ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Session Sheet 11: Big-Step Semantics

Assignment 1 (Substituting for Absent Variables)

Task: Prove that substituting absent variables has no effect, i.e., prove that

$$\forall e, e', x \cdot \big(x \notin \mathit{FV}(e) \implies e[x \mapsto e'] \equiv e\big).$$

Solution. Let x, e' be arbitrary. We define

$$P(e) \equiv \big(x \notin \mathit{FV}(e) \implies e[x \mapsto e'] \equiv e\big)$$

and prove $\forall e \cdot P(e)$ by strong structural induction on e. Thus, we have to prove $P(e)$ for some arbitrary arithmetic expression e and assume $\forall e'' \sqsubset e \cdot P(e'')$ as our induction hypothesis. We proceed by a case analysis on e:

- **Case** $e \equiv n$, for some numerical value n: By the definition of substitution, we have $n[x \mapsto e'] \equiv n$ from which the claim immediately follows.
- **Case** $e \equiv y$, for some variable y: We assume $x \notin \mathit{FV}(y)$ and seek to prove $y[x \mapsto e'] \equiv y$. Note that, since $\mathit{FV}(y) = \{y\}$, our assumption implies $x \not\equiv y$. Thus, by the definition of substitution, we have $y[x \mapsto e'] \equiv y$.
- **Case** $e \equiv e_1 \mathit{op} e_2$, for some arithmetic expressions e_1, e_2 and some arithmetic operator op: We assume $x \notin \mathit{FV}(e_1 \mathit{op} e_2)$ and seek to prove $(e_1 \mathit{op} e_2)[x \mapsto e'] \equiv e_1 \mathit{op} e_2$. Since $e_1 \sqsubset e$ and $e_2 \sqsubset e$, we can apply the induction hypothesis twice to get $P(e_1)$ and $P(e_2)$, i.e., that

  $$x \notin \mathit{FV}(e_1) \implies e_1[x \mapsto e'] \equiv e_1 \qquad \text{and} \qquad x \notin \mathit{FV}(e_2) \implies e_2[x \mapsto e'] \equiv e_2.$$

  Note that $\mathit{FV}(e_1 \mathit{op} e_2) = \mathit{FV}(e_1) \cup \mathit{FV}(e_2) \supseteq \mathit{FV}(e_1)$ and therefore, by our assumption, we have $x \notin \mathit{FV}(e_1)$. Analogously, we get $x \notin \mathit{FV}(e_2)$. Using these facts along our induction hypothesis, we obtain that $e_1[x \mapsto e'] \equiv e_1$ and $e_2[x \mapsto e'] \equiv e_2$. Thus, by the definition of substitution, we can obtain our desired result:

  $$(e_1 \mathit{op} e_2)[x \mapsto e'] \equiv e_1[x \mapsto e'] \mathit{op} e_2[x \mapsto e'] \equiv e_1 \mathit{op} e_2$$

Assignment 2 (`do-times` statement)

Consider the statement

$$\texttt{do } e \texttt{ times } s \texttt{ end}$$

where s is a statement and e is an arithmetic expression. The intuitive semantics of this statement is to execute s for e times.

Task. Give rules for the natural semantics that capture the semantics of this loop construct

Note: There is more than one possible solution.

Solution. Below, we present three different solutions..

- If e is greater than zero, we can define it as the execution of s followed by the execution of $\texttt{do } e-1 \texttt{ times } s \texttt{ end}$. The formal definition is as follows:

$$\frac{}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to \sigma}\ (\textsc{Times1F}) \text{ if } \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{ff}$$

$$\frac{\langle s, \sigma\rangle \to \sigma'' \quad \langle \texttt{do } e-1 \texttt{ times } s \texttt{ end}, \sigma''\rangle \to \sigma'}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to \sigma'}\ (\textsc{Times1T}) \text{ if } \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{tt}$$

  Intuitively, this means that if $e > 0$, the arithmetic expression $e-1$ is evaluated in the state resulting from the execution of s. If s modifies some variables involved in e, then this may affect its evaluation, and the termination of the execution is not necessarily guaranteed, e.g., consider $\texttt{do x times x:=x+1 end}$.

- If the arithmetic expression e is greater than zero, we can define the execution of the construct as the execution of $\texttt{do } e-1 \texttt{ times } s \texttt{ end}$ followed by the execution of s. The formal definition is as follows:

$$\frac{}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to \sigma}\ (\textsc{Times2F}) \text{ if } \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{ff}$$

$$\frac{\langle \texttt{do } e-1 \texttt{ times } s \texttt{ end}, \sigma\rangle \to \sigma'' \quad \langle s, \sigma''\rangle \to \sigma'}{\langle \texttt{do } e \texttt{ times } s \texttt{ end}, \sigma\rangle \to \sigma'}\ (\textsc{Times2T}) \text{ if } \mathcal{B}[\![e > 0]\!]\sigma = \mathrm{tt}$$

  Intuitively, this means that the arithmetic expression $e-1$ is evaluated only in the initial state. Then the termination of the execution is guaranteed. For instance, the execution of $\texttt{do x times x:=x+1 end}$ from a state in which the value of `x` is 3 will end in a state in which the value of `x` is 6.

