2D1454 Semantics of Programming Languages

Examination Problems with solution sketches
19 December 2006

Dilian Gurov
KTH CSC
tel: 08-790 8198

---

1. Let us extend the simple imperative programming language **IMP** with *threads* by adding the statement **thread** c **end**. The intended behaviour of this statement is that it generates a new thread executing command c. Multiple threads are executed non-deterministically: At any point of an execution, any of the threads can become active (that is, be scheduled for execution). 6p

   (a) Give an abstract machine semantics (see lecture notes) for **IMP** with threads. Configurations will now have multisets Γ of stacks of commands (one stack per thread), and transitions will have the shape $\langle \Gamma, \sigma \rangle \rightarrow_{AM} \langle \Gamma', \sigma' \rangle$. Configurations $\langle \emptyset, \sigma \rangle$ can be abbreviated as σ. If we take a formal-sum notation for multisets (where, for example, $2a+3b$ denotes the multiset $\{a, a, b, b, b\}$), we could give (for example) the following axiom for **skip**:

$$\langle (\mathbf{skip} \cdot \gamma) + \Gamma, \sigma \rangle \rightarrow_{AM} \langle \gamma + \Gamma, \sigma \rangle$$

   Solution: The rules are almost identical to the ones presented in class for **IMP** without threads, but, as in the case for **skip** given above, have an additional $+\Gamma$ component in the configurations. The only interesting rule is the (new) rule for **thread** c **end**:

$$\langle (\mathbf{thread}\ c\ \mathbf{end} \cdot \gamma) + \Gamma, \sigma \rangle \rightarrow_{AM} \langle c + \gamma + \Gamma, \sigma \rangle$$

   (b) Use your semantics to execute the program **thread** $X := 0$ **end**; $X := 1$ starting from an arbitrary initial state σ. Clearly identify the rules used in the derivation.

   Solution: The execution can be presented schematically as:

$$\begin{array}{ll} & \langle \mathbf{thread}\ X := 0\ \mathbf{end}; X := 1, \sigma \rangle \\ \rightarrow_{AM} & \langle \mathbf{thread}\ X := 0\ \mathbf{end}\ \cdot X := 1, \sigma \rangle \\ \rightarrow_{AM} & \langle (X := 0) + (X := 1), \sigma \rangle \\ & \rightarrow_{AM} \langle X := 1, \sigma[0/X] \rangle \rightarrow_{AM} \sigma[1/X] \\ & \rightarrow_{AM} \langle X := 0, \sigma[1/X] \rangle \rightarrow_{AM} \sigma[0/X] \end{array}$$

   Notice the non-deterministic branching at the (third) configuration $\langle (X := 0) + (X := 1), \sigma \rangle$!

---

2. Consider the big-step operational semantics of **IMP**. 6p

   (a) Show that $\Vdash \langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$ implies $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$. You will need to use a special kind of induction here, namely (mathematical) induction on the *depth* of the derivation trees for transitions $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$. (That is, you assume that $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$ holds whenever a transition of the shape $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$ is derivable with a derivation tree of depth n, and you prove that then this also holds for $n+1$.)

   Solution: In the base case $n = 0$ the result holds vacuously, since there are no derivations of depth 0 (that is, axioms) for transitions of the shape $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$. For the induction case, assume that for all $\sigma, \sigma' \in \Sigma$, $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$ holds whenever $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$ is derivable with a derivation tree with depth n (induction hypothesis). Assume $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$ is derivable with a derivation tree with depth $n+1$. (We show that $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$.) We consider the two possible cases for the last rule applied in the derivation of $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \rightarrow \sigma'$.

   Case 1: Last rule is $\mathbf{while}_F$. Then $\Vdash \langle b, \sigma \rangle \rightarrow \mathbf{false}$ and $\sigma' = \sigma$, and hence $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$.

   Case 2: Last rule is $\mathbf{while}_T$. Then, for some σ'', $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma'' \rangle \rightarrow \sigma'$ is derivable with a derivation of depth n. By the induction hypothesis, $\Vdash \langle b, \sigma' \rangle \rightarrow \mathbf{false}$.

