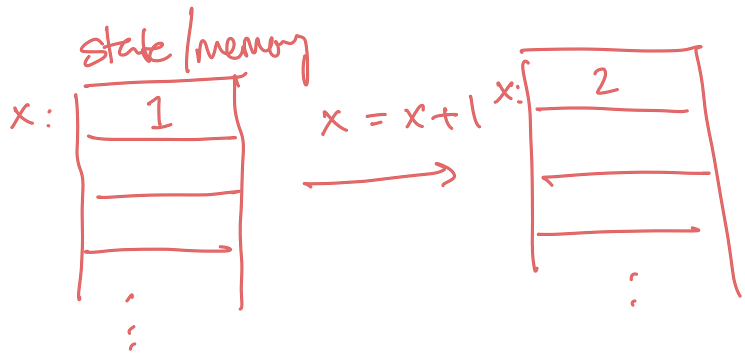
Semantics

$t ::= x \mid \lambda x.t \mid t\ t$

β reduction (semantics) $\qquad t \longrightarrow t'$

operational semantics



denotational semantics

pixels $\longrightarrow$ NN $\longrightarrow$ prediction

$\mathbb{R}^n \longrightarrow f \longrightarrow \mathbb{R}^m$

differentiable smooth $\qquad$ probabilities

mathematical interpretation of a program

"high-level" view

axiomatic semantics (Hoare / Floyd-Hoare logic)

$x = x+1$

axiom if $x > 0$ then after executing $x = x+1$ x is still > 0

Semantics with applications

arithmetic expression

$a ::= n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2$

Boolea expressions

$b ::= \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b_1 \mid b_1 \wedge b_2$

NOT AND

Program

$P ::= x := a \mid \text{skip} \mid P_1 ; P_2 \mid$
$\text{if } b \text{ then } P_1 \text{ else } P_2 \mid$
$\text{while } b \text{ then } P_1$

Var set of all variables

a state $s : \text{Var} \to \mathbb{Z}$ int

E.g. $s = \begin{cases} x \mapsto 0 \\ y \mapsto 70 \\ z \mapsto -200 \end{cases}$

$s[x \mapsto 10] = \begin{cases} x \mapsto 10 \\ y \mapsto 70 \\ z \mapsto -200 \end{cases}$

substitution

Semantics of expressions

$[\![a]\!] : State \to \mathbb{Z}$ $\qquad$ $[\![b]\!] : State \to \mathbb{B}$ $\{true, false\}$

set of all possible states

e.g. $[\![x + y]\!](s) = 0 + 70 = 70$

$[\![x = y]\!](s) = false$

Semantics of programs

Natural / Big step semantics

$[\![p]\!] : State \to \{State, undef\}$

program may not terminate

$\langle P, s \rangle \to s'$

program $\qquad$ state $\qquad$ final state

$\boxed{\text{Skip}}$ $\qquad$ $\langle skip, s \rangle \to s$

(for any state s, if you execute "skip" you arrive at state s)

$\boxed{\text{asn}}$ $\qquad$ $\langle x := a, s \rangle \to s[x \mapsto \underbrace{[\![a]\!](s)}_{\text{simpler notation}}]$

$s(a)$

Sequential composition

$\langle P_1 ; P_2, s\rangle \longrightarrow s''$

where $\langle P_1, s\rangle \longrightarrow s'$

$\langle P_2, s'\rangle \longrightarrow s''$

for some s'

alternative notation

$$\frac{\langle P_1, s\rangle \longrightarrow s' \qquad \langle P_2, s'\rangle \to s''}{\langle P_1 ; P_2, s\rangle \longrightarrow s''}$$

if

$\langle \text{if } b \text{ then } P_1 \text{ else } P_2, s\rangle \longrightarrow s'$

assuming $[\![b]\!](s) = \text{true}$

and $\langle P_1, s\rangle \longrightarrow s'$

$\langle \text{if } b \ldots, s\rangle \longrightarrow s'$

assuming $[\![b]\!](s) = \text{false}$

and $\langle P_2, s\rangle \longrightarrow s'$

while

$$\langle \text{while } b \text{ do } P, s\rangle \longrightarrow s$$

$$\text{if } [\![b]\!](s) = \text{false}$$

$$\langle \text{while } b \text{ do } P, s\rangle \longrightarrow s''$$

$$\text{if } [\![b]\!](s) = \text{true}$$

$$\langle P, s\rangle \longrightarrow s'$$

$$\langle \text{while } b \text{ do } P, s'\rangle \longrightarrow s''$$

E.g. $\langle z := x \,;\, x := y \,,\, s_0\rangle$

$$s_0 = \begin{cases} x \mapsto 5 \\ y \mapsto 7 \\ z \mapsto 0 \end{cases}$$

$$\langle z := x, s_0\rangle = s_1 = \begin{cases} x \mapsto 5 \\ y \mapsto 7 \\ z \mapsto 5 \end{cases}$$

$$\langle x := y, s_1\rangle = s_2 = \begin{cases} x \mapsto 7 \\ y \mapsto 7 \\ z \mapsto 5 \end{cases}$$

Equivalence

two programs are equivalent iff

for any s, s' $\quad \langle P_1, s\rangle \to s'$ iff $\langle P_2, s\rangle \to s'$

E.g. we can prove that

while b do P

(for any b, P)

is equivalent to

if b then
  P;
  while b do P;
else
  skip;

Theorem: the semantics are deterministic

assume $\langle P, s\rangle \to s'$

if $\langle P, s\rangle \to s''$ then $s' = s''$

$$[\![P]\!] : \text{State} \to \text{State} \cup \{\text{undef}\}$$

$$[\![P]\!](s) = \begin{cases} s' & \text{if } \langle P, s\rangle \to s' \\ \text{undef} & \text{o/w} \end{cases}$$

$$[\![\text{while true do } x := 1]\!](s) = \text{undef}$$

---

why "big-step" (Natural)?

Small-step typically useful for concurrency, low level overhead