- The last solution we propose is to evaluate the arithmetic expression if it is not a numeral, and to iterate the loop n times (where the numeral n and the expression e evaluate to the same value) without re-evaluating the arithmetic expression each time.

$$\frac{\langle \texttt{do}\ \ n\ \ \texttt{times}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma'}{\langle \texttt{do}\ \ e\ \ \texttt{times}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma'}\ (\textsc{Times3Eval})^{(*)}$$

$$\frac{}{\langle \texttt{do}\ \ e\ \ \texttt{times}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma}\ (\textsc{Times3F})\ \text{if}\ \mathcal{A}[\![e]\!]\sigma \leq 0$$

$$\frac{\langle \texttt{do}\ \ n'\ \ \texttt{times}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma''\quad \langle s, \sigma''\rangle \to \sigma'}{\langle \texttt{do}\ \ n\ \ \texttt{times}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma'}\ (\textsc{Times3T})^{(**)}$$

where the side conditions are:

$^{(*)}$ if e is not a numeral and $\mathcal{A}[\![e]\!]\sigma > 0$ and $\mathcal{A}[\![e]\!]\sigma = \mathcal{N}[\![n]\!]$

$^{(**)}$ if $\mathcal{N}[\![n]\!] > 0$ and $\mathcal{N}[\![n']\!] = \mathcal{N}[\![n]\!] - 1$

Assignment 3 (`time` statement)

Task. Extend the natural semantics of **IMP** to support the statement $x := \texttt{time}\ \ s$, where $\texttt{time}\ \ s$ returns an integer that counts the number of assignments executed during the execution of the statement s.

Solution. We have to modify the structure of the state in order to count the number of assignments. The new state could be a state or a pair composed by a state and an integer counting the number of assignments: $\mathit{ExtendedState} = \mathit{State} \cup (\mathit{State} \times \mathbb{Z})$.

We extend the natural semantics of **IMP** with the following rules:

$$\frac{\langle s, (\sigma, 0)\rangle \to (\sigma', n)}{\langle x := \texttt{time}\ \ s, \sigma\rangle \to \sigma'[x \mapsto n]}\ (\textsc{Time})$$

$$\frac{\langle s, (\sigma, 0)\rangle \to (\sigma', n')}{\langle x := \texttt{time}\ \ s, (\sigma, n)\rangle \to (\sigma'[x \mapsto n'], n + n' + 1)}\ (\textsc{TimeExt})$$

$$\frac{\langle s, (\sigma, 0)\rangle \to (\sigma', n')}{\langle x := \texttt{time}\ \ s, (\sigma, n)\rangle \to (\sigma'[x \mapsto n'], n + 1)}\ (\textsc{TimeExt}')$$

$$\frac{}{\langle x := e, (\sigma, n)\rangle \to (\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma], n + 1)}\ (\textsc{AssExt})$$

Note that

- the rule $(\textsc{Time})$ uses n in the state update (for x) without using $\mathcal{N}$ (semantics of numerals) since n is already an integer value

- the rule (TimeExt) could return a count of assignments that is $n + n' + 1$ or $n + 1$. Both solutions are reasonable interpretations of the verbal description: in the first case we take into account the number of assignments also in nested `time` statements, while in the second case we do not consider them

We now have to define rules over the extended states which are pairs of "old" states and integer values. The extended rules simply duplicate the semantics defined on "old" states.

$$\frac{}{\langle \texttt{skip}, (\sigma, n)\rangle \to (\sigma, n)}\ (\textsc{SkipExt})$$

$$\frac{\langle s_1, (\sigma, n)\rangle \to (\sigma'', n_1) \quad \langle s_2, (\sigma'', n_1)\rangle \to (\sigma', n')}{\langle s_1;\ s_2, (\sigma, n)\rangle \to (\sigma', n')}\ (\textsc{SeqExt})$$

$$\frac{\langle s_1, (\sigma, n)\rangle \to (\sigma', n')}{\langle \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}, (\sigma, n)\rangle \to (\sigma', n')}\ (\textsc{IfTExt})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma = \mathrm{tt}$$

$$\frac{\langle s_2, (\sigma, n)\rangle \to (\sigma', n')}{\langle \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}, (\sigma, n)\rangle \to (\sigma', n')}\ (\textsc{IfFExt})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma = \mathrm{ff}$$

$$\frac{\langle s, (\sigma, n)\rangle \to (\sigma'', n_1) \quad \langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, (\sigma'', n_1)\rangle \to (\sigma', n')}{\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, (\sigma, n)\rangle \to (\sigma', n')}\ (\textsc{WhTExt})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma = \mathrm{tt}$$

$$\frac{}{\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, (\sigma, n)\rangle \to (\sigma, n)}\ (\textsc{WhFExt})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma = \mathrm{ff}$$

Assignment 4 (`repeat-until` and `while` loops)

