ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Session Sheet 12: Small Step Semantics

Assignment 1 (Applying Small-Step Semantics)

Let s be the following statement:

```
y := 1;
while x > 0 do
  y := y * 2;
  x := x - 1
end
```

Task. Let σ be a state with $\sigma(\mathrm{x}) = 2$. Prove that there is a state σ' with $\sigma'(\mathrm{y}) = 4$ such that $\langle s, \sigma\rangle \rightarrow_1^* \sigma'$ using the SOS rules of **IMP**.

Solution. Let s_b be the statement `y := y * 2; x := x - 1`, and let s_w be the statement `while x>0 do` s_b `end`. Then, we can derive:

$$
\begin{array}{rl}
 & \langle \texttt{y := 1; } s_w, \sigma\rangle \\
\rightarrow_1^1 & \langle s_w, \sigma[y \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle \texttt{if x>0 then } s_b \texttt{ ; } s_w \texttt{ else skip end}, \sigma[y \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle s_b \texttt{ ; } s_w, \sigma[y \mapsto 1]\rangle \\
\equiv & \langle \texttt{(y := y * 2; x := x - 1) ; } s_w, \sigma[y \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle \texttt{x := x - 1 ; } s_w, \sigma[y \mapsto 2]\rangle \\
\rightarrow_1^1 & \langle s_w, \sigma[y \mapsto 2][x \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle \texttt{if x>0 then } s_b \texttt{ ; } s_w \texttt{ else skip end}, \sigma[y \mapsto 2][x \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle \texttt{(y := y * 2; x := x - 1) ; } s_w, \sigma[y \mapsto 2][x \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle \texttt{x := x - 1 ; } s_w, \sigma[y \mapsto 4][x \mapsto 1]\rangle \\
\rightarrow_1^1 & \langle s_w, \sigma[y \mapsto 4][x \mapsto 0]\rangle \\
\rightarrow_1^1 & \langle \texttt{if x>0 then } s_b \texttt{ ; } s_w \texttt{ else skip end}, \sigma[y \mapsto 4][x \mapsto 0]\rangle \\
\rightarrow_1^1 & \langle \texttt{skip}, \sigma[y \mapsto 4][x \mapsto 0]\rangle \\
\rightarrow_1^1 & \sigma[y \mapsto 4][x \mapsto 0]
\end{array}
$$

The first four single-step transitions are justified by the following four derivation trees:

$$\dfrac{\dfrac{}{\langle \mathtt{y} := 1, \sigma\rangle \rightarrow_1 \sigma[\mathtt{y} \mapsto 1]}\ (\textsc{Ass}_{SOS})}{\langle \mathtt{y} := 1\,; s_w, \sigma\rangle \rightarrow_1 \langle s_w, \sigma[\mathtt{y} \mapsto 1]\rangle}\ (\textsc{Seq1}_{SOS})$$

$$\dfrac{}{\langle s_w, \sigma[\mathtt{y} \mapsto 1]\rangle \rightarrow_1 \langle \mathtt{if\ x} > 0\ \mathtt{then}\ (s_b\,; s_w)\ \mathtt{end}, \sigma[\mathtt{y} \mapsto 1]\rangle}\ (\textsc{While}_{SOS})$$

$$\dfrac{}{\langle \mathtt{if\ x} > 0\ \mathtt{then}\ (s_b\,; s_w)\ \mathtt{end}, \sigma[\mathtt{y} \mapsto 1]\rangle \rightarrow_1 \langle s_b\,; s_w, \sigma[\mathtt{y} \mapsto 1]\rangle}\ (\textsc{IfT}_{SOS})$$

Where the side condition for $\textsc{IfT}_{SOS}$ namely $\mathcal{B}[\![\mathtt{x} > 0]\!]\sigma[\mathtt{y} \mapsto 1] = \mathrm{tt}$ holds.

