# 1 CoPylot

## 1.1 Grammar

The grammar of the language to be analyzed appears in Figure 2.

We assume throughout the rest of this document that a fixed program $\hat{S}$ is under analysis. *(TODO: describe here the idea of a bijection between labels and statements in this fixed program. – ZP)*

## 1.2 Control Flow

The grammar of control flow graphs appears in Figure 3. *(Discuss construction of initial graph. – ZP)*

We write $\hat{o} \overset{?}{\blacktriangleleft} \hat{o}'$ to denote $(\hat{o} \blacktriangleleft \hat{o}' \in \hat{G}$ when $\hat{G}$ is understood from context. Likewise, we write $\hat{o} \overset{?}{\ll} \hat{o}'$ to denote $(\hat{o} \ll \hat{o}' \in \hat{G}$ when $\hat{G}$ is understood from context.

We define a relation $\longrightarrow^1$ to perform control flow graph closure.

**Definition 1.7.** *Let $\hat{G} \longrightarrow^1 \hat{G}'$ be the least relation satisfying the rules appearing in Figure 4. Throughout these rules, the predicates $\overset{?}{\ll}$ and $\overset{?}{\blacktriangleleft}$ refer to graph $\hat{G}$.*

## 1.3 Value Lookup

The value lookup function uses the additional grammar in Figure 5.

**Definition 1.8.** *Given a control-flow graph $\hat{G}$, let $\hat{G}(\hat{o}_0, \hat{K})$ be the function returning the least set $\hat{V}$ which satisfies the following conditions:*

1. ***Value Manipulation***

   (a) $\boxed{\text{RESULT}}$
   *If $\hat{K} = [\hat{v}]$, then $\hat{v} \in \hat{V}$.*

2. ***Variable Lookup***

   (a) $\boxed{\text{VALUE DISCOVERY}}$
   *If $\hat{o}_1 \overset{?}{\ll} \hat{o}_0$, $\hat{o}_1 = \hat{\ell}_1 : \hat{\ell}_2 : \hat{x} = \hat{v}$, and $\hat{K} = [\hat{x}] \,||\, \hat{K}'$, then $\hat{G}(\hat{o}_1, [\hat{v}] \,||\, \hat{K}') \subseteq \hat{V}$.*

   (b) $\boxed{\text{VALUE SKIP}}$
   *If $\hat{o}_1 \overset{?}{\ll} \hat{o}_0$, $\hat{o}_1 = \hat{\ell}_1 : \hat{\ell}_2 : \hat{x}' = \hat{v}$, $\hat{K} = [\hat{x}] \,||\, \hat{K}'$, and $\hat{x} \neq \hat{x}'$, then $\hat{G}(\hat{o}_1, \hat{K}) \subseteq \hat{V}$.*

   (c) $\boxed{\text{VALUE ALIASING}}$
   *If $\hat{o}_1 \overset{?}{\ll} \hat{o}_0$, $\hat{o}_1 = \hat{\ell}_1 : \hat{\ell}_2 : \hat{x} = \hat{x}'$, and $\hat{K} = [\hat{x}] \,||\, \hat{K}'$, then $\hat{G}(\hat{o}_1, [\hat{x}'] \,||\, \hat{K}) \subseteq \hat{V}$.*

$$
\begin{array}{llll}
x & ::= & \textsc{AlphaNumeric} \mid \star\textsc{AlphaNumeric} & \textit{variables} \\
\overset{*}{\ell} & ::= & \ell \mid \textsc{End} & \textit{labels} \\
T & ::= & [t, \ldots] & \textit{stack} \\
t & ::= & \overset{*}{\ell} \times S & \textit{stack frames} \\
S & ::= & [s, \ldots] & \textit{programs} \\
d & ::= & x \texttt{=} e \mid \texttt{def}\ x(x, \ldots) \texttt{=} \{S\} \mid \texttt{return}\ x \mid \texttt{goto}\ \ell \mid \texttt{goto}\ \ell\ \texttt{if}\ \texttt{not}\ x & \textit{directives} \\
 & & \mid \texttt{raise}\ x \mid \texttt{catch}\ x & \\
B & ::= & \{x \mapsto m, \ldots\} & \textit{bindings} \\
H & ::= & \{m \mapsto v, \ldots\} & \textit{heap} \\
v & ::= & \mathbb{Z} \mid \langle m,\ \texttt{def}\ (x) \to S\rangle \mid \langle m, m,\ \texttt{def}\ (x) \to S\rangle \mid [m, \ldots] \mid (m, \ldots) & \textit{values} \\
 & & \mid B \mid \mathfrak{F} \mid \langle m, \mathfrak{M}\rangle \mid \texttt{undefined} \mid \texttt{None} & \\
e & ::= & v \mid x \mid x(x, \ldots) \mid [x, \ldots] \mid (x, \ldots) & \textit{expressions} \\
P & ::= & m \mapsto m & \textit{parental map} \\
m & & & \textit{memory locations} \\
\mathfrak{F} & & & \textit{magic functions} \\
\mathfrak{M} & & & \textit{magic methods}
\end{array}
$$

