Introduction to Operational Semantics

- An Imperative Language: **IMP**
  - syntax
  - semantics

Syntactic Sets

- number N
- truth values B
- arithmetic expressions $Aexp$
- boolean expression $Bexp$
- commands Com
- locations Loc

meta-variables

n, m : range over $\boldsymbol{N}$

X, Y: range over $\boldsymbol{Loc}$

a: to represent an arbitrary element of $\boldsymbol{Aexp}$

b: to represent an arbitrary element of $\boldsymbol{Bexp}$

c : to represent an arbitrary element of $\boldsymbol{Com}$

formation rules

- ***Aexp***

$$a := \mathtt{n} \mid X \mid a_0 \texttt{ + } a_1 \mid a_0 \texttt{ - } a_1 \mid a_0 \times a_1$$

- ***Bexp***

$$b ::= \mathtt{true} \mid \mathtt{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

- ***Com***

$$c ::= \mathtt{skip} \mid c_1 \texttt{ ; } c_2 \mid \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1 \mid X \texttt{ := } a \mid \mathtt{while}\ b\ \mathtt{do}\ c$$

Definition. Let e_0 and e_1 be from the same syntactic set. We say that e_0 and e_1 are (syntactically) identical if e_0 and e_1 have been built-up in exactly the same way (i.e. the same parse tree).

Notation. we write $e_0 \equiv e_1$ for e_0 and e_1 are identical

$$3 + 5 \not\equiv 8$$

$$5 + 3 \not\equiv 8$$

Let ***S*** be the universe of derivable objects;
let $S, S_0, \ldots, S_i, \ldots$ be generic elements of ***S***.

Rules:

i. axioms (0-ary rules) $\overline{S}\ ax$ or symply S

ii. k-ary rules $\dfrac{S_0, \ldots, S_{k-1}}{S}\ r$

Derivation trees

- If $\dfrac{}{S}\,ax$ is an axiom then S is a derivation tree

- if $\dfrac{S_0, \ldots, S_k}{S}\,r$ is a rule (instance) and

  and $\forall j \in [0, k]$ $\quad \begin{array}{c} \vdots\, D_j \\ S_j \end{array}$ is a derivation tree then

$$\dfrac{\begin{array}{c} \vdots\, D_0 \\ S_0 \end{array} \quad \cdots \quad \begin{array}{c} \vdots\, D_k \\ S_k \end{array}}{S}\,r$$

is a derivation tree

Notation:

R-derivation: a derivation that use a set R of rules

D is an R-derivation of S

$$D \vdash_R S$$

$\vdash_R S$ means that there is a derivation *D* *s.t.* $D \vdash_R S$

we will omit the subscript *R* when the set *R* of rule is understood

Book Notation

$d \Vdash_R y$ (*d* is an R-derivation of *y*)

Operational Semantics

- **evaluation** of arithmetic and boolean expression
- **execution** of commands
  - **semantic sets**

$\mathbf{Z}$: set of "*machine number*"

$\mathbf{\Sigma}$: set of "*states*"

$$\mathbf{\Sigma} : \{\sigma | \sigma : \mathbf{Loc} \rightarrow \mathbf{Z}\}$$

FINE 9 GENN

C: set of "*configurations*"

$$\mathbf{C} = \{\langle p, \sigma \rangle | p \in \mathbf{Aexp}, \mathbf{Bexp},\ \mathbf{Com}, \sigma \in \mathbf{\Sigma}\}$$

evaluation of **Aexp** elements

evalutation relation: $\langle a, \sigma \rangle \to n \quad (n \in \mathbf{Z})$

Numbers:

$$\langle \mathtt{n}, \sigma \rangle \to n$$

locations:

$$\langle X, \sigma \rangle \to \sigma(X)$$

sums:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \,\mathtt{+}\, a_1, \sigma \rangle \to n_0 \, + \, n_1}$$

substractions:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \;\mathtt{-}\; a_1, \sigma \rangle \to n_0 \, - \, n_1}$$

products:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \times a_1, \sigma \rangle \to n_0 \, \times \, n_1}$$

example:

$$a \equiv (\mathtt{Int}\ +\ \mathtt{5}) + (\mathtt{7} + \mathtt{9})$$

state σ_0

$$\sigma_0(\mathtt{Int}) = 0$$

derivation tree

$$\frac{\dfrac{\overline{\langle \mathtt{Int}, \sigma_0 \rangle \to 0} \quad \overline{\langle \mathtt{5}, \sigma_0 \rangle \to 5}}{\langle (\mathtt{Int}\ +\ \mathtt{5}), \sigma_0 \rangle \to 5} \qquad \dfrac{\overline{\langle \mathtt{7}, \sigma_0 \rangle \to 7} \quad \overline{\langle \mathtt{9}, \sigma_0 \rangle \to 9}}{\langle (\mathtt{7}\ +\ \mathtt{9}), \sigma_0 \rangle \to 16}}{\langle (\mathtt{Int}\ +\ \mathtt{5})\ +\ (\mathtt{7}\ +\ \mathtt{9}), \sigma_0 \rangle \to 21}$$

step 1 (tree construction)

$$\dfrac{\dfrac{\overline{\langle Int, \sigma_0 \rangle \rightarrow ?} \quad \overline{\langle 5, \sigma_0 \rangle \rightarrow ?}}{\langle (\texttt{Int} \ +5), \sigma_0 \rangle \rightarrow ?} \qquad \dfrac{\overline{\langle 7, \sigma_0 \rangle \rightarrow ?} \quad \overline{\langle 9, \sigma_0 \rangle \rightarrow ?}}{\langle (\ 7 \ + \ 9), \sigma_0 \rangle \rightarrow ?}}{\langle (Int + \ 5) \ + \ (7 \ + \ 9), \sigma_0 \rangle \rightarrow ?}$$

step 2 (replacement of all the “?”)

$$\dfrac{\dfrac{\overline{\langle Int, \sigma_0 \rangle \rightarrow ?} \quad \overline{\langle 5, \sigma_0 \rangle \rightarrow ?}}{\langle (\texttt{Int} \ +5), \sigma_0 \rangle \rightarrow ?} \qquad \dfrac{\overline{\langle 7, \sigma_0 \rangle \rightarrow ?} \quad \overline{\langle 9, \sigma_0 \rangle \rightarrow ?}}{\langle (\ 7 \ + \ 9), \sigma_0 \rangle \rightarrow ?}}{\langle (Int + \ 5) \ + \ (7 \ + \ 9), \sigma_0 \rangle \rightarrow ?}$$

an equivalence relation

$$a_0 \sim a_1 \text{ iff } (\forall n \in \boldsymbol{Z}\ \forall \sigma \in \sum.\ \langle a_0, \sigma \rangle \to n \Leftrightarrow \langle a_1, \sigma \rangle \to n)$$

... it is necessary to show that such a relation is:

