Small-Step Semantics

①

wenn ihr Abkürzungen wie $s_w, \dots$ verwenden wollt, müssen die definiert werden

Let $s_w \equiv$ while $x>0$ do $y := y*2 ; x := x-1$ end

$s_b \equiv y := y*2 ; x := x-1$

Idee: Tree zuerst, um nächsten Step zu finden

this step is justified by this tree

$$\frac{\dfrac{}{\langle y := 1, \sigma\rangle \to \sigma[y \mapsto 1]}\,\text{Ass}}{\langle s, \sigma\rangle \to_1 \langle s_w, \sigma[y \mapsto 1]\rangle}\,\text{Seq1}$$

$\langle s, \sigma\rangle$

$\to_1 \langle s_w, \sigma[y \mapsto 1]\rangle$

$\to_1 \langle$ if $x>0$ then $(s_b ; s_w)$ else skip end, $\sigma[y \mapsto 1]\rangle$

$\to_1 \langle s_b ; s_w, \sigma[y \mapsto 1]\rangle$

$\to_1 \langle x := x-1 ; s_w, \sigma[y \mapsto 2]\rangle$

$\to_1 \langle s_w, \sigma[y \mapsto 2][x \mapsto 1]\rangle$ (by Seq1)

$\to_1 \dots$

$$\frac{}{\langle s_w, \sigma[y \mapsto 1]\rangle \to_1 \langle \text{if } x>0 \text{ then } (s_b ; s_w) \text{ else skip end}, \sigma[y \mapsto 1]\rangle}\,\text{Wh}$$

$$\frac{}{\langle \text{if } x>0 \text{ then } (s_b ; s_w) \text{ else skip end}, \sigma[y \mapsto 1]\rangle \to_1 \langle s_b ; s_w, \sigma[y \mapsto 1]\rangle}\,\text{IfT}$$

since $B[\![x>0]\!]\sigma[y \mapsto 1] = tt$

$$\frac{\dfrac{\dfrac{}{\langle y := y*2, \sigma[y \mapsto 1]\rangle \to \sigma[y \mapsto 2]}\,\text{Ass}}{\langle s_b, \sigma[y \mapsto 1]\rangle \to_1 \langle x := x-1, \sigma[y \mapsto 2]\rangle}\,\text{Seq1}}{\langle s_b ; s_w, \sigma[y \mapsto 1]\rangle \to_1 \langle x := x-1 ; s_w, \sigma[y \mapsto 2]\rangle}\,\text{Seq2}$$

② "Only consider Ass$_{ns}$, Wh$_{ns}$ rules"

→ klingt nach Induction on the shape of the derivation tree

Was ist eig Aussage des Statements?

→ Wann immer ein derivation tree T mit $root(T) \equiv \langle s, \sigma \rangle \to \sigma'$ exist. (also Big-Step), dann gibt es finite derivation seq $\langle s, \sigma \rangle \to_1^* \sigma'$ (also small step)

Let $P(T) \equiv \forall \sigma, \sigma', s.\ (root(T) \equiv (\langle s, \sigma \rangle \to \sigma') \Rightarrow \langle s, \sigma \rangle \to_1^* \sigma')$

Prove $\forall T.\ P(T)$ by strong induct. on shape of deriv. tree T

For arbitrary T, prove $P(T)$ assuming IH: $\forall T' \subset T.\ P(T')$

wie immer

Let σ, σ', s arbitrary but fixed st. LHS of implic. holds

Case distinct. on last rule applied in T.

Case Ass$_{ns}$: We get $s \equiv x := e$ for some var x, arithm. expr. e

T is of the form ← 1.) wie sieht T aus, was lernen wir

$$\frac{}{\langle x := e, \sigma \rangle \to \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]}\ \text{Ass}_{ns}$$

So $\sigma' = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$

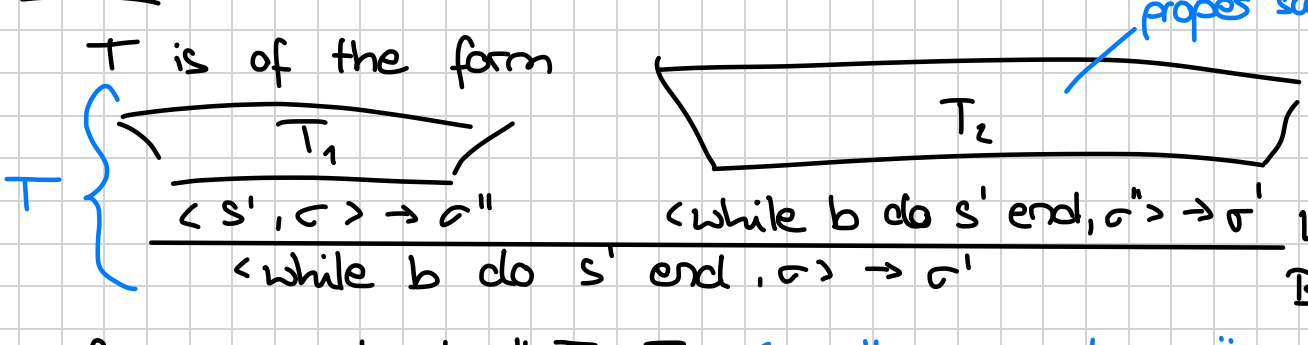
We construct ← 2. Konstruiere deriv. seq.

