Imperative Programming

Recall our expression syntax: we had numerals, Booleans, variables, arithmetic operations, Boolean operations, comparisons, pairing, projection, if-then-else, abstracts ...

We now look at a different toy language, an imperative one. This language will have arithmetic and Boolean expressions, but also have commands.

$$c ::= \mathrm{skip} \mid x := e \mid c_1 ; c_2 \mid \mathrm{if}\ b\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2 \mid \mathrm{while}\ b\ \mathrm{do}\ c\ \mathrm{end}$$

A command tells the machine what to do next.

Executing a command takes the machine from one state to another.

finite map from variables to values

Already saw this: table γ

We know how to evaluate expressions at a particular state

under a particular table

How do we specify what it means to execute a command?

As earlier, by using big-step semantics.

By $\gamma \xrightarrow{[c]} \gamma'$, we denote the fact that executing command c in state γ takes the system to state γ'.

$$\frac{}{\gamma \xrightarrow{[\texttt{skip}]} \gamma}\ \textit{skip} \qquad \frac{\gamma \vdash e \Rightarrow \underline{a}}{\gamma \xrightarrow{[x := e]} \gamma[\underline{x} \mapsto \underline{a}]}\ \textit{asgn}$$

$$\frac{\gamma \xrightarrow{[c_1]} \gamma'' \qquad \gamma'' \xrightarrow{[c_2]} \gamma'}{\gamma \xrightarrow{[c_1 ;\, c_2]} \gamma'}\ \textit{seq}$$

$$\frac{\gamma \vdash \underline{b} \Rightarrow \underline{T} \qquad \gamma \xrightarrow{[c_1]} \gamma'}{\gamma \xrightarrow{[\text{if } b \text{ then } c_1 \text{ else } c_2]} \gamma'}\ \textit{ifT} \qquad \frac{\gamma \vdash \underline{b} \Rightarrow \underline{F} \qquad \gamma \xrightarrow{[c_2]} \gamma'}{\gamma \xrightarrow{[\text{if } b \text{ then } c_1 \text{ else } c_2]} \gamma'}\ \textit{ifF}$$

$$\frac{\gamma \vdash \underline{b} \Rightarrow \underline{F}}{\gamma \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma}\ \textit{whileF}$$

$$\frac{\gamma \vdash \underline{b} \Rightarrow \underline{T} \qquad \gamma \xrightarrow{[c]} \gamma' \qquad \gamma' \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma''}{\gamma \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma''}\ \textit{whileT}$$

$x := 2;$ c_1

$\text{if } (x <= 1) \text{ then } z := 3 \text{ else } y := 4$ c_2

Denote by (x^0, y^0) the state $(x \mapsto x^0,\ y \mapsto y^0)$.

Can we show that $(0,0) -[c_1 ; c_2]\to (2,4)$.

$$
\dfrac{
\dfrac{(0,0) \vdash \underline{2} = \underline{2}}{(0,0) -[c_1]\to (2,0)}
\qquad
\dfrac{(2,0) \vdash (\underline{2 <= 1}) \Rightarrow \underline{f} \qquad \dfrac{(2,0) \vdash \underline{4} = \underline{4}}{(2,0) -[y := 4]\to (2,4)}}{(2,0) -[c_2]\to (2,4)}
}{(0,0) -[c_1 ; c_2]\to (2,4)}
$$

Theorem (Determinism of commands):

For any command c and state γ, there is at most one state γ' such that $\gamma \dashv [c] \mapsto \gamma'$.

$\forall c, \gamma, \gamma_1, \gamma_2$:

if $\gamma \dashv [c] \mapsto \gamma_1$ and $\gamma \dashv [c] \mapsto \gamma_2$, then $\gamma_1 = \gamma_2$.

Proof by induction on the structure of the proof for $\gamma \dashv [c] \mapsto \gamma_1$

Let π be a proof of $\gamma \dashv [c] \mapsto \gamma_1$ ending in a rule r.