1. reflexive
2. symmetric
3. transitive

evaluation of **Bexp** elements

evalutation relation: $\langle b, \sigma \rangle \to t \quad t \in \{\mathbf{true}, \mathbf{false}\}$

$$\langle \texttt{true}, \sigma \rangle \to true$$

$$\langle \texttt{false}, \sigma \rangle \to false$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \;\; \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \texttt{ = } a_1, \sigma \rangle \to true} \text{ when } n_0 = n_1$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \;\; \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \texttt{ = } a_1, \sigma \rangle \to false} \text{ when } n_0 \neq n_1$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \;\; \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \leq a_1, \sigma \rangle \to true} \text{ when } n_0 \leq n_1$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \;\; \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \leq a_1, \sigma \rangle \to false} \text{ when } n_0 \not\leq n_1$$

$$\frac{\langle b, \sigma \rangle \to t}{\langle \neg b, \sigma \rangle \to \neg t}$$

$$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \wedge b_1, \sigma \rangle \to t_0 \wedge t_1}$$

$$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \vee b_1, \sigma \rangle \to t_0 \vee t_1}$$

$$b_0 \sim b_1 \text{ iff } (\forall t \in \{false, true\} \ \forall \sigma \in \sum . \ \langle b_0, \sigma \rangle \to t \Leftrightarrow \langle b_1, \sigma \rangle \to t)$$

left-first-sequential evaluation:

$$\frac{\langle b_0, \sigma \rangle \to false}{\langle b_0 \wedge b_1, \sigma \rangle \to false}$$

$$\frac{\langle b_0, \sigma \rangle \to true \quad \langle b_1, \sigma \rangle \to false}{\langle b_0 \wedge b_1, \sigma \rangle \to false}$$

$$\frac{\langle b_0, \sigma \rangle \to true \quad \langle b_1, \sigma \rangle \to true}{\langle b_0 \wedge b_1, \sigma \rangle \to true}$$

execution of commands

command execution relation → $\langle c, \sigma \rangle \to \sigma'$

we assume the existence of an *initial state* σ_0 such that

$$(\forall X \in \boldsymbol{Loc}.\ \sigma_0(X) = 0)$$

$\langle c, \sigma \rangle \to \sigma'$ ⇒ the execution of c in σ terminates in σ ′

Notation. Let σ be a state. Let $m \in \boldsymbol{Z}$. Let $X \in Loc$. We write $\sigma[m/X]$ for "the state obtained from σ by replacing the contents of X by m", i.e.

$$\sigma[m/X](Y) = \begin{cases} m & \text{if } Y = X \\ \sigma(Y) & \text{otherwise} \end{cases}$$

Example: consider σ such that $\sigma(X) = 2$, $\sigma(Y) = 4$. Let σ' be $\sigma[5/X]$. We have that $\sigma'(X) = 5$, $\sigma'(Y) = 4$

Atomic commands:

$$\langle \texttt{skip},\ \sigma\rangle \rightarrow \sigma$$

$$\frac{\langle a,\ \sigma\rangle \rightarrow m}{\langle X\ \texttt{:=}\ a,\ \sigma\rangle \rightarrow \sigma[m/X]}$$

Sequencing:

$$\frac{\langle c_0, \sigma\rangle \rightarrow \sigma''\quad \langle c_1, \sigma''\rangle \rightarrow \sigma'}{\langle c_0\ \texttt{;}\ \ c_1, \sigma\rangle \rightarrow \sigma'}$$

Conditionals:

$$\frac{\langle b, \sigma \rangle \to true \ \ \langle c_0, \sigma \rangle \to \sigma'}{\langle \texttt{if} \ \ b \ \texttt{then} \ \ c_0 \ \texttt{else} \ \ c_1, \sigma \rangle \to \sigma'}$$

$$\frac{\langle b, \sigma \rangle \to false \ \ \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if} \ \ b \ \texttt{then} \ \ c_0 \ \texttt{else} \ \ c_1, \sigma \rangle \to \sigma'}$$

While-loops:

$$\frac{\langle b, \sigma \rangle \rightarrow false}{\langle \texttt{while}\ \ b\ \texttt{do}\ \ c, \sigma \rangle \rightarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \rightarrow true \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \texttt{while}\ \ b\ \texttt{do}\ \ c, \sigma'' \rangle \rightarrow \sigma'}{\langle \texttt{while}\ \ b\ \texttt{do}\ \ c, \sigma \rangle \rightarrow \sigma'}$$

equivalence of commands

$$c_0 \sim c_1 \text{ iff } (\forall \sigma, \sigma' \in \sum . \langle c_0, \sigma \rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma')$$

Proposition

$$
\begin{aligned}
w &\equiv \texttt{while } b \texttt{ do } c \\
w' &\equiv \texttt{if } b \texttt{ then } c \texttt{ ; } w \texttt{ else skip}
\end{aligned}
\quad \Rightarrow \quad w \sim w'
$$

Proof:

we want to show that:

$$(\forall \sigma, \sigma' \in \sum . \ \langle w, \sigma \rangle \to \sigma' \ \Leftrightarrow \ \langle w', \sigma \rangle \to \sigma')$$

Let σ, σ' be arbitrary elements in $\sum$

$$\langle w, \sigma \rangle \to \sigma' \;\Rightarrow\; \langle w', \sigma \rangle \to \sigma$$

consider the derivation tree of $\langle w, \sigma \rangle \to \sigma'$

last rule

(i) **or** (ii)

$$\frac{\langle b,\sigma\rangle \to false}{\langle w,\sigma\rangle \to \sigma} \qquad \frac{\langle b,\sigma\rangle \to true \quad \langle c,\sigma\rangle \to \sigma'' \quad \langle w,\sigma''\rangle \to \sigma'}{\langle w,\sigma\rangle \to \sigma'}$$

case (i)

$$\frac{\dfrac{\vdots\;\pi}{\langle b,\sigma\rangle \to false}}{\langle w,\sigma\rangle \to \sigma}$$

by using π we obtain

$$\frac{\dfrac{\vdots\;\pi}{\langle b,\sigma\rangle \to false} \qquad \overline{\langle \texttt{skip},\sigma\rangle \to \sigma}}{\langle \underbrace{\texttt{if } b \texttt{ then } c \texttt{ ; } w \texttt{ else skip}}_{w'},\sigma\rangle \to \sigma}$$

case (ii)

$$\dfrac{\vdots\,\pi_1}{\langle b,\sigma\rangle\to true}\qquad\dfrac{\vdots\,\pi_2}{\langle c,\sigma\rangle\to\sigma''}\qquad\dfrac{\vdots\,\pi_3}{\langle w,\sigma''\rangle\to\sigma'}$$

by using π_2, π_3 we obtain

$$\dfrac{\dfrac{\vdots\,\pi_2}{\langle c,\sigma\rangle\to\sigma''}\qquad\dfrac{\vdots\,\pi_3}{\langle w,\sigma''\rangle\to\sigma'}}{\langle c\ ;\ w,\sigma\rangle\to\sigma'}$$

