Big Step Semantics

State

$$
\begin{array}{lcl}
\mathit{state} & = & \mathit{heap} \times \mathit{locals} \\
\mathit{locals} & = & \mathit{vname} \rightharpoonup \mathit{val} \\
\mathit{heap} & = & \mathit{addr} \rightharpoonup \mathit{obj} \\
\mathit{obj} & = & \mathit{cname} \times \mathit{fields} \\
\mathit{fields} & = & \mathit{vname} \times \mathit{cname} \rightharpoonup \mathit{val}
\end{array}
$$

Example state:

$$
\begin{array}{l}
([7 \mapsto (B, [(F,B) \mapsto \mathit{Addr}\ 1]), \\
\quad 5 \mapsto (C, [(F,C) \mapsto \mathit{Bool\ True}, (F,B) \mapsto \mathit{Intg}\ 5])], \\
\ [V \mapsto \mathit{Null}])
\end{array}
$$

Is this state “welformed”?

Question

What is the key difference between our abstract state and some concrete representation in memory?

Variable names

$$\begin{aligned} s &:: \textit{state} \\ l &:: \textit{locals} \\ h &:: \textit{heap} \end{aligned}$$

Abbreviations

$$
\begin{aligned}
\textit{true} &\equiv \texttt{Val}(\textit{Bool True}) & \qquad \textit{false} &\equiv \texttt{Val}(\textit{Bool False}) \\
\textit{addr a} &\equiv \texttt{Val}(\textit{Addr a}) & \qquad \textit{null} &\equiv \texttt{Val}\ \textit{Null} \\
\textit{unit} &\equiv \texttt{Val}\ \textit{Unit}
\end{aligned}
$$

Expression evaluation

Transition format:

$$P \vdash \langle e,s \rangle \Rightarrow \langle e',s' \rangle$$

where e' is fully evaluated: a value (or an exception)

Evaluation rules

Val and Var

$$P \vdash \langle \texttt{Val}\, v, s \rangle \Rightarrow \langle \texttt{Val}\, v, s \rangle$$

$$l\, V = \lfloor v \rfloor \Longrightarrow P \vdash \langle \texttt{Var}\, V, (h, l) \rangle \Rightarrow \langle \texttt{Val}\, v, (h, l) \rangle$$

Note

Semantics is defensive

Example: $\langle \mathit{Var}\ V, (h, l)\rangle$ can only evaluate if $V \in \mathit{dom}\ l$

Field access

$$[\![P \vdash \langle e, s_0 \rangle \Rightarrow \langle addr\ a, (h, l) \rangle;$$
$$h\ a = \lfloor (C, fs) \rfloor; fs\ (F, D) = \lfloor v \rfloor]\!]$$
$$\Longrightarrow P \vdash \langle e.F\{D\}, s_0 \rangle \Rightarrow \langle \mathtt{Val}\ v, (h, l) \rangle$$

Binary operations

$$[\![P \vdash \langle e_1,s_0\rangle \Rightarrow \langle \mathtt{Val}\, v_1,s_1\rangle;$$

$$P \vdash \langle e_2,s_1\rangle \Rightarrow \langle \mathtt{Val}\, v_2,s_2\rangle;$$

$$binop\ (bop, v_1, v_2) = \lfloor v \rfloor]\!]$$

$$\Longrightarrow P \vdash \langle e_1 \ll bop \gg e_2,s_0\rangle \Rightarrow \langle \mathtt{Val}\, v,s_2\rangle$$

new

$$\textit{new-Addr } h \equiv$$
$$\textit{if } \exists a.\ h\ a = \textit{None then } \lfloor \textit{SOME } a.\ h\ a = \textit{None} \rfloor \textit{ else None}$$

$$\textit{init-fields} :: ((\textit{vname} \times \textit{cname}) \times \textit{ty})\ \textit{list} \Rightarrow \textit{fields}$$
$$\textit{init-fields} \equiv \textit{map-of} \circ \textit{map}\ (\lambda(F, T).\ (F, \textit{default-val}\ T))$$

$$[\![\textit{new-Addr } h = \lfloor a \rfloor;$$
$$P \vdash C \textit{ has-fields FDTs};$$
$$h' = h(a \mapsto (C, \textit{init-fields FDTs}))]\!]$$
$$\Longrightarrow P \vdash \langle \texttt{new}\ C,(h, l)\rangle \Rightarrow \langle \textit{addr}\ a,(h', l)\rangle$$

Cast

$$P \vdash \langle e, s_0 \rangle \Rightarrow \langle null, s_1 \rangle \Longrightarrow$$
$$P \vdash \langle \texttt{Cast}\ C\ e, s_0 \rangle \Rightarrow \langle null, s_1 \rangle$$

$$[\![P \vdash \langle e, s_0 \rangle \Rightarrow \langle addr\ a, (h, l) \rangle;$$
$$h\ a = \lfloor (D, fs) \rfloor; P \vdash D \preceq^* C]\!]$$
$$\Longrightarrow P \vdash \langle \texttt{Cast}\ C\ e, s_0 \rangle \Rightarrow \langle addr\ a, (h, l) \rangle$$

Variable assignment

$$P \vdash \langle e, s_0 \rangle \Rightarrow \langle \texttt{Val}\, v, (h, l) \rangle \Longrightarrow$$

$$P \vdash \langle V \texttt{ := } e, s_0 \rangle \Rightarrow \langle unit, (h, l(V \mapsto v)) \rangle$$

In Java: `Val` *v* instead of *unit*.

Problem:

$$\texttt{if } (b)\ V_1 \texttt{ := } e_1 \texttt{ else } V_2 \texttt{ := } e_2$$

Would not be type correct in Jinja if type(V_1) $\neq$ type(V_2).

Field assignment

$$[\![P \vdash \langle e_1,s_0\rangle \Rightarrow \langle addr\ a,s_1\rangle;$$
$$P \vdash \langle e_2,s_1\rangle \Rightarrow \langle \texttt{Val}\ v,(h_2, l_2)\rangle;$$
$$h_2\ a = \lfloor(C, fs)\rfloor; fs' = fs((F, D) \mapsto v);$$
$$h_2{}' = h_2(a \mapsto (C, fs'))]\!]$$
$$\Longrightarrow P \vdash \langle e_1.F\{D\}\ \texttt{:=}\ e_2,s_0\rangle \Rightarrow$$
$$\langle unit,(h_2{}', l_2)\rangle$$

Method call

$$\llbracket P \vdash \langle e,s_0\rangle \Rightarrow \langle addr\ a,s_1\rangle;$$
$$P \vdash \langle ps,s_1\rangle\ [\Rightarrow]\ \langle map\ \mathtt{Val}\ vs,(h_2,\ l_2)\rangle;$$
$$h_2\ a = \lfloor (C,\ fs)\rfloor;$$
$$P \vdash C\ sees\ M\colon Ts \rightarrow T\ = (pns,\ body)\ in\ D;$$
$$|vs| = |pns|;\ l_2{}' = [this \mapsto Addr\ a,\ pns\ [\mapsto]\ vs];$$
$$P \vdash \langle body,(h_2,\ l_2{}')\rangle \Rightarrow \langle e',(h_3,\ l_3)\rangle\rrbracket$$
$$\Longrightarrow P \vdash \langle e.M(ps),s_0\rangle \Rightarrow \langle e',(h_3,\ l_2)\rangle$$

$\mathbf{P} \vdash \langle \mathbf{es},\mathbf{s}\rangle\ [\Rightarrow]\ \langle \mathbf{es}',\mathbf{s}'\rangle$

Evaluate *es* from left to right:

$$P \vdash \langle [],s\rangle\ [\Rightarrow]\ \langle [],s\rangle$$

$$[\![P \vdash \langle e,s_0\rangle \Rightarrow \langle \mathtt{Val}\ v,s_1\rangle;$$
$$P \vdash \langle es,s_1\rangle\ [\Rightarrow]\ \langle es',s_2\rangle]\!]$$
$$\Longrightarrow P \vdash \langle e \cdot es,s_0\rangle\ [\Rightarrow]\ \langle \mathtt{Val}\ v \cdot es',s_2\rangle$$

Why `Val` v and not just e'?