(b) Consider the transformation on **IMP** programs from program **while** b **do** (**while** b **do** c) to program **while** b **do** c. Show that the transformation is a semantics-preserving *optimization* by proving that the two programs are *equivalent*.

Solution: The proof is standard, by transforming every derivation of $\langle \textbf{while } b \textbf{ do } (\textbf{while } b \textbf{ do } c), \sigma\rangle \to \sigma'$ to a derivation of $\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma'$, and *vice versa*, by using (a).

---

3. Consider the **IMP** program c: **5p**

$$\textbf{while } \neg(X = Y) \textbf{ do } (X := X + 1; Y := Y - 1)$$

Use the big-step operational semantics of **IMP** to prove that the program *terminates* for all (initial) states in $S \stackrel{def}{=} \{\sigma \in \Sigma \mid \exists k \geq 0.\, \sigma(Y) = \sigma(X) + 2k\}$.

Solution: Define $\prec \subseteq S \times S$ as follows:

$$\sigma \prec \sigma' \stackrel{def}{\Leftrightarrow} \sigma(Y) - \sigma(X) < \sigma'(Y) - \sigma'(X)$$

Since $\sigma(Y) - \sigma(X) \geq 0$ for all $\sigma \in \Sigma$, $\prec$ is well-founded. We use well-founded induction to prove $\forall \sigma \in S.\, \exists \sigma' \in \Sigma.\ \Vdash \langle c, \sigma\rangle \to \sigma'$.

Let $\sigma \in S$, and let $\exists \sigma' \in \Sigma.\ \Vdash \langle c, \sigma''\rangle \to \sigma'$ hold for all $\sigma'' \in S$ such that $\sigma'' \prec \sigma$ (induction hypothesis).

Case 1: $\sigma(X) = \sigma(Y)$. It is straightforward to produce a direct derivation of $\langle c, \sigma\rangle \to \sigma$, with last rule applied $\textbf{while}_F$, and hence $\exists \sigma' \in \Sigma.\ \Vdash \langle c, \sigma\rangle \to \sigma'$.

Case 2: $\sigma(X) \neq \sigma(Y)$. Again, we construct a derivation of $\langle c, \sigma\rangle \to \sigma'$, with last rule applied $\textbf{while}_T$. The sub-derivations of the first two premises $\langle \neg(X = Y), \sigma\rangle \to \textbf{true}$ and $\langle X := X + 1; Y := Y - 1, \sigma\rangle \to \sigma[\sigma(X)+1/X, \sigma(Y)-1/Y]$ are easy to construct. The existence of a sub-derivation for the third sub-goal $\langle c, \sigma[\sigma(X) + 1/X, \sigma(Y) - 1/Y]\rangle \to \sigma'$ is guaranteed (for some σ'!) by the induction hypothesis, since $\sigma[\sigma(X)+1/X, \sigma(Y)-1/Y] \in S$ and $\sigma[\sigma(X)+1/X, \sigma(Y)-1/Y] \prec \sigma$ whenever $\sigma \in S$. Therefore $\exists \sigma' \in \Sigma.\ \Vdash \langle c, \sigma\rangle \to \sigma'$.

---

4. Consider the following **IMP** program c for computing $sum(n) \stackrel{def}{=} \sum_{k=0}^{n} k$. **4p**

$Z := 0;$

$X := 1;$

while $X \leq Y$ **do**

  $Z := Z + X;$

  $X := X + 1$

Use the axiomatic semantics of **IMP** to verify that the program meets the specification

$$\{Y = n \wedge Y \geq 0\}\; c \;\{Z = sum(n)\}$$

(a) Present the proof (preferrably) as a proof tableau (that is, as a fully annotated program).

Solution: The annotation is standard once one has chosen a suitable loop invariant. One such choice is $X \leq Y + 1 \wedge Y = n \wedge Z = sum(X - 1)$.

(b) Identify and justify the resulting proof obligations.

Solution: The standard annotation produces 3 proof obligations, which are easily discharged. The main property needed here is $sum(m) = sum(m - 1) + m$.