and therefore

$$\dfrac{\dfrac{\vdots\,\pi_1}{\langle b,\sigma\rangle\to true}\qquad\dfrac{\dfrac{\vdots\,\pi_2}{\langle c,\sigma\rangle\to\sigma''}\qquad\dfrac{\vdots\,\pi_3}{\langle w,\sigma''\rangle\to\sigma'}}{\langle c\ ;\ w,\sigma\rangle\to\sigma'}}{\langle\underbrace{\texttt{if}\ b\ \texttt{then}\ c\ ;\ w\ \texttt{else skip}}_{w'},\sigma\rangle\to\sigma'}$$

$$\langle w', \sigma \rangle \to \sigma' \;\Rightarrow\; \langle w, \sigma \rangle \to \sigma'$$

consider the derivation tree of $\langle w', \sigma \rangle \to \sigma'$

the last rule is either

(i-b) $$\frac{\langle b,\sigma\rangle \to true \quad \langle c\ ;\ w,\sigma\rangle \to \sigma'}{\langle \texttt{if}\ b\ \texttt{then}\ c\ ;\ w\ \texttt{else}\ \texttt{skip},\sigma\rangle \to \sigma'}$$

or

(ii-b) $$\frac{\langle b,\sigma\rangle \to false \quad \langle \texttt{skip},\sigma\rangle \to \sigma}{\langle \texttt{if}\ b\ \texttt{then}\ c\ ;\ w\ \texttt{else}\ \texttt{skip},\sigma\rangle \to \sigma}$$

case (i-b)

the derivation is

$$\dfrac{\dfrac{\vdots\ \pi}{\langle b,\sigma\rangle\rightarrow true} \qquad \dfrac{\vdots\ \pi'}{\langle c\ ;\ w,\sigma\rangle\rightarrow\sigma'}}{\langle \texttt{if}\ b\ \texttt{then}\ c\ ;\ w\ \texttt{else}\ \texttt{skip},\sigma\rangle\rightarrow\sigma'}$$

where π’ is

$$\dfrac{\dfrac{\vdots\ \alpha}{\langle c,\sigma\rangle\rightarrow\sigma''} \qquad \dfrac{\vdots\ \beta}{\langle w,\sigma''\rangle\rightarrow\sigma'}}{\langle c\ ;\ w,\sigma\rangle\rightarrow\sigma'}$$

we conclude with

$$\dfrac{\dfrac{\vdots\ \pi}{\langle b,\sigma\rangle\rightarrow true} \qquad \dfrac{\vdots\ \alpha}{\langle c,\sigma\rangle\rightarrow\sigma''} \qquad \dfrac{\vdots\ \beta}{\langle w,\sigma''\rangle\rightarrow\sigma'}}{\langle w,\sigma\rangle\rightarrow\sigma'}$$

case (ii-b)

the derivation is

$$\dfrac{\dfrac{\vdots\ \pi}{\langle b,\sigma\rangle \to false} \quad \overline{\langle \mathtt{skip},\sigma\rangle \to \sigma}}{\langle \mathtt{if}\ b\ \mathtt{then}\ c\ ;\ w\ \mathtt{else}\ \mathtt{skip},\sigma\rangle \to \sigma}$$

we conclude with

$$\dfrac{\dfrac{\vdots\ \pi}{\langle b,\sigma\rangle \to false}}{\langle w,\sigma\rangle \to \sigma}$$

QED

$$w \equiv \texttt{while } 0 < \texttt{x do } (\texttt{y} := 2 * \texttt{y}; \texttt{x} := \texttt{x} - 1)$$

(a) Let $\sigma = \sigma[\texttt{x} \mapsto 2, \texttt{y} \mapsto 3]$. Find σ_* such that $\langle w, \sigma \rangle \to \sigma_*$ can be derived. Give complete derivation tree.

(b) Prove that if $\sigma(x) = a \geqslant 0$, $\sigma(y) = b$ and $\langle w, \sigma \rangle \to \sigma_*$ then $\sigma_*(y) = 2^a \cdot b$.

Let $\sigma_* = \sigma[\mathtt{y}\mapsto 12, \mathtt{x}\mapsto 0]$. The derivation of $\langle w, \sigma\rangle \to \sigma_*$ looks as follows

$$\dfrac{\dfrac{\langle \mathtt{0}, \sigma\rangle \to 0 \quad \langle \mathtt{x}, \sigma\rangle \to 2}{\langle \mathtt{0} < \mathtt{x}, \sigma\rangle \to \mathit{true}} \quad \dfrac{\boxed{\text{A}}}{\langle (\mathtt{y} := \mathtt{2} * \mathtt{y}; \mathtt{x} := \mathtt{x} - \mathtt{1}), \sigma\rangle \to \sigma_1} \quad \dfrac{\boxed{\text{B}}}{\langle w, \sigma_1\rangle \to \sigma_*}}{\langle w, \sigma\rangle \to \sigma_*}$$

where

$$\dfrac{\boxed{A}}{} \equiv \dfrac{\dfrac{\dfrac{\langle 2,\sigma\rangle \to 2 \quad \langle \mathtt{y},\sigma\rangle \to 3}{\langle 2 * \mathtt{y},\sigma\rangle \to 6}}{\langle \mathtt{y} := 2 * \mathtt{y},\sigma\rangle \to \sigma[\mathtt{y}\mapsto 6] = \sigma_2} \quad \dfrac{\dfrac{\langle \mathtt{x},\sigma_2\rangle \to 2 \quad \langle 1,\sigma_2\rangle \to 1}{\langle \mathtt{x} - 1,\sigma_2\rangle \to 1}}{\langle \mathtt{x} := \mathtt{x} - 1,\sigma_2\rangle \to \sigma_2[\mathtt{x}\mapsto 1]}}{}$$

hence $\sigma_1 = \sigma_2[\mathtt{x}\mapsto 1] = \sigma[\mathtt{y}\mapsto 6, \mathtt{x}\mapsto 1]$

$$\dfrac{\boxed{B}}{} \equiv \dfrac{\dfrac{\langle 0,\sigma_1\rangle \to 0 \quad \langle \mathtt{x},\sigma_1\rangle \to 1}{\langle 0 < \mathtt{x},\sigma_1\rangle \to \textit{true}} \quad \dfrac{\boxed{C}}{\langle (\ldots),\sigma_1\rangle \to \sigma_*} \quad \dfrac{\dfrac{\langle 0,\sigma_*\rangle \to 0 \quad \langle \mathtt{x},\sigma_*\rangle \to 0}{\langle 0 < \mathtt{x},\sigma_*\rangle \to \textit{false}}}{\langle w,\sigma_*\rangle \to \sigma_*}}{}$$

$$\frac{\boxed{C}}{} \equiv \frac{\dfrac{\dfrac{\langle 2, \sigma_1\rangle \to 2 \quad \langle \mathtt{y}, \sigma_1\rangle \to 6}{\langle 2 * \mathtt{y}, \sigma_1\rangle \to 12}}{\langle \mathtt{y := 2 * y}, \sigma_1\rangle \to \sigma_1[\mathtt{y}\mapsto 12] = \sigma_3} \quad \dfrac{\dfrac{\langle \mathtt{x}, \sigma_3\rangle \to 1 \quad \langle \mathtt{1}, \sigma_3\rangle \to 1}{\langle \mathtt{x - 1}, \sigma_3\rangle \to 0}}{\langle \mathtt{x := x - 1}, \sigma_3\rangle \to \sigma_3[\mathtt{x}\mapsto 0]}}{}$$

Observe that $\sigma_3[\mathtt{x}\mapsto 0] = \sigma_1[\mathtt{y}\mapsto 12, \mathtt{x}\mapsto 0] = \sigma_*$.

Big Step VS One Step

the evaluation and execution relations are
Big Step Relations

what about One Step Relation?

$$\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$$

a possible rule:

$$\frac{\langle b, \sigma \rangle \rightarrow_1 \langle \mathbf{true}, \sigma \rangle}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma \rangle \rightarrow_1 \langle c_0, \sigma \rangle}$$

fine 10 genn.

$$\frac{\langle a, \sigma \rangle \to n}{\langle \texttt{x := } a, \sigma \rangle \to_1 \langle \texttt{skip}, \sigma[x \mapsto n] \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \to_1 \langle c_0', \sigma' \rangle}{\langle c_0 \texttt{;}\, c_1, \sigma \rangle \to_1 \langle c_0' \texttt{;}\, c_1, \sigma' \rangle}$$

$$\langle \texttt{skip;}\, c_1, \sigma \rangle \to_1 \langle c_1, \sigma \rangle$$

$$\frac{\langle b, \sigma \rangle \to \texttt{true}}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1, \sigma \rangle \to_1 \langle c_0, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \texttt{false}}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1, \sigma \rangle \to_1 \langle c_1, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \texttt{true}}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to_1 \langle c; \texttt{while } b \texttt{ do } c, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \texttt{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to_1 \langle \texttt{skip}, \sigma \rangle}$$

We execute program $p \equiv \texttt{x := 7; y := 4;}\ w$, where

$$w \equiv \texttt{while not(x = y) do}$$
$$\texttt{if x < y then y := y - 1}$$
$$\texttt{else x := x - y}$$

We denote body of the loop by c.

$$
\begin{aligned}
& \langle \texttt{x := 7; y := 4; } w \, , \sigma \rangle \\
\rightarrow_1 & \langle \texttt{skip; y := 4; } w \, , \sigma[\mathrm{x} \mapsto 7] \rangle \\
\rightarrow_1 & \langle \texttt{y := 4; } w \, , \sigma[\mathrm{x} \mapsto 7] \rangle \\
\rightarrow_1 & \langle \texttt{skip; } w \, , \sigma[\mathrm{x} \mapsto 7, \mathrm{y} \mapsto 4] \rangle \\
\rightarrow_1 & \langle \texttt{while not(x = y) do } c \, , \sigma[\mathrm{x} \mapsto 7, \mathrm{y} \mapsto 4] \rangle \\
\rightarrow_1 & \langle \texttt{if x < y then y := y - 1 else x := x - y; } w \, , \sigma[\mathrm{x} \mapsto 7, \mathrm{y} \mapsto 4] \rangle \\
\rightarrow_1 & \langle \texttt{x := x - y; } w \, , \sigma[\mathrm{x} \mapsto 7, \mathrm{y} \mapsto 4] \rangle
\end{aligned}
$$

$$
\begin{aligned}
&\rightarrow_1 \langle \quad \texttt{skip;}\ w\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 4]\rangle \\
&\rightarrow_1 \langle \quad \texttt{while not(x = y) do}\ c\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 4]\rangle \\
&\rightarrow_1 \langle \texttt{if x < y then y := y - 1 else x := x - y;}\ w\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 4]\rangle \\
&\rightarrow_1 \langle \quad \texttt{y := y - 1;}\ w\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 4]\rangle \\
&\rightarrow_1 \langle \quad \texttt{skip;}\ w\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 3]\rangle \\
&\rightarrow_1 \langle \quad \texttt{while not(x = y) do}\ c\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 3]\rangle \\
&\rightarrow_1 \langle \quad \texttt{skip}\ , \sigma[\mathrm{x}\mapsto 3, \mathrm{y}\mapsto 3]\rangle
\end{aligned}
$$

Thm 1 If $\langle c, \sigma \rangle \rightarrow_1^* \langle \texttt{skip}, \sigma_* \rangle$ then $\langle c, \sigma \rangle \rightarrow \sigma_*$.

Thm 2 If $\langle c, \sigma \rangle \rightarrow \sigma_*$ then $\langle c, \sigma \rangle \rightarrow_1^* \langle \texttt{skip}, \sigma_* \rangle$.

PROOF *(of Theorem 1)* We assume that $\langle c, \sigma \rangle \rightarrow_1^k \langle \mathtt{skip}, \sigma_* \rangle$ for some k. The proof will go by induction on k.

From our assumption

$$\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle \text{ and } \langle c', \sigma' \rangle \rightarrow_1^{k-1} \langle \mathtt{skip}, \sigma_* \rangle$$

We may use our IH (for $k - 1$) to infer that $\langle c', \sigma' \rangle \rightarrow \sigma_*$. From $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$ and $\langle c', \sigma' \rangle \rightarrow \sigma_*$ it follows that $\langle c, \sigma \rangle \rightarrow \sigma_*$ (using the lemma).

For the base case $k = 1$ our assumption is $\langle c, \sigma \rangle \rightarrow_1 \langle \mathtt{skip}, \sigma_* \rangle$. Clearly $\langle \mathtt{skip}, \sigma_* \rangle \rightarrow \sigma_*$. We can use the lemma again to infer that $\langle c, \sigma \rangle \rightarrow \sigma_*$.

Lemma

If $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$ and $\langle c', \sigma' \rangle \rightarrow \sigma''$ then $\langle c, \sigma \rangle \rightarrow \sigma''$.

PROOF *(By induction over structure of command c)* There are 7 possible cases depending on which rule was used at the bottom of the derivation tree of $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$.

Case (i) $c \equiv \texttt{x := } a$ and the rule used was

$$\frac{\langle a, \sigma \rangle \to n}{\langle \texttt{x := } a, \sigma \rangle \to_1 \langle \texttt{skip}, \sigma[\texttt{x} \mapsto n] \rangle}$$

Hence $c' \equiv \texttt{skip}$, $\sigma' = \sigma[\texttt{x} \mapsto n]$ and we must have a derivation $\boxed{\text{A}}$ for $\langle a, \sigma \rangle \to n$. We assume that $\langle c', \sigma' \rangle \to \sigma''$ can be derived and this is possible only when $\sigma'' = \sigma' = \sigma[\texttt{x} \mapsto n]$. Therefore we can derive $\langle c, \sigma \rangle \to \sigma''$ as follows

$$\frac{\dfrac{\boxed{\text{A}}}{\langle a, \sigma \rangle \to n}}{\langle \texttt{x := } a, \sigma \rangle \to \sigma[\texttt{x} \mapsto n]}$$

Case (ii) $c \equiv c_0 \,;\, c_1$ and the rule used was

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma'\rangle}{\langle c_0 \,;\, c_1, \sigma\rangle \rightarrow_1 \langle c_0' \,;\, c_1, \sigma'\rangle}$$

Hence $c' \equiv c_0' \,;\, c_1$ and $\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma'\rangle$ is derivable. We assume that $\langle c', \sigma'\rangle \rightarrow \sigma''$ and the only possible derivation of that transition must look as follows:

$$\frac{\dfrac{\boxed{\text{A}}}{\langle c_0', \sigma'\rangle \rightarrow \sigma_1} \qquad \dfrac{\boxed{\text{B}}}{\langle c_1, \sigma_1\rangle \rightarrow \sigma''}}{\langle c_0' \,;\, c_1, \sigma'\rangle \rightarrow \sigma''}$$

Hence $\langle c_0', \sigma' \rangle \to \sigma_1$ is derivable. We know also that $\langle c_0, \sigma \rangle \to_1 \langle c_0', \sigma' \rangle$ is derivable and since c_0 is simpler than c we can use IH to infer that there exists a derivatin $\boxed{\text{C}}$ of $\langle c_0, \sigma \rangle \to \sigma_1$. Therefore we can derive $\langle c, \sigma \rangle \to \sigma''$ as follows:

$$\dfrac{\dfrac{\boxed{\text{C}}}{\langle c_0, \sigma \rangle \to \sigma_1} \qquad \dfrac{\boxed{\text{B}}}{\langle c_1, \sigma_1 \rangle \to \sigma''}}{\langle c_0 ; c_1, \sigma \rangle \to \sigma''}$$

-

Case (iii) $c \equiv \texttt{while}\ b\ \texttt{do}\ d$ and the rule used was

$$\frac{\langle b, \sigma \rangle \rightarrow \textit{true}}{\langle \texttt{while}\ b\ \texttt{do}\ d, \sigma \rangle \rightarrow_1 \langle d; \texttt{while}\ b\ \texttt{do}\ d, \sigma \rangle}$$

Hence $c' \equiv d;\, c$, $\sigma' = \sigma$ and we must have a derivation $\boxed{\text{A}}$ for $\langle b, \sigma \rangle \rightarrow$ *true*. The only possible way of deriving $\langle c', \sigma' \rangle \rightarrow \sigma''$ is

$$\frac{\dfrac{\boxed{\text{B}}}{\langle d, \sigma \rangle \rightarrow \sigma_1} \qquad \dfrac{\boxed{\text{C}}}{\langle c, \sigma_1 \rangle \rightarrow \sigma''}}{\langle d;\, c, \sigma \rangle \rightarrow \sigma''}$$

Hence we can use derivations $\boxed{\text{A}}$, $\boxed{\text{B}}$ and $\boxed{\text{C}}$ to derive $\langle c, \sigma \rangle \to \sigma''$ as follows:

$$\dfrac{\dfrac{\boxed{\text{A}}}{\langle b, \sigma \rangle \to \textit{true}} \quad \dfrac{\boxed{\text{B}}}{\langle d, \sigma \rangle \to \sigma_1} \quad \dfrac{\boxed{\text{C}}}{\langle c, \sigma_1 \rangle \to \sigma''}}{\langle c, \sigma \rangle \to \sigma''}$$

Proof Structure /Case (ii)/

1. $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$ derivable *(assumption)*
2. $\langle c', \sigma' \rangle \rightarrow \sigma''$ derivable *(assumption)*
3. $c \equiv c_0 ; c_1$ and the only applicable rule was used to derive (1) *(case assumption)*
4. $c' \equiv c'_0 ; c_1$ and $\langle c_0, \sigma \rangle \rightarrow_1 \langle c'_0, \sigma' \rangle$ is derivable *(from 3)*
5. $\langle c'_0, \sigma' \rangle \rightarrow \sigma_1$ can be derived and
6. $\langle c_1, \sigma_1 \rangle \rightarrow \sigma''$ can be derived *(from 2 and 4)*
7. $\langle c_0, \sigma \rangle \rightarrow \sigma_1$ can be derived *(from 4,5 and IH)*
8. $\langle c, \sigma \rangle \rightarrow \sigma''$ can be derived *(from 7 and 6, QED)*

Proof Structure /Case (iii)/

1. $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ derivable *(assumption)*
2. $\langle c', \sigma' \rangle \to \sigma''$ derivable *(assumption)*
3. $c \equiv (\texttt{while}\ b\ \texttt{do}\ d)$ and the first applicable rule was used to derive (1) *(case assumption)*
4. $c' \equiv (d; c)$, $\sigma' = \sigma$ and
5. $\langle b, \sigma \rangle \to$ *true* is derivable *(from 3)*
6. $\langle d, \sigma \rangle \to \sigma_1$ and $\langle c, \sigma_1 \rangle \to \sigma''$ are derivable *(from 2 and 4)*
7. $\langle c, \sigma \rangle \to \sigma''$ can be derived *(from 5 and 6, QED)*

Semantica

induction

Induction

- how to prove that $\sum_{i=0}^{n} i = \frac{1}{2}n(n+1), \forall n \geq 0$?

an answer: by induction!

In order to prove that $\forall n \in \omega . P(n)$

- basis: prove that *P*(0)
- induction step: prove that

$$\forall m \in \omega .\ P(m) \Rightarrow P(m+1)$$

The Principle of Induction (IND)

$$(P(0) \& (\forall m \in \omega . P(m) \Rightarrow P(m+1)) \Rightarrow \forall n \in \omega . P(n)$$

Course-of-values induction (C-IND)

$$(\forall m \in \omega.[(\forall k < m.Q(k)) \Rightarrow Q(m)]) \Rightarrow \forall n \in \omega.Q(n)$$

IND è equivalente a C-IND

IND $\Rightarrow$ C–IND

Let $T(u)$ be $\forall x < u.Q(x)$

Let us suppose that

$\forall m \in \omega.(\forall k < m.Q(k)) \Rightarrow Q(m)$

we want to show that:

$\forall v \in \omega.Q(v)$

Observe that $\forall m.T(m) \Rightarrow T(m+1)$

and $T(0)$ are true.

By means of IND we conclude that

$\forall v \in \omega.T(v)$ is true and therefore $\forall v \in \omega.Q(v)$

C–IND $\Rightarrow$ IND

Let us suppose that

1)$P(0)$

and

2) $(\forall m \in \omega . P(m) \Rightarrow P(m+1))$

we want to show that

$\forall v \in \omega . P(v)$

Let $\alpha(m)$ be $(\forall k < m . P(k)) \Rightarrow P(m)$

We want to show that for each $m \in \omega$ $\alpha(m)$ is true.

By cases:

- if m is 0 then $\alpha(m)$ is equivalent to $P(0)$;
- if m is $n+1$ then if $(\forall k < n+1 . P(k))$ we have $P(n)$; by means of (2) we conclude that $P(n+1)$ is true and therefore $\alpha(n+1)$ is true.

By means of C–IND we conclude.

Structural Induction

Principle: The induction is based on the structure of the elements.

First, show that the property holds for all *atomic* elements

Second, show that the *formation rules* to build *non-atomic* elements *preserve* the property

Example: in order to show that a property P holds for all the arithmetic expressions it it is sufficient to show that"

$(\forall m \in \boldsymbol{N}.\ P(m)) \wedge$

$(\forall X \in \boldsymbol{Loc}.\ P(X)) \wedge$

$(\forall a_0, a_1 \in \boldsymbol{Aexp}.\ P(a_0) \wedge P(a_1) \Rightarrow P(a_0 + a_1)) \wedge$

$(\forall a_0, a_1 \in \boldsymbol{Aexp}.\ P(a_0) \wedge P(a_1) \Rightarrow P(a_0 - a_1)) \wedge$

$(\forall a_0, a_1 \in \boldsymbol{Aexp}.\ P(a_0) \wedge P(a_1) \Rightarrow P(a_0 \times a_1))$

Well Founded Relation

Well-founded relation. A well-founded relation is a *binary relation* $\prec$ on a set A such that there are no infinite descending chains $\cdots a_i \prec \cdots \prec a_1 \prec a_0$. For two elements a and b in A, if $a \prec b$, then we say that a is a *predecessor* of b.

Hence, a well-founded relation on A is such that no element of A has an infinite number of predecessors.

Note. A well-founded relation is necessarily *irreflexive*. That is, there is no $a \in A$ such that $a \prec a$.

Notation. In the sequel, we shall use $\preceq$ for the reflexive closure of $\prec$. That is, for $a, b \in A$, $a \preceq b \Leftrightarrow a = b$ or $a \prec b$ for

Well-Founded Induction (W-IND)

$$(\forall a \in A.((\forall b \prec a.P(b))) \Rightarrow P(b))) \rightarrow \forall a \in A.P(a)$$

Observation. Note that mathematical induction, course-of-values induction and structural induction are both special cases of well-founded induction

Proposition. Let $\prec$ be a binary relation on a set A. The relation $\prec$ is well-founded **if and only if** *any non-empty* subset Q of A has a *minimal element*. More formally,

$$(\forall Q \subseteq A.\ (Q \neq \emptyset \Rightarrow (\exists m \in Q.\ (\forall b \prec m.\ b \notin Q))))$$

Induction on derivation trees

- define the size $\#D$ of derivation D:

1. if D is an axiom S then $\#D$=0
2. id D is

$$\frac{\begin{array}{c}\vdots\, D_0\\ S_0\end{array} \qquad \cdots \quad \begin{array}{c}\vdots\, D_k\\ S_k\end{array}}{S}\; r$$

then $$\#D = sup\{\#D_i + 1 | i \leq k\}$$

fine 11 gen

remember that....subderivations are subtrees!

Rule instance. A rule instance is a pair (X/y), where X (resp. y) is a finite set of premises (resp. the conclusion) of the rule instance

Set of rule instances R: set of pairs (X/y)

Definition. An *R-derivation* of y is either

$$(\emptyset/y) \text{ or}$$

$$(\{d_1, \cdots, d_n\}/y)$$

where $(\{x_1, \cdots, x_n\}/y)$ is a rule instance and d_i is an *R-derivation* of x_i, $(1 \leq i \leq n)$

$d \Vdash_R y$ to mean "d is an *R-derivation* of y"

$$(\emptyset/y) \Vdash_R y \text{ if } (\emptyset/y) \in R$$

$$(\{d_1, \cdots, d_n\}/y) \Vdash_R y$$

$$\text{if } ((\{x_1, \cdots, x_n\}/y) \in R) \ \wedge \ (\bigwedge_{i=1}^{n}(d_i \Vdash_R x_i))$$

Immediate subderivation: We say that d' is an immediate subderivation of d and we write $d' \prec_1 d$ if and only if d has the form (D/y) with $d' \in D$

we denote the *transitive closure* of $\prec_1$ by $\prec$

We say that d' is a *proper subderivation* of d iff $d' \prec d$

Reminder. *Transitive closure* of a relation r on a set X is

$$r^+ = \bigcup_{k \in \omega} r^{k+1}$$

where $r^0 = Id_X$ is the identity relation on X, and for $k > 0$, $r^k = \underbrace{r \circ r \circ \cdots \circ r}_{k\ times}$

The *transitive, reflexive closure* of r is $r^* = r^+ \cup Id_X$

Note. $\prec_1$ and $\prec$ are well-founded because derivations are finite

Semantica

denotational semantics

Denotational Semantics of IMP

- mathematical meaning of syntactic objects
- the meaning of syntactic objects is given by suitable functions

Notation: We shall use $[\![\,]\!]$ around an argument of a semantic function

Vocabulary: Given x a syntactic object, $\mathcal{F}$ a semantic function, x is said to *denote* $\mathcal{F}[\![x]\!]$ and $\mathcal{F}[\![x]\!]$ is said to be a *denotation*, *meaning* of x

the informal idea:

Let $a \in \boldsymbol{Aexp}$

a represents a function that maps a state σ to a $n \in \boldsymbol{Z}$

Let $b \in \boldsymbol{Bexp}$

b represents a function that maps a state σ to a $t \in \boldsymbol{T} = \{false, true\}$

Let $c \in \boldsymbol{Com}$

represents a function that maps a state σ to a state σ'

what about non termination?

$$\mathcal{A} \quad : \quad \boldsymbol{Aexp} \quad \rightarrow \quad (\sum \rightarrow \boldsymbol{Z})$$

$$\mathcal{A}[\![\mathtt{n}]\!]\sigma = n$$

$$\mathcal{A}[\![X]\!]\sigma = \sigma(X)$$

$$\mathcal{A}[\![a_0 + a_1]\!]\sigma = \mathcal{A}[\![a_0]\!]\sigma \; + \; \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{A}[\![a_0 - a_1]\!]\sigma = \mathcal{A}[\![a_0]\!]\sigma \; - \; \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{A}[\![a_0 \times a_1]\!]\sigma = \mathcal{A}[\![a_0]\!]\sigma \; \times \; \mathcal{A}[\![a_1]\!]\sigma$$

give the semantics of:

`3 + 5`

`11 - 3`

$X{+}Y$ in state σ such that $\sigma(X) = 6$, $\sigma(Y) = 9$

$X{\times}Y$ in state σ such that $\sigma(X) = 2$, $\sigma(Y) = 3$

$$\mathcal{B} \quad : \quad \boldsymbol{Bexp} \quad \rightarrow \quad (\sum \rightarrow \boldsymbol{T})$$

$$\mathcal{B}[\![\texttt{false}]\!]\sigma \quad = \quad false$$

$$\mathcal{B}[\![\texttt{true}]\!]\sigma \quad = \quad true$$

$$\mathcal{B}[\![a_0 \texttt{ = } a_1]\!]\sigma \quad = \quad true \quad \text{when} \quad \mathcal{A}[\![a_0]\!]\sigma \quad = \quad \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{B}[\![a_0 \texttt{ = } a_1]\!]\sigma \quad = \quad false \quad \text{when} \quad \mathcal{A}[\![a_0]\!]\sigma \quad \neq \quad \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{B}[\![a_0 \leq a_1]\!]\sigma \quad = \quad true \quad \text{when} \quad \mathcal{A}[\![a_0]\!]\sigma \quad \leq \quad \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{B}[\![a_0 \leq a_1]\!]\sigma \quad = \quad false \quad \text{when} \quad \mathcal{A}[\![a_0]\!]\sigma \quad \not\leq \quad \mathcal{A}[\![a_1]\!]\sigma$$

$$\mathcal{B}[\![b_0 \wedge b_1]\!]\sigma \quad = \quad \mathcal{B}[\![b_0]\!]\sigma \ \wedge \ \mathcal{B}[\![b_1]\!]\sigma$$

$$\mathcal{B}[\![b_0 \vee b_1]\!]\sigma \quad = \quad \mathcal{B}[\![b_0]\!]\sigma \ \vee \ \mathcal{B}[\![b_1]\!]\sigma$$

$$\mathcal{B}[\![\neg b]\!]\sigma \quad = \quad \neg\mathcal{B}[\![b]\!]\sigma$$

$$\mathcal{C} \quad : \quad \boldsymbol{Com} \quad \rightarrow \quad (\Sigma \rightharpoonup \Sigma)$$

set of partial functions

$$\begin{aligned} \mathcal{C}[\![\texttt{skip}]\!]\sigma &= \sigma \\ \mathcal{C}[\![c_0 \texttt{ ; } c_1]\!]\sigma &= \mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) \end{aligned}$$

$$\mathcal{C}[\![\texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1]\!]\sigma = $$

$$= \begin{cases} \mathcal{C}[\![c_0]\!]\sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = true \\ \mathcal{C}[\![c_1]\!]\sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = false \end{cases}$$

fine 17 genn

$$w \equiv \texttt{while}\ b\ \texttt{do}\ c$$

we know that:

$$w \sim \texttt{if}\ b\ \texttt{then}\ c\ ;\ w\ \texttt{else}\ \texttt{skip}$$

$$\mathcal{C}[\![w]\!]\sigma = \begin{cases} \mathcal{C}[\![c\ ;\ w]\!]\sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = true \\ \sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = false \end{cases}$$

using the semantics of composition we have

$$\mathcal{C}[\![w]\!]\sigma = \begin{cases} \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!]\sigma) & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = true \\ \sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = false \end{cases}$$

How can we determine $\mathcal{C}[\![w]\!]$

(the currently unknown meaning of `while` b `do` c) ?

$$
\begin{aligned}
\Gamma : (\Sigma \rightharpoonup \Sigma) \quad &\to \quad (\Sigma \rightharpoonup \Sigma) \\
f \quad &\mapsto \quad \Gamma(f)
\end{aligned}
$$

$$
\Gamma(f) \quad : \quad \Sigma \quad \rightharpoonup \quad \Sigma
$$

$$
\sigma \quad \mapsto \quad [\Gamma(f)](\sigma) \;=\; \begin{cases} f(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}
$$

$$
g \equiv \mathcal{C}[\![c]\!] \text{ and } \beta \equiv \mathcal{B}[\![b]\!]
$$

se Γ(ff) = ff allora
ff e' punto fisso
e ff e' la
semantica del while

$$\sigma \quad \mapsto \quad f_1(\sigma) \; = \; \begin{cases} f_0(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

$$\sigma \quad \mapsto \quad f_2(\sigma) \; = \; \begin{cases} f_1(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

$$= \begin{cases} f_0(g^2(\sigma)) & \text{if } \beta(\sigma) \; \wedge \; \beta(g(\sigma)) \\ g(\sigma) & \text{if } \beta(\sigma) \; \wedge \; \neg\beta(g(\sigma)) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

$$\sigma \quad \mapsto \quad f_3(\sigma) = \begin{cases} f_2(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

$$= \begin{cases} f_0(g^3(\sigma)) & \text{if } \beta(\sigma) \ \wedge \ \beta(g(\sigma)) \ \wedge \ \beta(g^2(\sigma)) \\ g^2(\sigma) & \text{if } \beta(\sigma) \ \wedge \ \beta(g(\sigma)) \ \wedge \ \neg\beta(g^2(\sigma)) \\ g(\sigma) & \text{if } \beta(\sigma) \ \wedge \ \neg\beta(g(\sigma)) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

$$
f_{n+1}(\sigma) = \begin{cases} f_0(g^{n+1}(\sigma)) & \text{if } \bigwedge_{i=0}^{n} \beta(g^i(\sigma)) \\ g^n(\sigma) & \text{if } \bigwedge_{i=0}^{n-1} \beta(g^i(\sigma)) \ \wedge \ \neg\beta(g^n(\sigma)) \\ \ldots & \\ g(\sigma) & \text{if } \beta(\sigma) \ \wedge \ \neg\beta(g(\sigma)) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}
$$

$$w \equiv \texttt{while}\ b\ \texttt{do}\ c$$

$$\mathcal{C}[\![w]\!] =_{\mathrm{def}} \bigcup_{n\in\omega} f_n = \bigcup_{n\in\omega} \Gamma^n(\emptyset)$$

$$\mathcal{C}[\![w]\!]\sigma = \begin{cases} \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!]\sigma) & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = true \\ \sigma & \text{if} \quad \mathcal{B}[\![b]\!]\sigma = false \end{cases}$$

$$\sigma \quad \mapsto \quad [\Gamma(f)](\sigma) \ = \begin{cases} f(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{if } \neg\beta(\sigma) \end{cases}$$

Equivalence of Denotational and Operational semantics

Lemma: For any $a \in \boldsymbol{Aexp}$,

$$\mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$$

proof: induction on a

Lemma: For any $b \in \boldsymbol{Bexp}$,

$$\mathcal{B}[\![b]\!] = \{(\sigma, t) \mid \langle b, \sigma \rangle \to t\}$$

proof: induction on b

Lemma: For any command c and states σ, σ', we have

$$\langle c, \sigma \rangle \to \sigma' \Rightarrow (\sigma, \sigma') \in \mathcal{C}[\![c]\!]$$