**Definition 1.1.**

$$\textsc{Bind}(H, m_0, x, m) = H' \text{ such that } B = H[m_0], B' = B[x \mapsto m], H' = H[m_0 \mapsto B']$$

**Definition 1.2.**

$$\textsc{Bind}(H, m_0, x_1, m_1, \ldots, x_n, m_n) \\ = \textsc{Bind}(\ldots((\textsc{Bind}((\textsc{Bind}(H, m_0, x_1, m_1)), m_0, x_2, m_2), \ldots), m_0, x_n, m_n)$$

**Definition 1.3.**

$$\textsc{Lookup}(m_0, P, H, x_2) =$$

**Definition 1.4.** $\ell : \ell' : d \in S, \ell' : \ell'' : d' \in S, m' = P[m]$

$$\textsc{Catch}(\langle P, \ell, S, m\rangle) = \begin{cases} \ell', x, m' & \text{if } d' = \ \texttt{catch}\ x \\ \texttt{undefined}\ , m' & \text{otherwise} \end{cases} \tag{1}$$

**Definition 1.5.**

$$\textsc{GetObj}(H, m) = \begin{cases} B, & \text{if } v = B \\ B[\star x_{\texttt{value}} \mapsto v], & \text{otherwise} \end{cases}, H[m] = v \tag{2}$$

**Definition 1.6.** $H[m][\_\_call\_\_] = m', H[m'] = v$

$$\textsc{GetCall}(H, m) = \begin{cases} \textsc{GetCall}(H, m'), & \text{if } \ v = B \\ m', & \text{if } v = \langle m_0,\ \texttt{def}\ (x_1, \ldots, x_n) \to \\ & S\rangle \mid \langle m_0, m_{obj},\ \texttt{def}\ (x_1, \ldots, x_n) \to \\ & S\rangle \mid \langle \mathfrak{F}\rangle \mid \langle m_{obj}, \mathfrak{M}\rangle \end{cases} \tag{3}$$

Figure 1: Operational Semantics

Literal Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = v \\ B_{\texttt{obj}} = \{\star x_{\texttt{value}} \mapsto m\} \qquad H' = H[m \mapsto v, m' \mapsto B_{\texttt{obj}}] \\ m, m' \notin H \qquad \textsc{Bind}(H', m_0, x, m') = H'' \qquad \ell \overset{s}{\blacktriangleleft} \overset{*}{\ell}'' \end{array}}{[\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}'', S\rangle] \,||\, T, H'', P, m_0}$$

Name Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x_1 = v_2 \\ m = \textsc{Lookup}(m_0, P, H, x_2) \qquad \textsc{Bind}(H, m_0, x_1, m) = H' \qquad \ell \overset{s}{\blacktriangleleft} \overset{*}{\ell}'' \end{array}}{[\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}'', S\rangle] \,||\, T, H', P, m_0}$$

List Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = [x_1, \ldots, x_n] \qquad \forall i \in \{1, \ldots, n\}, m_i = \textsc{Lookup}(m_0, P, H, x_i) \\ v = [m_1, \ldots, m_n] \qquad B_{\texttt{obj}} = \{\star x_{\texttt{value}} \mapsto m\} \qquad H' = H[m \mapsto v, m' \mapsto B_{\texttt{obj}}] \\ m, m' \notin H \qquad \textsc{Bind}(H', m_0, x, m') = H'' \qquad \ell \overset{s}{\blacktriangleleft} \overset{*}{\ell}'' \end{array}}{\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}'', S\rangle] \,||\, T, H'', P, m_0}$$