---

5. Consider the axiomatic semantics of **IMP**. Show that for all commands $c \in Com$ and all assertions $A \in Assn$, **4p**

$$\Vdash \{A\}\; c \;\{\textbf{true}\}$$

Solution: We use structural induction on commands c to show $\forall c \in Com. \forall A \in Assn.\ \Vdash \{A\}\, c\, \{\mathbf{true}\}$. Here we only show the most interesting case $c \equiv \mathbf{while}\ b\ \mathbf{do}\ c'$. Assume $\forall A \in Assn.\ \Vdash \{A\}\ c'\ \{\mathbf{true}\}$ (induction hypothesis). Let $A \in Assn$. Consider the (incomplete) derivation:

$$\dfrac{\dfrac{\checkmark}{\models A \Rightarrow \mathbf{true}} \qquad \dfrac{\{\mathbf{true} \wedge b\}\ c'\ \{\mathbf{true}\}}{\{\mathbf{true}\}\ \mathbf{while}\ b\ \mathbf{do}\ c'\ \{\mathbf{true} \wedge \neg b\}} \qquad \dfrac{\checkmark}{\models \mathbf{true} \wedge \neg b \Rightarrow \mathbf{true}}}{\{A\}\ \mathbf{while}\ b\ \mathbf{do}\ c'\ \{\mathbf{true}\}}$$

The two resulting proof obligations hold trivially, while the sub-goal $\{\mathbf{true} \wedge b\}\ c'\ \{\mathbf{true}\}$ is derivable due to the induction hypothesis. Hence $\forall A \in Assn.\ \Vdash \{A\}\ \mathbf{while}\ b\ \mathbf{do}\ c'\ \{\mathbf{true}\}$.

---

6. Consider again the **IMP** program from problem 4 above. Use the fixed–point characterization of the denotational semantics of while–loops of **IMP** and the Fixed–Point Theorem to iteratively compute the denotation of the above program. That is: 5p

(a) Determine the transformer Γ for the while loop. Simplify it as much as possible.
Solution: After simplification, we obtain from the definition of $\Gamma_{b,c}$:

$$\begin{array}{rcl} \Gamma(F) & = & \{(\sigma, \sigma') \mid \sigma(X) \leq \sigma(Y) \wedge (\sigma[\sigma(Z) + \sigma(X)/Z, \sigma(X) + 1/X], \sigma') \in F\} \\ & \cup & \{(\sigma, \sigma) \mid \sigma(X) > \sigma(Y)\} \end{array}$$

(b) Use Γ to compute the first three (non–empty) approximants of the fixed–point computation.

(c) Guess the general shape of the i–th approximant.
Solution: The i–th approximant can be presented as:

$$\begin{array}{rcl} \Gamma^i(\emptyset) & = & \{(\sigma, \sigma[\sigma(Z) + sum(\sigma(X), \sigma(Y))/Z, \sigma(Y) + 1/X]) \mid 0 \leq \sigma(Y) - \sigma(X) \leq i - 2\} \\ & \cup & \{(\sigma, \sigma) \mid \sigma(X) > \sigma(Y)\} \end{array}$$

where we use $sum(m, n)$ to denote $\sum_{k=m}^{n} k$.

(d) Use this to obtain the limit value (which is the denotation of the while loop).
Solution: The limit of the fixed-point construction is:

$$\begin{array}{rcl} \bigcup_{i \in \omega} \Gamma^i(\emptyset) & = & \{(\sigma, \sigma[\sigma(Z) + sum(\sigma(X), \sigma(Y))/Z, \sigma(Y) + 1/X]) \mid \sigma(X) \leq \sigma(Y)\} \\ & \cup & \{(\sigma, \sigma) \mid \sigma(X) > \sigma(Y)\} \end{array}$$

(e) Compute the denotation of the whole program.
Solution: For the whole program, we obtain:

$$\begin{array}{rcl} \mathcal{C}[\![c]\!] & = & \{(\sigma, \sigma[sum(1, \sigma(Y))/Z, \sigma(Y) + 1/X]) \mid \sigma(Y) \geq 1\} \\ & \cup & \{(\sigma, \sigma[0/Z, 1/X]) \mid \sigma(Y) < 1\} \end{array}$$