$$\dfrac{\dfrac{\dfrac{}{\langle \mathtt{y} := \mathtt{y} * 2, \sigma[\mathtt{y} \mapsto 1]\rangle \rightarrow_1 \sigma[\mathtt{y} \mapsto 2]}\ (\textsc{Ass}_{SOS})}{\langle \mathtt{y} := \mathtt{y} * 2\,; \mathtt{x} := \mathtt{x} - 1, \sigma[\mathtt{y} \mapsto 1]\rangle \rightarrow_1 \langle \mathtt{x} := \mathtt{x} - 1, \sigma[\mathtt{y} \mapsto 2]\rangle}\ (\textsc{Seq1}_{SOS})}{\langle (\mathtt{y} := \mathtt{y} * 2\,; \mathtt{x} := \mathtt{x} - 1)\,; s_w, \sigma[\mathtt{y} \mapsto 1]\rangle \rightarrow_1 \langle \mathtt{x} := \mathtt{x} - 1\,; s_w, \sigma[\mathtt{y} \mapsto 2]\rangle}\ (\textsc{Seq2}_{SOS})$$

Assignment 2 (Equivalence Lemma)

In this exercise, we consider the two lemmas from the lecture that formalize the equivalence of small-step and big-step semantics.

Task 2.1. We *partially* prove the following statement:

$$\forall \sigma, \sigma', s \cdot \vdash \langle s, \sigma\rangle \rightarrow \sigma' \implies \langle s, \sigma\rangle \rightarrow_1^* \sigma'$$

Here, we only consider the $\textsc{Ass}_{\mathrm{NS}}$-rule and the $\textsc{WhT}_{\mathrm{NS}}$-rule; the remaining cases are left for the exercise sheet.

Solution. We define

$$P(T) \equiv \forall \sigma, \sigma', s \cdot \big(\mathit{root}(T) \equiv (\langle s, \sigma\rangle \rightarrow \sigma') \implies \langle s, \sigma\rangle \rightarrow_1^* \sigma'\big)$$

and prove $\forall T \cdot P(T)$ by strong induction on the shape of the derivation tree T. Thus, for some arbitrary T, we get as induction hypothesis $\forall T' \sqsubset T \cdot P(T')$, and need to prove $P(T)$.

Let σ, σ', s be arbitrary. We assume $\mathit{root}(T) \equiv (\langle s, \sigma\rangle \rightarrow \sigma')$ and prove $\langle s, \sigma\rangle \rightarrow_1^* \sigma'$. The proof proceeds by case splitting on the last rule applied on T.

- **Case** $\textsc{Ass}_{\text{NS}}$: Then T is of the form:

$$\frac{}{\langle x \texttt{ := } e, \sigma\rangle \to \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]}\ (\textsc{Ass}_{NS})$$

  for some x, e such that $s \equiv x \texttt{ := } e$ and $\sigma' = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$. Now we can construct a derivation tree to justify $\langle s, \sigma\rangle \to_1^1 \sigma'$:

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

- **Case** $\textsc{WhT}_{\text{NS}}$: Then T is of the form

$$\frac{\dfrac{T_4}{\langle s', \sigma\rangle \to \sigma''} \qquad \dfrac{T_5}{\langle \texttt{while } b \texttt{ do } s' \texttt{ end}, \sigma''\rangle \to \sigma'}}{\langle \texttt{while } b \texttt{ do } s' \texttt{ end}, \sigma\rangle \to \sigma'}\ (\textsc{WhT}_{NS})$$

  for some $b, s', \sigma'', T_4, T_5$, such that $s \equiv \texttt{while } b \texttt{ do } s' \texttt{ end}$ and $\mathcal{B}[\![b]\!]\sigma = tt$.