$$\frac{}{\langle x := e, \sigma \rangle \to_1 \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma}\ \text{Ass}_{sos}$$

and get $\langle s, \sigma \rangle \to_1^1 \sigma'$

Case While:

T is of the form



WHT, so $[\![b]\!]\sigma = tt$ *

for some $b, s', \sigma'', T_1, T_2$ ← alle neu, also müssen sie kurz erwähnt werden

s.t. that $s \equiv \text{while } b \text{ do } s' \text{ end}$

We know $T_1 \sqsubset T$, $T_2 \sqsubset T$, so $P(T_1)$, $P(T_2)$ hold per IH.

We get $\langle s', \sigma \rangle \to_1^* \sigma''$ and $\langle \text{while } b \text{ do } s' \text{ end}, \sigma'' \rangle \to_1^* \sigma'$

So for some k, we have $\langle s', \sigma \rangle \to_1^k \sigma''$.

↳ wenn ich hier noch an das Progr. was dranhänge, dann bin ich nach k Schritten genau am Anhängsel

By lecture:

$\langle s'; \text{while } b \text{ do } s' \text{ end}, \sigma \rangle \to_1^k \langle \text{while } b \text{ do } s' \text{ end}, \sigma'' \rangle$

We construct

$\langle s, \sigma \rangle$

$\to_1^1 \langle \text{if } b \text{ then } (s'; s) \text{ else skip end}, \sigma \rangle$ (by Wh)

$\to_1^1 \langle s'; s, \sigma \rangle$ (by IfT, since *)

$\to_1^* \langle s, \sigma'' \rangle$

$\to_1^* \sigma'$

Merke: manchmal machen wir einen Schritt, manchmal mehrere gleichzeitig

hinschreiben

(2.2) Analog: Was wollen wir hier überhaupt zeigen?

→ wenn k-step deriv. seq. exist., die in terminal config (σ') endet, dann können wir auch einen Big-Step Tree zeichnen, des das Programm ausführt & zum gleichen "Ergebnis" kommt

Wie? → k ist Länge des deriv. seq., da bietet sich Induktion darüber direkt an

Let $P(k) \equiv \forall \sigma, \sigma', s.\ (\langle s,\sigma\rangle \to_1^k \sigma' \Rightarrow \vdash \langle s,\sigma\rangle \to \sigma')$

Prove $\forall k.\ P(k)$ by strong mathematical induction on k (induction on length of deriv. seq.)

So for arbitrary k, prove P(k) assuming IH: $\forall k' < k.\ P(k')$

Let σ, σ', s arbitrary.

neu: Fall k=0 abdecken

For k=0 we get $\langle s,\sigma\rangle \to_1^0 \sigma'$, so $\langle s,\sigma\rangle = \sigma'$. Contradiction

So k>0, assume LHS of implication holds ← neu. hier unten, da für k=0 LHS nie hält

Unroll derivat. seq. once ← das ist neu: wir schauen uns den ersten Schritt genaues an

$\langle s,\sigma\rangle \to_1 \gamma \to_1^{k-1} \sigma'$ for some γ (γ ist Platzhalter für irgendeine Config)

Let T be the tree that justifies the first transition.

Case distinct. on last rule applied in T:

Case Ass$_{sos}$:

T is

$$\frac{}{\underbrace{\langle x := e,\ \sigma\rangle}_{\equiv s} \to \underbrace{\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]}_{\equiv \gamma}}\ \text{Ass}_{sos}$$

for some x,e ← kurz erwähnen, wie oben

γ is final state, so no further steps (k=1)

So $\sigma' = \gamma = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$

Construct

$$\frac{}{\langle x := e,\ \sigma\rangle \to \sigma'}\ \text{Ass}_{ns}$$

Case Seq 1, Seq:

oft bei induction on length of derivation seq: beide gleichzeitig abgedeckt

We get $root(T) \equiv \langle s_1; s_2, \sigma \rangle \to_1 \gamma$ for some s_1, s_2, γ, $s \equiv s_1; s_2$

Lemma from lecture for $\langle s_1; s_2, \sigma \rangle \to_n^k \sigma'$:

$\langle s_1, \sigma \rangle \to_1^{k_1} \sigma''$ and $\langle s_2, \sigma'' \rangle \to_1^{k_2} \sigma'$ for some σ'', k_1, k_2

w/ $k_1 + k_2 = k$. We get $k_1 > 0$, $k_2 > 0$ from def of $\to_n$

We get $k_1 < k$, $k_2 < k$ therefore $P(k_1)$, $P(k_2)$ from IH.