Consider the extension of the programming language **IMP** with the statement

$$\texttt{repeat}\ s\ \texttt{until}\ b$$

where s is a statement and b is a Boolean expression. In the natural semantics, the semantics of this new statement is captured by the following two rules:

$$\frac{\langle s, \sigma\rangle \to \sigma'}{\langle \texttt{repeat}\ s\ \texttt{until}\ b, \sigma\rangle \to \sigma'}\ (\textsc{RepT})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma' = \mathrm{tt}$$

$$\frac{\langle s, \sigma\rangle \to \sigma'' \quad \langle \texttt{repeat}\ s\ \texttt{until}\ b, \sigma''\rangle \to \sigma'}{\langle \texttt{repeat}\ s\ \texttt{until}\ b, \sigma\rangle \to \sigma'}\ (\textsc{RepF})\ \text{if}\ \mathcal{B}[\![b]\!]\sigma'' = \mathrm{ff}$$

Task. Prove that, for all σ, σ', b, s, if

$$\vdash \langle \texttt{repeat}\ s\ \texttt{until}\ b, \sigma\rangle \to \sigma'$$

then

$$\vdash \langle s;\ \texttt{while not}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma\rangle \to \sigma'.$$

Solution. We define

$$P(T) \equiv \forall \sigma, \sigma', b, s \cdot \big(\mathit{root}(T) \equiv \langle \texttt{repeat}\ \ s\ \ \texttt{until}\ \ b, \sigma\rangle \to \sigma' \\ \Longrightarrow \vdash \langle s; \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma'\big)$$

and prove $\forall T.\, P(T)$ (which is equivalent to the statement to be proved) by induction on the shape of the derivation tree T. Thus, for arbitrary T, our I.H. is $\forall T' \sqsubset T.\, P(T')$, and we need to prove $P(T)$.

Let σ, σ', b, s be arbitrary and assume the left-hand side of the implication. We show the right-hand side of the implication (PO) by a case analysis of the last rule applied in T, which must be either $(\textsc{RepT})$ or $(\textsc{RepF})$:

- **Case** $(\textsc{RepT})$: Then T has the form:

$$\dfrac{\begin{array}{c}T_1\\ \langle s, \sigma\rangle \to \sigma'\end{array}}{\langle \texttt{repeat}\ \ s\ \ \texttt{until}\ \ b, \sigma\rangle \to \sigma'}\ (\textsc{RepT})$$

  for some derivation tree T_1, and we must have $\mathcal{B}[\![b]\!]\sigma' = \mathrm{tt}$, which implies $\mathcal{B}[\![\texttt{not}\ \ b]\!]\sigma' = \mathrm{ff}$.

  We can construct a suitable derivation tree to justify (PO) as follows:

$$\dfrac{\begin{array}{c}T_1\\ \langle s, \sigma\rangle \to \sigma'\end{array} \qquad \dfrac{}{\langle \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma'\rangle \to \sigma'}\ (\textsc{WhF}_{\mathit{NS}})}{\langle s; \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma\rangle \to \sigma'}\ (\textsc{Seq}_{\mathit{NS}})$$

- **Case** $(\textsc{RepF})$: In this case, the derivation tree T has the form:

$$\dfrac{\begin{array}{c}T_1\\ \langle s, \sigma\rangle \to \sigma''\end{array} \qquad \begin{array}{c}T_2\\ \langle \texttt{repeat}\ \ s\ \ \texttt{until}\ \ b, \sigma''\rangle \to \sigma'\end{array}}{\langle \texttt{repeat}\ \ s\ \ \texttt{until}\ \ b, \sigma\rangle \to \sigma'}\ (\textsc{RepF})$$

  for some state σ'' and derivation trees T_1, T_2 where $\mathcal{B}[\![b]\!]\sigma'' = \mathrm{ff}$. Thus $\mathcal{B}[\![\texttt{not}\ \ b]\!]\sigma'' = \mathrm{tt}$.

  Since T_2 is a proper subtree of the derivation tree T, we can use the I.H. to obtain $P(T_2)$. Instantiating the quantified variables appropriately, we can ensure that the left-hand side of the implication holds. Thus, we get $\vdash \langle s; \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \texttt{end}, \sigma''\rangle \to \sigma'$, i.e., there exists some derivation tree, say T_3, with $\mathit{root}(T_3) \equiv \langle s; \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma''\rangle \to \sigma'$.

  The last rule applied in T_3 must be $\textsc{Seq}_{\mathit{NS}}$, so T_3 must have the form:

$$\dfrac{\begin{array}{c}T_4\\ \langle s, \sigma''\rangle \to \sigma'''\end{array} \qquad \begin{array}{c}T_5\\ \langle \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma'''\rangle \to \sigma'\end{array}}{\langle s; \texttt{while not}\ \ b\ \ \texttt{do}\ \ s\ \ \texttt{end}, \sigma''\rangle \to \sigma'}\ (\textsc{Seq}_{\mathit{NS}})$$

for some state σ''' and derivation trees T_4, T_5.

We can now construct a derivation tree to justify (PO) as follows: