The Judgement

$\boxed{H \,;\, e \Downarrow c}$ means "e evals to c under H"

- just a relation over $H \times e \times c$, i.e. $\subseteq$

  just a set of tuples
  "true" for tuples in set, false otherwise

- weird syntax follows PL convention, typical

Consider:

- $\bullet, x \mapsto 3 \,;\, x + y \Downarrow 3$
  - will be true (in our relation)
- $\bullet, x \mapsto 3 \,;\, x + y \Downarrow 6$
  - will be false (not in our relation)

Inference Rules : how we defn relations in PL

instantiate mvars by class

$$\frac{}{H ; c \Downarrow c}\,\text{const} \qquad \frac{}{H ; x \Downarrow H(x)}\,\text{Var}$$

$$\frac{H ; e_1 \Downarrow c_1 \quad H ; e_2 \Downarrow c_2}{H ; e_1 + e_2 \Downarrow c_1 + c_2}\,\text{ADD} \qquad \frac{\text{"}}{\text{"}}\,\text{MULT}$$

Top : hypotheses / input

Bottom : Conclusion / output

how to construct

- usually read bottom - up
- if hyps hold, then concl holds
- rules really "schema" you instantiate <u>consistently</u>

Example Instantion

$$\frac{\bullet, y \mapsto 4 ;\ 3+y \Downarrow 7 \qquad \bullet, y \mapsto 4 ;\ 5 \Downarrow 5}{\bullet, y \mapsto 4 ;\ (3+y)+5 \Downarrow 12}$$

Instantiates ADD w/

$H = \bullet, y \mapsto 4$

$e_1 = 3 + y$

$c_1 = 7$

$e_2 = 5$

$c_2 = 5$

Derivations

complete derivation has no hyps dangling

$$\dfrac{\dfrac{}{\cdot, y \mapsto 4 ; 3 \Downarrow 3} \qquad \dfrac{}{\cdot, y \mapsto 4 ; y \Downarrow 4}}{\vdots}$$

We say $H ; e \Downarrow c$

if $\exists$ derivation w/ $H ; e \Downarrow c$ @ root

Precise Meaning for our Inference Rules

What relation are we defining? we said all this subset stuff what gives?

$R_0 = \emptyset$

$(i > 0)$ $R_i = R_{i-1} \cup (H, e, c)$ s.t. $H; e \Downarrow c$ derivable using an ~~in~~ inference rule instantiated w/ hyps all in R_{i-1}

$\Longrightarrow R_i =$ all triples @ bottom of ~~ht~~ height j derivations for $j \leq i$

(Q: $<$?)

$R = \bigcup_{i \geq 0} R_i \sim R_\infty$

"smallest" relation closed under inference rules

Q: ~~How~~ prove 2 semantics equiv?

What have we done?

defined a sort of very precise, abstract, interpreter

- each hyp in deriv dispatches recursive cal
- "syntax directed" → no need for interp to search
  - ⤳ attach eval rule to each rule of grammar, unambiguous
- proof system: prove facts from other facts

Our Semantics is Nice (function)

"Total" ~~Progress~~: $\forall H, e, \exists c, \; H; e \Downarrow c$

Deterministic: $H; e \Downarrow c_1 \wedge H; e \Downarrow c_2 \Rightarrow c_1 = c_2$

what would div do? gettime?

Proofs by structural induction.

($\approx$ induction on height of e)

Onward: statements

similar
statements have exprs → use vars
→ need heaps to assign meaning

different
- statements don't eval to constants
- what do they ~~do~~? A: produce new heaps!

Q: always? A: No! non-termination!!

We could do something like:

$$H_1 ; S \Downarrow H_2$$

- but it'd be partial $H_1 \times S \to H_2$
- nothing wrong w/ that, problem?

