# *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:

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

Is this state “welformed”?

# *Question*

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

# Variable names

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

# 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*

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

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

# *Note*

Semantics is defensive

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

# *Field access*

$$\frac{\begin{array}{c} P \vdash \langle e, s_0 \rangle \Rightarrow \langle addr\ a, (h, l) \rangle \\ h\ a = \lfloor (C, fs) \rfloor \\ fs\ (F, D) = \lfloor v \rfloor \end{array}}{P \vdash \langle e.F\{D\}, s_0 \rangle \Rightarrow \langle \texttt{Val}\ v, (h, l) \rangle}$$

# Binary operations

$$\frac{\begin{array}{c} P \vdash \langle e_1, s_0 \rangle \Rightarrow \langle \mathtt{Val}\ v_1, s_1 \rangle \\ P \vdash \langle e_2, s_1 \rangle \Rightarrow \langle \mathtt{Val}\ v_2, s_2 \rangle \\ \mathit{binop}\ (\mathit{bop},\ v_1,\ v_2) = \lfloor v \rfloor \end{array}}{P \vdash \langle e_1\ \text{«}\mathit{bop}\text{»}\ e_2, s_0 \rangle \Rightarrow \langle \mathtt{Val}\ v, s_2 \rangle}$$

# *new*

$$\textit{new-Addr}\ h \equiv$$

$$\textit{if}\ \exists\, a.\ h\ a = \textit{None then}\ \lfloor \textit{SOME}\ a.\ h\ a = \textit{None} \rfloor\ \textit{else None}$$

$$\textit{init-fields} :: ((\textit{vname} \times \textit{cname}) \times \textit{ty})\ \textit{list} \Rightarrow \textit{fields}$$

$$\textit{init-fields} \equiv \textit{map-of} \circ \textit{map}\ (\lambda(F, T).\ (F, \textit{default-val}\ T))$$

$$\frac{\begin{array}{c}\textit{new-Addr}\ h = \lfloor a \rfloor \\ P \vdash C\ \textbf{has-fields}\ \textit{FDTs} \\ h' = h(a \mapsto (C, \textit{init-fields FDTs}))\end{array}}{P \vdash \langle \texttt{new}\ C,(h, l)\rangle \Rightarrow \langle \textit{addr}\ a,(h', l)\rangle}$$

# *Cast*

$$\frac{\begin{array}{c} P \vdash \langle e,s_0\rangle \Rightarrow \langle addr\ a,(h,\ l)\rangle \\ h\ a = \lfloor(D,\ fs)\rfloor \\ P \vdash D \preceq^* C \end{array}}{P \vdash \langle \texttt{Cast}\ C\ e,s_0\rangle \Rightarrow \langle addr\ a,(h,\ l)\rangle}$$

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

# Variable assignment

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

# Field assignment

$$\frac{\begin{array}{c} P \vdash \langle e_1,s_0\rangle \Rightarrow \langle addr\ a,s_1\rangle \\ P \vdash \langle e_2,s_1\rangle \Rightarrow \langle \texttt{Val}\ v,(h_2,\ l_2)\rangle \\ h_2\ a = \lfloor(C,\ fs)\rfloor \\ fs' = fs((F,\ D) \mapsto v) \\ h_2' = h_2(a \mapsto (C,\ fs')) \end{array}}{P \vdash \langle e_1.F\{D\}\ \texttt{:=}\ e_2,s_0\rangle \Rightarrow \langle \texttt{Val}\ v,(h_2',\ l_2)\rangle}$$

# Method call

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

# $P \vdash \langle es,s\rangle \ [\Rightarrow]\ \langle es',s'\rangle$

Evaluate *es* from left to right:

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

$$\frac{P \vdash \langle e,s_0\rangle \Rightarrow \langle \texttt{Val}\ v,s_1\rangle \qquad P \vdash \langle es,s_1\rangle \ [\Rightarrow]\ \langle es',s_2\rangle}{P \vdash \langle e\cdot es,s_0\rangle \ [\Rightarrow]\ \langle \texttt{Val}\ v\cdot es',s_2\rangle}$$

Why `Val` *v* and not just $e'$?

Because of exceptions (later).

# Example

```
class C extends Object
{ field F:Integer
  method m(i:Integer):Integer =
    this.F{C} + Var i
}
```

State:

$$h = [a \mapsto (C, fs)]$$
$$fs = [(F, C) \mapsto Intg\ 2]$$
$$l = [V \mapsto Addr\ a]$$

Evaluation:

$$P \vdash \langle \texttt{Var}\ V.m([\texttt{Val}\ (Intg\ 1)]),(h, l)\rangle \Rightarrow \langle \texttt{Val}\ (Intg\ 3),(h, l)\rangle$$

# *Fields versus methods*

- Field lookup $e.F\{D\}$ is *static*:
  annotation $D$ is based on the static type of $e$.
- Method lookup $e.M(es)$ is *dynamic*:
  it is based on the dynamic type of the object $e$.

# Local variable

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

# Sequential composition

$$\frac{P \vdash \langle e_0, s_0 \rangle \Rightarrow \langle \texttt{Val}\ v, s_1 \rangle \qquad P \vdash \langle e_1, s_1 \rangle \Rightarrow \langle e_2, s_2 \rangle}{P \vdash \langle e_0\texttt{;}\ e_1, s_0 \rangle \Rightarrow \langle e_2, s_2 \rangle}$$

# Conditional

$$\frac{P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{true}, s_1 \rangle \qquad P \vdash \langle e_1, s_1 \rangle \Rightarrow \langle e', s_2 \rangle}{P \vdash \langle \texttt{if}\ (e)\ e_1\ \texttt{else}\ e_2, s_0 \rangle \Rightarrow \langle e', s_2 \rangle}$$

$$\frac{P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{false}, s_1 \rangle \qquad P \vdash \langle e_2, s_1 \rangle \Rightarrow \langle e', s_2 \rangle}{P \vdash \langle \texttt{if}\ (e)\ e_1\ \texttt{else}\ e_2, s_0 \rangle \Rightarrow \langle e', s_2 \rangle}$$

# *While loop*

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

$$\frac{\begin{array}{c} P \vdash \langle e, s_0 \rangle \Rightarrow \langle \mathit{true}, s_1 \rangle \\ P \vdash \langle c, s_1 \rangle \Rightarrow \langle \texttt{Val}\ v_1, s_2 \rangle \\ P \vdash \langle \texttt{while}\ (e)\ c, s_2 \rangle \Rightarrow \langle e_3, s_3 \rangle \end{array}}{P \vdash \langle \texttt{while}\ (e)\ c, s_0 \rangle \Rightarrow \langle e_3, s_3 \rangle}$$

# 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

$final\ e \equiv \exists v.\ e = \texttt{Val}\ v$

$finals\ es \equiv \exists vs.\ es = map\ \texttt{Val}\ vs$

**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.