overset{\star\prime}{\ell} : x = [x_1, \ldots, x_n] \\ Y = [(x_1, \textsc{LookUp}), \ldots, (x_n, \textsc{LookUp}), \textsc{Tuple}\ n, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

FunctionDef Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = \ \texttt{def}\ (x_1, \ldots, x_n) = \{S'\} \qquad v = \langle \eta,\ \texttt{def}\ (x_1, \ldots, x_n) \to S' \rangle \\ Y = [v, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Attribute Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = x_1.x_2 \\ Y = [x_1, \textsc{LookUp}, x_2, \textsc{Retrieve}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Call Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = x_0(x_1, \ldots, x_n) \\ Y = [x_0, \textsc{LookUp}, \ldots, x_n, \textsc{LookUp}, \textsc{GetCall}\ n, \textsc{Convert}\ n] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Figure 5: Operational Semantics: Assignment

Pass

$$\frac{S(\ell) = \ell : \overset{\star\prime}{\ell} \ : \ \texttt{pass} \qquad Y = [\textsc{Advance}]}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Return

$$\frac{\begin{array}{c} S(\ell) = \ell : \overset{\star\prime}{\ell} \ : \ \texttt{return}\ x \qquad T = [\langle \eta', \ell'', S'\rangle] \,||\, T' \\ S(\ell'') = \ell'' : \overset{\star\prime\prime\prime}{\ell} \ : x' = e \qquad Y = [x, \textsc{LookUp}, \textsc{Pop}, x', \textsc{Bind}, \textsc{Advance}] \end{array}}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

Goto

$$\frac{S(\ell) = \ell : \overset{\star\prime}{\ell} \ : \ \texttt{goto}\ \ell'' \qquad Y = [\textsc{Goto}\ \ell'']}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

GotoIfNot

$$\frac{S(\ell) = \ell : \ell' : \ \texttt{goto}\ \ell''\ \texttt{if not}\ x \qquad Y = [x, \textsc{Lookup}, \textsc{GotoIfN}\ \ell'']}{P, [\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

End of Function

$$\frac{t = \langle \eta', \ell, S'\rangle \qquad S(\ell) = \ell : \overset{\star\prime}{\ell} \ : x = e \qquad Y = [\textsc{Pop}, m_{\texttt{None}}, x, \textsc{Bind}, \textsc{Advance}]}{P, [\,], [\langle \eta, *, S\rangle, t] \,||\, T, H \longrightarrow^{1} P, Y, [\langle \eta, *, S\rangle, t] \,||\, T, H}$$

($m_{\texttt{None}}$ is a memory location reserved for `None`. – TC)

End of Program

$$\frac{T = [\langle \eta, *, S\rangle] \qquad Y = [\textsc{Pop}]}{P, [\,], T, H \longrightarrow^{1} P, Y, T, H}$$

Figure 6: Operational Semantics: Flow

Definition 0.2.

$$\textsc{Lookup}(P, H, \eta, x) = \textit{(todo – TC)}$$

Definition 0.3.

$$H[m] = v, B_{obj} = \{\star x_{value} \mapsto v, \_\_getattribute\_\_ \mapsto \mathfrak{Getattribute}\}$$

$$\textsc{GetObj}(H, m) = \begin{cases} B, & \textit{if } v = B \\ B = B_{obj}[\_\_class\_\_ \mapsto \star\textsc{funType}], & \textit{if } v = F \end{cases} \tag{1}$$

Definition 0.4.

$$H[m] = B, B[\star x_{value}] = v$$

$$\textsc{GetCall}(H, m) = \begin{cases} v, & \textit{if } v = F \mid M \\ *, & \textit{otherwise} \end{cases} \tag{2}$$

Figure 7: Helper Functions