Instead: <u>small-step semantics</u>

- so single step of eval
- "iterate" to "run" the prog

write relation as

$$H_1 ; s_1 \rightarrow H_2 ; s_2$$

$$\boxed{H_1 ; s_1 \rightarrow H_2 ; s_2}$$

$$\frac{H ; e \Downarrow c}{H ; x := e \rightarrow H, x \mapsto c ; \text{skip}} \text{Assign}$$

$$\frac{H ; e \Downarrow c \qquad c > 0}{H ; \text{if } e\ s_1\ s_2 \rightarrow H ; s_1} \text{If T}$$

$$\frac{}{H ; \text{skip}; s \rightarrow H ; s} \text{seq}$$

$$\frac{H ; s_1 \rightarrow H' ; s_1'}{H ; s_1 ; s_2 \rightarrow H' ; s_1' ; s_2} \text{seq}$$

... If F

What about while? (do s & loop if e>0)

$$\frac{}{H;\ \text{while } e\ s \rightarrow H;\ \text{if } e\ (s;\ \text{while } e\ s)\ \text{skip}}\ \text{while}$$

Are these syntax directed?

So we defined $\boxed{H\ ;\ s \rightarrow H'\ ;\ s'}$,

but what does "s" mean / do ?

- It transforms a heap in one step

☆ - which heap to start in ?

We can iterate:

$$H_1\ ;\ s_1 \underbrace{\rightarrow}_{\substack{\text{each step}\\ \text{justified}\\ \text{by complete}\\ \text{derivation}}} H_2;\ s_2 \rightarrow \ldots$$

~~Notation~~

"reachable"

$H_1 ; s_1 \xrightarrow{n} H_2 ; s_2 \equiv$ takes n steps

$H_1 ; s_1 \xrightarrow{*} H_2 ; s_2 \equiv$ takes 0 or more steps

"reflexive, transitive closure"

equivalence class ? No!

no symmetry

---

could pick special variable "ans" for answer

Then "s produces c" if $\bullet ; s \xrightarrow{*} H ; \text{skip} \wedge H(\text{ans}) =$

Does every s produce a c?

Example program execution

$$\mathrm{x} := 3; (\mathrm{y} := 1; \textbf{while}\ \mathrm{x}\ (\mathrm{y} := \mathrm{y} * \mathrm{x}; \mathrm{x} := \mathrm{x}{-}1))$$

Let's write some of the state sequence. You can justify each step with a full derivation. Let $s = (\mathrm{y} := \mathrm{y} * \mathrm{x}; \mathrm{x} := \mathrm{x}{-}1)$.

$$
\begin{array}{ll}
 & \cdot;\ \mathrm{x} := 3; \mathrm{y} := 1; \textbf{while}\ \mathrm{x}\ s \\
\rightarrow & \cdot, \mathrm{x} \mapsto 3;\ \textbf{skip}; \mathrm{y} := 1; \textbf{while}\ \mathrm{x}\ s \\
\rightarrow & \cdot, \mathrm{x} \mapsto 3;\ \mathrm{y} := 1; \textbf{while}\ \mathrm{x}\ s \\
\rightarrow^2 & \cdot, \mathrm{x} \mapsto 3, \mathrm{y} \mapsto 1;\ \textbf{while}\ \mathrm{x}\ s \\
\rightarrow & \cdot, \mathrm{x} \mapsto 3, \mathrm{y} \mapsto 1;\ \textbf{if}\ \mathrm{x}\ (s; \textbf{while}\ \mathrm{x}\ s)\ \textbf{skip} \\
\rightarrow & \cdot, \mathrm{x} \mapsto 3, \mathrm{y} \mapsto 1;\ \mathrm{y} := \mathrm{y} * \mathrm{x}; \mathrm{x} := \mathrm{x} - 1; \textbf{while}\ \mathrm{x}\ s
\end{array}
$$

Recap

IMP, Proofs

$\underbrace{H ; e \Downarrow c}_{\text{exprs done w/ "big step"}}$, $\underbrace{H ; s \rightarrow H' ; s'}_{\text{stmts done w/ "small step"}}$

Semantics by interp:

- prog means what very abstract interp says it means.
- precise interp written in Metalanguage

Big Step limitations

- can't distinguish errors & divergence
- IMP has no errors
- exprs can't diverge

Proving Progs

can reason about progs by "running" the