Because of exceptions (later).

Local variable

$$P \vdash \langle e_0, (h_0, l_0(V := None))\rangle \Rightarrow \langle e_1, (h_1, l_1)\rangle \Longrightarrow$$
$$P \vdash \langle \{V{:}T;\, e_0\}, (h_0, l_0)\rangle \Rightarrow \langle e_1, (h_1, l_1(V := l_0\ V))\rangle$$

Sequential composition

$$[\![P \vdash \langle e_0, s_0 \rangle \Rightarrow \langle \texttt{Val}\ v, s_1 \rangle;$$

$$P \vdash \langle e_1, s_1 \rangle \Rightarrow \langle e_2, s_2 \rangle]\!]$$

$$\Longrightarrow P \vdash \langle e_0\texttt{;}\ e_1, s_0 \rangle \Rightarrow \langle e_2, s_2 \rangle$$

Conditional

$$[\![P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{true}, s_1 \rangle; P \vdash \langle e_1, s_1 \rangle \Rightarrow \langle e', s_2 \rangle]\!]$$
$$\Longrightarrow P \vdash \langle \texttt{if}\,(e)\; e_1 \;\texttt{else}\; e_2, s_0 \rangle \Rightarrow \langle e', s_2 \rangle$$

$$[\![P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{false}, s_1 \rangle; P \vdash \langle e_2, s_1 \rangle \Rightarrow \langle e', s_2 \rangle]\!]$$
$$\Longrightarrow P \vdash \langle \texttt{if}\,(e)\; e_1 \;\texttt{else}\; e_2, s_0 \rangle \Rightarrow \langle e', s_2 \rangle$$

While loop

$$P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{false}, s_1 \rangle \Longrightarrow$$
$$P \vdash \langle \texttt{while}\,(e)\ c, s_0 \rangle \Rightarrow \langle \mathit{unit}, s_1 \rangle$$

$$[\![P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{true}, s_1 \rangle;$$
$$P \vdash \langle c, s_1 \rangle \Rightarrow \langle \texttt{Val}\ v_1, s_2 \rangle;$$
$$P \vdash \langle \texttt{while}\,(e)\ c, s_2 \rangle \Rightarrow \langle e_3, s_3 \rangle]\!]$$
$$\Longrightarrow P \vdash \langle \texttt{while}\,(e)\ c, s_0 \rangle \Rightarrow \langle e_3, s_3 \rangle$$

Digression: simultaneous inductive definitions

$P \vdash \langle e, s \rangle \Rightarrow \langle e', s' \rangle$ and $P \vdash \langle es, s \rangle \; [\Rightarrow] \; \langle es', s' \rangle$ are defined simultaneously:

> $P \vdash \langle e, s \rangle \Rightarrow \langle e', s' \rangle$ uses $P \vdash \langle es, s \rangle \; [\Rightarrow] \; \langle es', s' \rangle$ and
> $P \vdash \langle es, s \rangle \; [\Rightarrow] \; \langle es', s' \rangle$ uses $P \vdash \langle e, s \rangle \Rightarrow \langle e', s' \rangle$.

Consequence: *simultaneous rule induction*:

$$P \vdash \langle e, s \rangle \Rightarrow \langle e', s' \rangle \Longrightarrow R_1 \; P \; e \; s \; e' \; s'$$
$$P \vdash \langle es, s \rangle \Rightarrow \langle es', s' \rangle \Longrightarrow R_2 \; P \; es \; s \; es' \; s'$$

must be proved simultaneously.

Slight complication: property R_2 is auxiliary and needs to be determined. Usually R_2 is R_1 lifted to lists.

Example: Final expressions

$final\ e \equiv \exists v.\ e = \mathtt{Val}\ v$
$finals\ es \equiv \exists vs.\ es = map\ \mathtt{Val}\ vs$

Lemma If $P \vdash \langle e,s \rangle \Rightarrow \langle e',s' \rangle$ then $final\ e'$.
If $P \vdash \langle es,s \rangle\ [\Rightarrow]\ \langle es',s' \rangle$ then $finals\ es'$.

Proof by simultaneous rule induction.