$$
\begin{array}{rcll}
x & ::= & \textsc{alphanumeric} \mid \star\textsc{alphanumeric} & \textit{variables} \\
\overset{\star}{\ell} & ::= & \ell \mid * & \textit{general labels} \\
\ell & & & \textit{labels} \\
T & ::= & [t, \ldots] & \textit{stack} \\
t & ::= & \langle \eta, \overset{\star}{\ell}, S \rangle & \textit{stack frames} \\
S & ::= & [s, \ldots] & \textit{programs} \\
s & ::= & \ell : \overset{\star}{\ell} : d & \textit{clauses} \\
d & ::= & x = e \mid \texttt{return}\ x \mid \texttt{goto}\ \ell \mid \texttt{goto}\ \ell\ \texttt{if not}\ x & \textit{directives} \\
 & & \mid \texttt{raise}\ x \mid \texttt{catch}\ x \mid \texttt{pass} & \\
B & ::= & \{x \mapsto m, \ldots\} & \textit{bindings} \\
H & ::= & \{m \mapsto v, \ldots\} & \textit{heap} \\
v & ::= & \mathbb{Z} \mid \mathbb{S} \mid [m, \ldots] \mid (m, \ldots) \mid B \mid F \mid M \mid * & \textit{values} \\
e & ::= & \mathbb{Z} \mid \mathbb{S} \mid x \mid \texttt{def}\ x(x, \ldots) = \{S\} \mid x(x, \ldots) \mid x.x \mid [x, \ldots] \mid (x, \ldots) & \textit{expressions} \\
Y & ::= & [y, \ldots] & \textit{microcode stack} \\
Z & ::= & [z, \ldots] & \textit{microcode literal stack} \\
y & ::= & \textsc{Store} \mid \textsc{Wrap} \mid \textsc{Bind} \mid \textsc{LookUp} \mid \textsc{List}\ n \mid \textsc{Tuple}\ n & \textit{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{Convert}\ n \mid \textsc{Retrieve} & \\
 & & \mid \textsc{AllocNameError} \mid \textsc{AllocTypeError} \mid \textsc{AllocAttrError} & \\
z & ::= & x \mid m \mid v & \textit{microcode literals} \\
\overset{\star}{m} & ::= & m \mid \eta \mid * & \textit{general memory locations} \\
\eta, m & ::= & \textsc{<address>} & \textit{memory locations} \\
F & ::= & \langle \eta,\ \texttt{def}\ (x, \ldots) \to S \rangle \mid \mathfrak{F} \mid \mathfrak{M} & \textit{general functions} \\
M & ::= & \langle m, F \rangle & \textit{general methods} \\
\mathfrak{F}, \mathfrak{M} & ::= & \mathfrak{CallFunc} \mid \mathfrak{Type} & \textit{magic functions} \\
n & & & \textit{integers}
\end{array}
$$

Figure 1: Expression Grammar

Definition 0.1. *Initialization*

$$
\begin{aligned}
H_{\textsc{Init}} &= \{\eta_{\textsc{Init}} \mapsto B_{\textsc{Init}}, m_{\texttt{None}} \mapsto \{\ \}, m_{\texttt{AttrError}} \mapsto \{\ \}, m_{\texttt{FunType}} \mapsto \{\ \}\} \\
B_{\textsc{Init}} &= \{\star\texttt{None} \mapsto m_{\texttt{None}}, \textsc{AttributeError} \mapsto m_{\texttt{AttrError}}, \textsc{FunctionType} \mapsto m_{\texttt{FunType}}\} \\
t_{\textsc{Init}} &= \langle \ell_{\textsc{Init}}, S_{\textsc{Init}} \rangle \\
T_{\textsc{Init}} &= [t_{\textsc{Init}}] \\
\textsc{GetAttribute} &= \langle \eta,\ \texttt{def}\ (o, a) \to \$1 = o.a; \rangle \\
\textsc{GetCall} &= \langle \eta,\ \texttt{def}\ (f) \to \rangle
\end{aligned}
$$

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

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

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

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

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

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

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

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

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

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

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

Figure 2: Microcommands

RAISE (NO EXCEPTION LABEL)

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

RAISE (CAUGHT)

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

GOTO ℓ

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

GOTOIFN ℓ (SUCCESS)

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

GOTOIFN ℓ (FAILURE)

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

CALL FUNCTION m

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

CALL FUNCTION (WRONG ARGS)

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

Figure 3: Microcommands (cont.)

Convert Function v

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

Convert Method v

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

Retrieve x

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

Retrieve x (AttributeError)

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

Figure 4: Microcommands (cont.)

Figure 5: Magic Functions

LITERAL ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = e,\ e \text{ is of form } \mathbb{Z}, \mathbb{S} \qquad Y = [v, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}]}{[\,], [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

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

NAME ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x_1 = x_2 \qquad Y = [x_2, \textsc{Lookup}, x_1, \textsc{Bind}, \textsc{Advance}]}{[\,], [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

LIST ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = [x_1, \ldots, x_n] \qquad Y = [(x_1, \textsc{LookUp}), \ldots, (x_n, \textsc{LookUp}), \textsc{List}\ n, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}]}{[\,], [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

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

TUPLE ASSIGNMENT

$$\frac{S(\ell) = \ell : \overset{\star'}{\ell} : x = [x_1, \ldots, x_n] \qquad Y = [(x_1, \textsc{LookUp}), \ldots, (x_n, \textsc{LookUp}), \textsc{Tuple}\ n, \textsc{Store}, \textsc{Wrap}, \textsc{Store}, x, \textsc{Bind}, \textsc{Advance}]}{[\,], [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

FUNCTIONDEF ASSIGNMENT

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

ATTRIBUTE ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell' : x = x_1.x_2 \qquad Y = [x_1, \textsc{LookUp}, x_2, \textsc{Retrieve}, x, \textsc{Bind}, \textsc{Advance}]}{[\,], [\langle \eta, \ell, S \rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S \rangle] \,||\, T, H}$$

CALL ASSIGNMENT

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

Figure 6: Operational Semantics: Assignment

$$\textsc{Pass}\quad \frac{S(\ell) = \ell : \overset{\star'}{\ell} : \texttt{pass} \qquad Y = [\textsc{Advance}]}{[\,], [\langle \eta, \ell, S\rangle] \,||\, T, H \longrightarrow^1 Y, [\langle \eta, \ell, S\rangle] \,||\, T, H}$$

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

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

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

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

(m_{None} is a memory location reserved for None. – TC)

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

Figure 7: Operational Semantics: Flow

Definition 0.2.

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

Definition 0.3.

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

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

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

Figure 8: Helper Functions