Tuple Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x = (x_1, \ldots, x_n) \qquad \forall i \in \{1, \ldots, n\}, m_i = \textsc{Lookup}(m_0, P, H, x_i) \\ v = (m_1, \ldots, m_n) \qquad B_{\texttt{obj}} = \{\star x_{\texttt{value}} \mapsto m\} \qquad H' = H[m \mapsto v, m' \mapsto B_{\texttt{obj}}] \\ m, m' \notin H \qquad \textsc{Bind}(H', m_0, x, m') = H'' \qquad \ell \overset{s}{\blacktriangleleft} \overset{*}{\ell}'' \end{array}}{\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}'', S\rangle] \,||\, T, H'', P, m_0}$$

Function Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x_1 = \texttt{def}\ (x_2, \ldots, x_n) = \{S\} \\ v = \langle m_0, \texttt{def}\ (x_2, \ldots, x_n) \to S\rangle \\ B_{\texttt{obj}} = \{\star x_{\texttt{value}} \mapsto m, \_\_\text{call}\_\_ \mapsto \langle m', \mathfrak{M}_{\texttt{call}}\rangle\} \qquad H' = H[m \mapsto v, m' \mapsto B_{\texttt{obj}}] \\ m, m' \notin H \qquad \textsc{Bind}(H', m_0, x_1, m') = H'' \qquad \ell \overset{s}{\blacktriangleleft} \overset{*}{\ell}'' \end{array}}{[\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}'', S\rangle] \,||\, T, H'', P, m_0}$$

Function Call Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x_1 = x_2(x_3, \ldots, x_n) \qquad m_{\texttt{raw}} = \textsc{Lookup}(m_0, P, H, x_2) \\ m = \textsc{GetCall}(H, m_{\texttt{raw}}) \qquad H[m][\star x_{\texttt{obj}}] = \langle m_0', \texttt{def}\ (x_3', \ldots, x_n') \to S'\rangle \\ m_0'' \notin H \qquad H' = H[m_0'' \mapsto \{\ \}] \qquad \forall i, 3 \leq i \leq n, m_i' = \textsc{Lookup}(m_0, P, H, x_i) \\ \textsc{Bind}(H', m_0'', x_1', m_1', \ldots, x_n', m_n') = H'' \\ P' = P \cup \{m_0'' \mapsto m_0'\} \qquad S' = [\ell'' : \ell''' : d, \ldots] \end{array}}{[\langle \ell, S\rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \ell'', S'\rangle, \langle l, S\rangle] \,||\, T, H'', P', m_0''}$$

Figure 1: Operational Semantics (cont.)

Method Call Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x_1 = x_2(x_3, \ldots, x_n) \\ m_{\texttt{raw}} = \textsc{Lookup}(m_0, P, H, x_2) \qquad m = \textsc{GetCall}(H, m_{\texttt{raw}}) \\ H[m][\star x_{\texttt{obj}}] = \langle m_0', m_{\texttt{obj}},\ \texttt{def}\ (x_3', \ldots, x_n') \rightarrow S' \rangle \qquad m_0'' \notin H \\ H' = H[m_0'' \mapsto \{x_{\texttt{self}} \mapsto m_{\texttt{obj}}\}] \qquad \forall i, 3 \leq i \leq n, m_i' = \textsc{Lookup}(m_0, P, H, x_i) \\ \textsc{Bind}(H', m_0'', x_1', m_1', \ldots, x_n', m_n') = H'' \\ P' = P \cup \{m_0'' \mapsto m_0'\} \qquad S' = [\ell'' : \ell''' : d, \ldots] \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \ell'', S' \rangle, \langle l, S \rangle] \,||\, T, H'', P', m_0''}$$