IH: For any proof ϖ of size smaller than that of π, if ϖ has conclusion $\gamma_a \dashv [c'] \mapsto \gamma_b$, then for any γ_b' s.t. $\gamma_a \dashv [c'] \mapsto \gamma_b'$, $\gamma_b' = \gamma_b$.

The following cases arise.

- $r = skip$: π has the following structure.

$$\frac{}{\gamma \dashv [skip] \mapsto \gamma}$$

Thus, $\gamma_1 = \gamma$ and $c = skip$. So, $\gamma_2 = \gamma$.

- $r = asgn$:

π has the following form: $\dfrac{\gamma \vdash \underline{e} \Rightarrow \underline{a}}{\gamma \xrightarrow{[x := e]} \gamma[\underline{x} \mapsto \underline{a}]}$

So, $\gamma_1 = \gamma[\underline{x} \mapsto \underline{a}]$

Since there is only one $\underline{a}$ such that $\gamma \vdash \underline{e} \Rightarrow \underline{a}$, and augmenting γ with $\underline{x} \mapsto \underline{a}$ is deterministic, $\gamma_2 = \gamma_1$.

- $r = seq$:

π has the following form: $\dfrac{\gamma \xrightarrow{[c_1]} \gamma'' \quad \gamma'' \xrightarrow{[c_2]} \gamma'}{\gamma \xrightarrow{[c_1 ; c_2]} \gamma'}$

By IH, there is at most one γ'' and at most one γ' s.t. $\gamma \xrightarrow{[c_1]} \gamma''$ and $\gamma'' \xrightarrow{[c_2]} \gamma'$, so done.

- $r = ifT$

π has the following form: $\dfrac{\gamma \vdash \underline{b} \Rightarrow \underline{T} \quad \gamma \xrightarrow{[c_1]} \gamma'}{\gamma \xrightarrow{[\text{if } b \text{ then } c_1 \text{ else } c_2]} \gamma'}$

By IH, there is at most one γ' s.t. $\gamma \xrightarrow{[c_1]} \gamma'$, so done.

- $r = \text{if}F$ Proceeds very similarly to the case when $r = \text{if}T$.
- $r = \text{while}T$

  π looks as follows:

$$\frac{\gamma \vdash \underline{b} \Rightarrow \underline{T} \qquad \gamma \xrightarrow{[c]} \gamma' \qquad \gamma' \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma''}{\gamma \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma''}$$

  By IH, there is at most one γ' and one γ'' s.t. $\gamma \xrightarrow{[c]} \gamma'$ and $\gamma' \xrightarrow{[\text{while } b \text{ do } c \text{ end}]} \gamma''$, so done.

- $r = \text{while}F$

  $\gamma' = \gamma$, so done.

Theorem (Non-Termination of a "true" loop)

There are no two states γ and γ' s.t.

$$\gamma \text{ -}[\text{while } T \text{ do skip end}]\rightarrow \gamma'.$$

Proof by contradiction

Suppose there are two states γ_1 and γ_2 s.t.

$$\gamma_1 \text{ -}[\text{while } T \text{ do skip end}]\rightarrow \gamma_2.$$

Then, there must be a proof π with this conclusion of minimal size. Let the last rule of π be r. We can, thanks to the structure of while T do skip end, already rule out all possibilities for r other than whileF and whileT.

We can further rule out whileF since a premise that must be fulfilled for it to apply there is $\underline{T} \Rightarrow \underline{F}$, which is impossible under any γ.

Therefore, the only possibility left is $r = \text{whileT}$.

In this case, the proof π looks as follows.

$$\dfrac{\gamma_1 \vdash \underline{T} \Rightarrow \underline{T} \qquad \gamma_1 \dashv[\text{skip}]\!\!\to \gamma' \qquad \overbrace{\gamma' \dashv[\text{while T do skip end}]\!\!\to \gamma_2}^{\pi'}}{\gamma_1 \dashv[\text{while T do skip end}]\!\!\to \gamma_2}$$

Since $\gamma_1 \dashv[\text{skip}]\!\!\to \gamma'$, $\gamma' = \gamma_1$.