  We apply (IH) twice. From $P(T_4)$ we learn $\langle s', \sigma\rangle \to_1^* \sigma''$. From $P(T_5)$ we learn $\langle \texttt{while } b \texttt{ do } s' \texttt{ end}, \sigma''\rangle \to_1^* \sigma'$. $\langle s', \sigma\rangle \to_1^* \sigma''$ gives us $\langle s', \sigma\rangle \to_1^k \sigma''$ for some k. We can apply the result of Assignment 3 from the *optional exercises* sheet on it to get

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

  We conclude this case with the following derivation sequence:

$$\begin{array}{ll}
 & \langle \texttt{while } b \texttt{ do } s' \texttt{ end}, \sigma\rangle \\
\to_1^1 & \langle \texttt{if } b \texttt{ then } (s'; \texttt{while } b \texttt{ do } s' \texttt{ end}) \texttt{ else skip}, \sigma\rangle \\
\to_1^1 & \langle (s'; \texttt{while } b \texttt{ do } s' \texttt{ end}), \sigma\rangle \\
\to_1^* & \langle \texttt{while } b \texttt{ do } s' \texttt{ end}, \sigma''\rangle \\
\to_1^* & \sigma'
\end{array}$$

  The second transition is justified by $\textsc{IfT}_{SOS}$, since $\mathcal{B}[\![b]\!]\sigma = tt$.

Task 2.2 We partially prove the following statement:

$$\forall \sigma, \sigma', s, k \cdot \langle s, \sigma\rangle \to_1^k \sigma' \implies \vdash \langle s, \sigma\rangle \to \sigma'$$

Here, we only consider the $\textsc{Ass}_{\text{SOS}}$-rule and the $\textsc{Seq1}_{\text{SOS}}$, and $\textsc{Seq2}_{\text{SOS}}$-rules; the remaining cases are left for the exercise sheet.

Solution. We define

$$Q(k) \equiv \forall \sigma, \sigma', s \cdot \big(\langle s, \sigma\rangle \to_1^k \sigma' \implies \vdash \langle s, \sigma\rangle \to \sigma'\big)$$

and prove $\forall k \cdot Q(k)$ by strong mathematical induction on k.

For arbitrary k assume $\forall k' < k \cdot Q(k')$ and prove $Q(k)$. Let σ, σ', s be arbitrary. Case splitting on the condition $k > 0$ immediately proves the case for $k = 0$ (the assumptions lead to $\langle s, \sigma \rangle \rightarrow_1^0 \sigma'$, which is a contradiction). So we are left with case $k > 0$. Assume $\langle s, \sigma \rangle \rightarrow_1^k \sigma'$ and prove $\vdash \langle s, \sigma \rangle \rightarrow \sigma'$.

We unroll the derivation sequence once to $\langle s, \sigma \rangle \rightarrow_1^1 \gamma \rightarrow_1^{k-1} \sigma'$. Let T be the derivation tree which justifies the first transition. We inspect the last rule applied to T.

- **Case** $\textsc{Ass}_{\mathrm{SOS}}$: Then T is of the form

  $$\frac{}{\langle x \ \texttt{:=} \ e, \sigma \rangle \rightarrow_1 \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]} \ (\textsc{Ass}_{SOS})$$

  for some x, e such that $s \equiv x \ \texttt{:=} \ e$ and $\gamma = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$. Since γ is a final state there is no further derivation sequence ($k = 1$), and hence $\sigma' = \gamma = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$. Now we can construct a derivation tree for $\langle x \ \texttt{:=} \ e, \sigma \rangle \rightarrow \sigma'$:

  $$\frac{}{\langle x \ \texttt{:=} \ e, \sigma \rangle \rightarrow \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]} \ (\textsc{Ass}_{NS})$$

- **Case** $\textsc{Seq1}_{\mathrm{SOS}}$, $\textsc{Seq2}_{\mathrm{SOS}}$: Then we must have *root*$(T) \equiv \langle s_1; s_2, \sigma \rangle \rightarrow_1 \gamma$ and hence $\vdash \langle s_1; s_2, \sigma \rangle \rightarrow_1 \gamma$ for some statements s_1, s_2, such that $s \equiv s_1; s_2$.

  Returning to our original assumption, we apply the lemma proven on the lecture slides on $\langle s_1; s_2, \sigma \rangle \rightarrow_1^k \sigma'$. We get $\langle s_1, \sigma \rangle \rightarrow_1^{k_1} \sigma''$ and $\langle s_2, \sigma'' \rangle \rightarrow_1^{k_2} \sigma'$, for some σ'', k_1, k_2, such that $k_1 + k_2 = k$.