Attribute Assignment

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' : x_1 = x_2.x_3 \qquad m = \textsc{Lookup}(m_0, P, H, x_2) \\ H[m] = B \qquad B[x_3] = m' \qquad B_{\texttt{obj}} = \textsc{GetObj}(H, m') \\ m'' \notin H \qquad H' = H[m'' \mapsto B_{\texttt{obj}}] \qquad \textsc{Bind}(H', m_0, x_1, m'') = H'' \qquad \ell \overset{S}{\blacktriangleleft} \overset{*}{\ell}{}' \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}', S \rangle] \,||\, T, H'', P, m_0}$$

*(Gets values wrapped in objects. – TC)*

Raise Exception Caught

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' :\ \texttt{raise}\ x \qquad T = [\langle \ell_1, S_1 \rangle, \ldots, \langle \ell_n, S_n \rangle] \qquad m_0^0 = m_0 \\ \forall i, 1 \leq i \leq k, k < n, \textsc{Catch}(\langle P, \ell_i, S_i, m_0^{i-1} \rangle) =\ \texttt{undefined}\ ,\ \texttt{undefined}\ , m_0^i \\ \textsc{Catch}(\langle P, \ell_{k+1}, S_{k+1}, m_0^k \rangle) = \ell', x', m_0' \\ m = \textsc{Lookup}(m_0', P, H, x) \qquad \textsc{Bind}(H, m_0, x', m) = H' \qquad \ell_s' \overset{S_{k+1}}{\blacktriangleleft} \overset{*}{\ell}{}'' \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}'', S_{k+1} \rangle] \,||\, T, H', P, m_0'}$$

Raise Exception Escaped

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' :\ \texttt{raise}\ x \qquad T = [\langle \ell_1, S_1 \rangle, \ldots, \langle \ell_n, S_n \rangle] \\ \ell_1 = \ell' \qquad \forall i, 1 \leq i \leq n, \textsc{Catch}(\langle \ell_i, S_i \rangle) =\ \texttt{undefined} \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\,], H, P, m_0}$$

Pass

$$\frac{S(\ell) = \ell : \ell' :\ \texttt{pass} \qquad \ell \overset{S}{\blacktriangleleft} \overset{*}{\ell}{}''}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}'', S \rangle] \,||\, T, H, P, m_0}$$

Return

$$\frac{\begin{array}{c} S(\ell) = \ell : \ell' :\ \texttt{return}\ x \\ T = [t, \langle \ell'', S' \rangle] \,||\, T' \qquad m = \textsc{Lookup}(m_0, P, H, x) \qquad m_0' = P[m_0] \\ S'(\ell'') = \ell'' : \ell''' : x_1 = e \qquad \textsc{Bind}(H, m_0', x_1, m) = H' \qquad \ell'' \overset{S'}{\blacktriangleleft} \overset{*}{\ell}{}'''' \end{array}}{[t, \langle \ell'', S' \rangle] \,||\, T', H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}'''', S' \rangle] \,||\, T', H', P, m_0'}$$

Figure 1: Operational Semantics (cont.)

$$\textsc{Goto}\quad \frac{S(\ell) = \ell : \ell' :\ \texttt{goto}\ \ell'' \qquad (\ell'' : \ell''' : d) \in S}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \ell'', S \rangle] \,||\, T, H, P, m_0}$$

$$\textsc{GotoIfNot}\quad \frac{\begin{array}{c} S(\ell) = \ell : \ell' :\ \texttt{goto}\ \ell''\ \texttt{if not}\ x \\ m = \textsc{Lookup}(m_0, P, H, x) \qquad H[m][\star x_{\texttt{value}}] = \textsc{False} \qquad (\ell'' : \ell''' : d) \in S \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \ell'', S \rangle] \,||\, T, H, P, m_0}$$

$$\textsc{GotoIfNot Failed}\quad \frac{\begin{array}{c} S(\ell) = \ell : \ell' :\ \texttt{goto}\ \ell''\ \texttt{if not}\ x \\ m = \textsc{Lookup}(m_0, P, H, x) \qquad H[m][\star x_{\texttt{value}}] = \textsc{True} \qquad \ell \overset{S'}{\blacktriangleleft} \overset{*}{\ell}{}'' \end{array}}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}'', S \rangle] \,||\, T, H, P, m_0}$$