So these are trees T_1, T_2 w/ $root(T_1) \equiv \langle s_1, \sigma \rangle \to \sigma''$,

$root(T_2) \equiv \langle s_2, \sigma'' \rangle \to \sigma'$.

Construct

$$\dfrac{\overset{T_1}{\langle s_1, \sigma \rangle \to \sigma''} \qquad \overset{T_2}{\langle s_2, \sigma'' \rangle \to \sigma'}}{\langle s_1; s_2, \sigma \rangle \to \sigma'}\ Seq_{NS}$$

(3) Need to change entirety of WP to accomodate this stmt

→ Idea: States now contain addl. flag (do/don't exit loop)

If flag is set, all stmts loose their effect.

Exiting loop resets break flag

State' = {tt, ff} × State

Let $T, T_1, \ldots$ range over elems of State' ← for shorter notat. later

$$\frac{}{\langle \mathbf{break}, (q, \sigma)\rangle \to_1 (\mathrm{tt}, \sigma)}\ (\textsc{Break}_{SOS})$$

$$\frac{}{\langle x := e, (\mathrm{ff}, \sigma)\rangle \to_1 (\mathrm{ff}, \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma])}\ (\textsc{Ass}_{SOS})$$

$$\frac{}{\langle x := e, (\mathrm{tt}, \sigma)\rangle \to_1 (\mathrm{tt}, \sigma)}\ (\textsc{AssInBreak}_{SOS})$$

$$\frac{}{\langle \mathbf{skip}, \tau\rangle \to_1 \tau}\ (\textsc{Skip}_{SOS})$$

← need to change all SOS-rules, since we got a different representation of state

$$\frac{\langle s_1, \tau\rangle \to_1 \tau'}{\langle s_1; s_2, \tau\rangle \to_1 \langle s_2, \tau'\rangle}\ (\textsc{Seq1}_{SOS}) \qquad \frac{\langle s_1, \tau\rangle \to_1 \langle s_1', \tau'\rangle}{\langle s_1; s_2, \tau\rangle \to_1 \langle s_1'; s_2, \tau'\rangle}\ (\textsc{Seq2}_{SOS})$$

$$\frac{}{\langle \mathtt{if}\ b\ \mathtt{then}\ s_1\ \mathtt{else}\ s_2, \tau\rangle \to_1 \langle s_1, \tau\rangle}\ (\textsc{IfT}_{SOS}) \quad \mathcal{B}[\![b]\!]\sigma = \mathrm{tt}$$

$$\frac{}{\langle \mathtt{if}\ b\ \mathtt{then}\ s_1\ \mathtt{else}\ s_2, \tau\rangle \to_1 \langle s_2, \tau\rangle}\ (\textsc{IfF}_{SOS}) \quad \mathcal{B}[\![b]\!]\sigma = \mathrm{ff}$$

$$\frac{}{\langle \mathtt{while}\ b\ \mathtt{do}\ s\ \mathtt{end}, (v, \sigma)\rangle \to_1 (v, \sigma)}\ (\textsc{WhF}_{SOS}) \quad v = \mathrm{tt} \text{ or } \mathcal{B}[\![b]\!]\sigma = \mathrm{ff}$$

The above rules were all rather straightforward, and also similar to the previously defined NS rules. The next `while` rule, however, is quite different and makes use of an additional statement `leave` that is only used internally and must not occur anywhere else in the program.

$$\frac{}{\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, (\mathrm{ff}, \sigma)\rangle \to_1 \langle s; (\texttt{leave;while}\ b\ \texttt{do}\ s\ \texttt{end}), (\mathrm{ff}, \sigma)\rangle}\ (\textsc{WhT}_{SOS}) \quad \mathcal{B}[\![b]\!]\sigma = \mathrm{tt}$$

The `leave` statement is used as a marker to indicate that a loop is not only to be skipped – which would be possible already with $\textsc{WhF}_{SOS}$ – but also, that the flag must be reset to false, in order to not skip all of the remaining program statements:

$$\frac{}{\langle \texttt{leave;while}\ b\ \texttt{do}\ s\ \texttt{end}, (\mathrm{tt}, \sigma)\rangle \to_1 (\mathrm{ff}, \sigma)}\ (\textsc{LeaveT}_{SOS})$$

If the break flag is not set, then `leave` behaves just like `skip`, i.e. it does not exit the loop by skipping it:

$$\frac{}{\langle \texttt{leave;while}\ b\ \texttt{do}\ s\ \texttt{end}, (\mathrm{ff}, \sigma)\rangle \to_1 \langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, (\mathrm{ff}, \sigma)\rangle}\ (\textsc{LeaveF}_{SOS})$$