  Note that $k_1 \neq 0$ and $k_2 \neq 0$ (otherwise, by the definition of $\rightarrow_1^0$ we would have to have a non-final configuration equal to a state, e.g. $\langle s_1, \sigma \rangle \equiv \sigma''$, which is impossible). Therefore, we must have $k_1 < k$ and $k_2 < k$.

  Since $k_1, k_2 < k$ we can apply the IH twice. From $Q(k_1)$ we learn $\vdash \langle s_1, \sigma \rangle \rightarrow \sigma''$ and from $Q(k_2)$ we learn $\vdash \langle s_2, \sigma'' \rangle \rightarrow \sigma'$. Let T_1, T_2 be the corresponding derivation trees, such that *root*$(T_1) \equiv \langle s_1, \sigma \rangle \rightarrow \sigma''$ and *root*$(T_2) \equiv \langle s_2, \sigma'' \rangle \rightarrow \sigma'$

  Now we can construct the derivation tree for $\vdash \langle s_1; s_2, \sigma \rangle \rightarrow \sigma'$ as follows:

  $$\frac{\overset{T_1}{\langle s_1, \sigma \rangle \rightarrow \sigma''} \qquad \overset{T_2}{\langle s_2, \sigma'' \rangle \rightarrow \sigma'}}{\langle s_1; s_2, \sigma \rangle \rightarrow \sigma'} \ (\textsc{Seq}_{NS})$$

Assignment 3 (`break` Statement)

In Assignment 4 of the *optional exercises* sheet 11, we defined big-step semantics rules for a `break` statement.

Task. Define small-step semantics rules for a `break` statement.

Solution. We again assume that `break` only occurs inside loop bodies, and extend the state with a flag that indicates whether the currently executed loop should be exited. We define a new set of states $State'$ that contains this additional flag:

$$State' = \{\mathrm{tt}, \mathrm{ff}\} \times State$$

Let $\tau, \tau', \ldots$ range over elements of set $State'$ and $\sigma, \sigma', \ldots$ over elements of set $State$.

The behavior of the `break` statement is to set this flag to true:

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

When the flag is true, this means that a break statement has been activated in the current while body. No change to the state must happen, until the flag is reset to false. This changes the rules for the assignment as follows:

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

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

Skipping should not change the state, regardless of the flag:

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

The sequential composition should behave as before, regardless of the flag. In case the flag is true, this propagates to the end of the sequential composition:

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

Conditionals are treated similarly, that is, the flag is simply propagated and the rule is otherwise unchanged (w.r.t. the standard SOS rules):

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

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

Note, that the rules for sequential composition and for conditionals could be optimised, in the sense that $s_1; s_2$ is skipped if the break flag is set, and similarly for the conditional.

Loops are skipped if the condition doesn't hold or if the break flag is set, which corresponds to the case that another while loop is inside the body of a while loop where the breakflag was set:

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

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:

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

Assignment 4 (Bonus: `do-times` Statements)

In a previous session we defined different NS rules for the IMP extension do e times s, where e is an arithmetic expression and s a statement.

Task. We now would like to define SOS rules for this type of loop.

Solution. There is nothing to be done if e does not evaluate to a positive integer:

$$\frac{}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to_1 \sigma}\,(\textsc{DoF}_{SOS}) \quad \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{ff}$$

Otherwise, we could either proceed by the rule

$$\frac{}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to_1 \langle \texttt{s;do } e\texttt{-1 times } s \texttt{ end}, \sigma\rangle}\,(\textsc{DoT}_{SOS}) \quad \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{tt}$$

which would result in an infinite derivation sequence for a loop such as

`do x times x := x+1 end`,

or we proceed by the rule

$$\frac{}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to_1 \langle \texttt{do } e\texttt{-1 times } s \texttt{ end;s}, \sigma\rangle}\,(\textsc{DoT}_{SOS}) \quad \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{tt}$$

which would result in a finite derivation sequence for the same loop (assuming the starting state maps x to a non-negative value).