$$\textsc{Name Statement}\quad \frac{S(\ell) = \ell : \ell' : e \qquad B = H[m_0] \qquad \forall x \in e, \exists B[x] \qquad \ell \overset{S'}{\blacktriangleleft} \overset{*}{\ell}{}''}{[\langle \ell, S \rangle] \,||\, T, H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}'', S \rangle] \,||\, T, H, P, m_0}$$

$$\textsc{End of Function}\quad \frac{\begin{array}{c} T = [\langle \textsc{End}, S \rangle, \langle \ell, S' \rangle] \,||\, T' \qquad m_0' = P[m_0] \qquad S'(\ell) = \ell : \ell'' : x = e \\ m \notin H \qquad H' = H[m \mapsto \texttt{None}] \qquad \textsc{Bind}(H', m_0', x, m) = H'' \qquad \ell \overset{S'}{\blacktriangleleft} \overset{*}{\ell}{}' \end{array}}{[\langle \textsc{End}, S \rangle, \langle \ell, S' \rangle] \,||\, T', H, P, m_0 \longrightarrow^1 [\langle \overset{*}{\ell}{}', S' \rangle] \,||\, T', H'', P, m_0'}$$

$$\textsc{End of Program}\quad \frac{T = [\langle \textsc{End}, S \rangle]}{T, H, P, m_0 \longrightarrow^1 [\ ], H, P, m_0}$$

Figure 1: Operational Semantics (cont.)

$$\begin{array}{rcll}
\hat{S} & ::= & [\hat{s}, \ldots] & \textit{abstract programs} \\
\hat{s} & ::= & \hat{\ell} : \hat{\ell} : \hat{d} & \textit{abstract statements} \\
\hat{d} & ::= & \hat{x} \,\texttt{=}\, \hat{v} \mid \hat{x} \,\texttt{=}\, \hat{x} & \textit{abstract directives} \\
\hat{v} & ::= & \texttt{int}^{+} \mid \texttt{int}^{-} \mid \texttt{int}^{0} & \textit{abstract values} \\
\hat{x} & & & \textit{abstract variables} \\
\hat{\ell} & & & \textit{abstract labels}
\end{array}$$

Figure 2: Normalized Python Language Grammar

$$
\begin{array}{rcll}
\hat{G} & ::= & \{\hat{g}, \ldots\} & \textit{control flow graphs} \\
\hat{g} & ::= & \hat{o} \blacktriangleleft \hat{o} \mid \hat{o} \ll \hat{o} & \textit{control flow graph edge} \\
\hat{o} & ::= & \textsc{Start} \mid \textsc{End} \mid \hat{s} & \textit{control flow graph nodes}
\end{array}
$$

Figure 3: Control Flow Graph Grammar

$$
\textsc{Lexical Start}\quad \frac{\textsc{Start} \overset{?}{\blacktriangleleft} \hat{o}}{\hat{G} \longrightarrow^1 \hat{G} \cup \{\textsc{Start} \ll \hat{o}\}}
\qquad
\textsc{Literal Assignment}\quad \frac{\hat{o}_1 = (\hat{x} \texttt{=} \hat{v}) \qquad \hat{o}_1 \overset{?}{\blacktriangleleft} \hat{o}_2}{\hat{G} \longrightarrow^1 \hat{G} \cup \{\hat{o}_1 \ll \hat{o}_2\}}
$$

$$
\textsc{Variable Accessible}\quad \frac{\hat{o}_1 = (\hat{x} \texttt{=} \hat{x}') \qquad \hat{o}_1 \overset{?}{\blacktriangleleft} \hat{o}_2 \qquad \hat{v} \in \hat{G}(\hat{o}_1, [\hat{x}']) \qquad \hat{v} \neq \textsc{Undefined}}{\hat{G} \longrightarrow^1 \hat{G} \cup \{\hat{o}_1 \ll \hat{o}_2\}}
$$

Figure 4: Control Flow Graph Closure

$$
\begin{array}{rcll}
\hat{K} & ::= & [\hat{k}, \ldots] & \textit{lookup stacks} \\
\hat{k} & ::= & \hat{x} \mid \hat{v} \mid \textsc{Capture}(\mathbb{N}) \mid \textsc{Jump}(\hat{o}) & \textit{lookup stack elements}
\end{array}
$$

Figure 5: Value Lookup Grammar