Proof

The proof is by rule-induction on the operational semantics of commands

For $c \in \boldsymbol{Com}, \sigma, \sigma' \in \sum$, we define

$$P(c, \sigma, \sigma') \ \equiv \ (\sigma, \sigma') \in \mathcal{C}[\![c]\!]$$

we prove that:

$\langle c, \sigma \rangle \to \sigma' \Rightarrow P(c, \sigma, \sigma')$ for any command c and states σ, σ'

we verify only one clause: the case of $w \ \equiv \ \texttt{while}\ b\ \texttt{do}\ c$

$$\langle w, \sigma \rangle \to \sigma' \Rightarrow P(w, \sigma, \sigma')$$

$\langle w, \sigma \rangle \to \sigma'$

$\Rightarrow$ { derivation rules for commands}

$$(i) \quad \frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to true} \quad \dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \quad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle w,\sigma\rangle \to \sigma'}$$

$$\text{or} \quad (ii) \quad \frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to false}}{\langle w,\sigma\rangle \to \sigma'}$$

case (ii)

$$\langle b, \sigma \rangle \rightarrow false$$

$\Rightarrow$ { Lemma (9) }

$$\mathcal{B}[\![b]\!](\sigma) = false$$

$\Rightarrow$ { definition of $\mathcal{C}[\![w]\!]$ and the above line }

$$\mathcal{C}[\![w]\!](\sigma) = \sigma' = \sigma \text{ i.e. } (\sigma, \sigma) \in \mathcal{C}[\![w]\!]$$

$\Rightarrow$ { definition of P }

$$P(w, \sigma, \sigma) \text{ holds}$$

case (i)

$$\langle b, \sigma \rangle \rightarrow true \ \wedge \ \langle c, \sigma \rangle \rightarrow \sigma'' \ \wedge \ P(c, \sigma, \sigma'')$$
$$\wedge \ \langle w, \sigma'' \rangle \rightarrow \sigma' \ \wedge \ P(w, \sigma'', \sigma')$$

$\Rightarrow$ { Lemma (9) and the above line }

$$\mathcal{B}[\![b]\!](\sigma) = true$$

$\Rightarrow$ { Definition of P and the above line }

$$\mathcal{B}[\![b]\!](\sigma) = true \ \wedge \ \mathcal{C}[\![c]\!](\sigma) = \sigma'' \ \wedge \ \mathcal{C}[\![w]\!](\sigma'') = \sigma'$$

$\Rightarrow$ { Definition of $\mathcal{C}[\![w]\!]$ when $\mathcal{B}[\![b]\!] = true$, the above line }

$$\mathcal{C}[\![w]\!](\sigma) = \mathcal{C}[\![c; w]\!](\sigma) = \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!](\sigma)) = \mathcal{C}[\![w]\!](\sigma'') = \sigma'$$

$\Rightarrow$ { Definition of P }

$$P(w, \sigma, \sigma')$$

Theorem: For any command c, we have

$$\mathcal{C}[\![c]\!] = \{(\sigma, \sigma') \mid \langle c, \sigma \rangle \to \sigma'\}$$

Equivalently,

$$(\sigma, \sigma') \in \mathcal{C}[\![c]\!] \Leftrightarrow \langle c, \sigma \rangle \to \sigma'$$

fine 24
genn

we have proved the "only if" part

proof by induction on c

$c \equiv$ `skip`

$$(\sigma, \sigma') \in \mathcal{C}[\![\texttt{skip}]\!]$$

$\Rightarrow$ { Definition of $\mathcal{C}[\![\texttt{skip}]\!]$ }

$$\sigma = \sigma'$$

$\Rightarrow$ { execution rule for `skip` }

$$\langle \texttt{skip}, \sigma \rangle \to \sigma'$$

$c \equiv \; X := \; a$

$$(\sigma, \sigma') \in \mathcal{C}[\![X := \; a]\!]$$

$\Rightarrow$ { definition of $\mathcal{C}[\![X := \; a]\!]$ }

$$\mathcal{A}[\![a]\!](\sigma) = n \;\wedge\; \sigma' = \sigma[n/X]$$

$\Rightarrow$

$$\langle a, \sigma \rangle \rightarrow n$$

$\Rightarrow$

$$\langle c, \sigma \rangle \rightarrow \sigma'$$

$c \equiv \textbf{while } b \textbf{ do } c_0$

$\mathcal{C}[\![c]\!] = fix(\Gamma)$ let $g \equiv \mathcal{C}[\![c_0]\!]$ and $\beta \equiv \mathcal{B}[\![b]\!]$

$$\Gamma(f)(\sigma) = \begin{cases} f(g(\sigma)) & \text{if } \beta(\sigma) \\ \sigma & \text{otherwise} \end{cases}$$

$$f_0 = \Gamma^0(\emptyset) = \emptyset$$

$$\forall n \in \omega . f_{n+1} = \Gamma(f_n) = \Gamma^{n+1}(\emptyset)$$

$$f_{n+1}(\sigma) = \Gamma(f_n)(\sigma) = \begin{cases} f_n(g(\sigma)) & \text{if } \beta(\sigma) = \textbf{true} \\ \sigma & \text{otherwise} \end{cases}$$

$$fix(\Gamma) = \bigcup_{n \in \omega} f_n$$

In order to show that

$$fix(\Gamma)(\sigma) = \sigma' \Rightarrow \langle c, \sigma \rangle \rightarrow \sigma'$$

we show by induction that:

$$\forall n. \forall \sigma, \sigma'. f_n(\sigma) = \sigma' \Rightarrow \langle c, \sigma \rangle \rightarrow \sigma'$$

base $n = 0$: trivial

induction step if $f_{n+1}(\sigma) = \sigma'$ we have two cases:

1. $\beta(\sigma) = \mathbf{true}:$
   by a previous lemma we have $\langle b, \sigma \rangle \rightarrow \mathbf{true}$ and
   by definition of f_i's, $f_n(g(\sigma)) = \sigma'$
   by induction hypothesis $\langle c, g(\sigma) \rangle \rightarrow \sigma'$
   let $\mathcal{C}[\![c_0]\!](\sigma) = \sigma''$
   by structural ind-hyp $\langle c_0, \sigma \rangle \rightarrow \sigma''$
   summarizing we have:

   $$\langle b, \sigma \rangle \rightarrow \mathbf{true}, \langle c, \sigma'' \rangle \rightarrow \sigma', \langle c_0, \sigma \rangle \rightarrow \sigma''$$

   and by means of the rule of **while** $\langle c, \sigma \rangle \rightarrow \sigma'$
2. $\beta(\sigma) = \mathbf{false}:$
   by a previous lemma we have $\langle b, \sigma \rangle \rightarrow \mathbf{false}$ and
   by definition of f_i's, $\sigma' = \sigma$
   and by means of the rule of **while** $\langle c, \sigma \rangle \rightarrow \sigma'$