However, this means that π' is a proof of $\gamma_1 \dashv[\text{while T do skip end}]\!\!\to \gamma_2$ of size **strictly** smaller than that of π.

This contradicts our assumption that π was a minimal proof of $\gamma_1 \dashv[\text{while T do skip end}]\!\!\to \gamma_2$.

Exercise:

$i := 0\ ;\ x := 0\ ;\ y := 1\ ;\ t := 0\ ;\ n := 3 \leftarrow c_1$

while $(i <= n)$ do

$c_2 \rightarrow i := i+1\ ;\ t := x\ ;\ x := y\ ;\ y := t+y$

end

ω

Denote by $(i^\circ, x^\circ, y^\circ, t^\circ, n^\circ)$ the state $(i \mapsto i^\circ, x \mapsto x^\circ, y \mapsto y^\circ, t \mapsto t^\circ, n \mapsto n^\circ)$.

$(0,0,1,0,3) \longrightarrow [c_1 ; \omega] \longrightarrow ?\ (3, 1, 2, 1, 3)$

$i, x, y, t, n \qquad 0 \quad 1 \quad 2 \quad 3$

$i+1, y, t+y, x, n$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ t \end{bmatrix} = \begin{matrix} x+y+t = 1 \\ x+2y+t \quad 2 \\ y+t \quad 1 \end{matrix} \qquad \begin{matrix} 0 & x \\ 1 & y \\ 0 & t \end{matrix}$$

$$x := 4;\; y := 1;\; z := 0; \;\longleftarrow c_1$$

$$\left.\begin{array}{l} \text{while } (x > z) \text{ do} \\ c_2 \longrightarrow \quad z := z+1;\; y := y * z \\ \text{end}\end{array}\right) w$$

Denote by (a, b, c) the state $[x \mapsto a, y \mapsto b, z \mapsto c]$.

Can we show that $(0, 0, 0) \xrightarrow{[c_1; w]} (4, 24, 4)$?

* More importantly,

Show that for all $m \in \mathbb{N}$,

$$(0, 0, 0) \xrightarrow{[x := m;\ y := 1;\ z := 0;\ w]} (m, m!, m)$$

We first prove that for all $m, n, p \in \mathbb{N}$ where $p <= m$,

$$(m, n, p) \xrightarrow{[w]} (m, f(m,n,p), m) \text{ with}$$

$$f(m,n,p) = m * (m-1) * (m-2) * \cdots (p+2) * (p+1) * n.$$

How? Induction! (on $m-p$).

Base case: $m-p = 0$.

$m = p$ and $f(m,n,p) = n$.

$(m,n,p) \not\models z < x$, we have $(m,n,p) \xrightarrow{[w]} (m, \underbrace{n}_{f(m,n,p)}, p)$

Induction step: $m-p > 0$

$p < m \Rightarrow p+1 <= m$.

Can do this since IH applies for <u>ALL</u> m, n, p s.t. $p <= m$.

By IH, $(m, n*(p+1), p+1) \xrightarrow{[w]} (m, f(m, n*(p+1), p+1), m)$.

$$f(m, n*(p+1), p+1) = m*(m-1)* \cdots (p+2)* n*(p+1)$$
$$= f(m, n, p)$$

$(m, n, p) \xrightarrow{[z := z+1;\ y := y*z]} (m, n*(p+1), p+1)$

and $(m, n*(p+1), p+1) \xrightarrow{[w]} (m, f(m, n, p), m)$

So, $(m, n, p) \xrightarrow{[w]} (m, f(m, n, p), m)$.

In particular, $(0, 0, 0) \xrightarrow{[x := m;\ y := 1;\ z := 0]} (m, 1, 0)$ and $(m, 1, 0) \xrightarrow{[w]} (m, \underbrace{f(m, 1, 0)}_{m!}, m)$