DD2454 Semantics of Programming Languages

Examination Problems
with solution sketches
19 December 2007

Dilian Gurov
KTH CSC
tel: 08-790 8198

---

1. Consider the transformation on **IMP** programs, from command [2p]

$$\textbf{if } b_0 \textbf{ then } (\textbf{if } b_1 \textbf{ then } c_0 \textbf{ else } c_1) \textbf{ else } c_1$$

to command

$$\textbf{if } b_0 \wedge b_1 \textbf{ then } c_0 \textbf{ else } c_1$$

Use the big-step operational semantics of **IMP** to show that the transformation is a semantics preserving optimization, by proving equivalence of the two commands.

Solution: (Sketch) Let's abbreviate the first command by c and the second by c'. We have to show $c \sim c'$, that is $\forall \sigma, \sigma'.\ (\Vdash \langle c, \sigma \rangle \to \sigma' \;\Leftrightarrow\; \Vdash \langle c', \sigma \rangle \to \sigma')$.

In the first direction, we show that for any derivation of $\langle c, \sigma \rangle \to \sigma'$ there is a derivation of $\langle c', \sigma \rangle \to \sigma'$. To this end, we consider four cases, depending on the values to which b_0 and b_1 evaluate in state σ. In each case, we show how the sub-derivations of $\langle c, \sigma \rangle \to \sigma'$ can be combined into a derivation of $\langle c', \sigma \rangle \to \sigma'$. The second dicrection is established by the same derivation schemes.

---

2. Let us extend the simple imperative programming language **IMP** with another iterative control statement, namely the command [5p]

$$\textbf{for } X \textbf{ in } m..n \textbf{ do } c$$

where m and n are numbers, with the expected behaviour: the body c of the statement is executed consecutively for all values of location X from m to n. So, at each iteration, X is assigned the corresponding value, which is incremented by one after each execution of the body. If $m > n$ the command behaves as **skip**.

(a) Consider the small-step operational semantics of **IMP** (see lecture notes and handouts). Define the meaning of the new command by providing rules for it.

Solution: Two rules suffice to capture the intended meaning of the new command:

$$\textsc{For1}\quad \frac{-}{\langle \textbf{for } X \textbf{ in } m..n \textbf{ do } c, \sigma \rangle \to_S \langle c; \textbf{for } X \textbf{ in } (m+1)..n \textbf{ do } c, \sigma[m/X] \rangle} \quad m \leq n$$

$$\textsc{For2}\quad \frac{-}{\langle \textbf{for } X \textbf{ in } m..n \textbf{ do } c, \sigma \rangle \to_S \sigma} \quad m > n$$

(b) Use your semantics to execute the program

$$Y := 0;\ \textbf{for } X \textbf{ in } 1..2 \textbf{ do } Y := Y + X$$

from an arbitrary initial state σ to a final configuration. Show all derivations.

Solution: (Sketch) There are six small-step transitions (with their corresponding derivations), the last one leading to the final configuration $\sigma[2/X, 3/Y]$.

---

3. Let Com_{WF} denote the set of **while**–free commands of **IMP**. Prove termination of execution of **while**–free programs: [4p]

$$\forall c \in Com_{WF}.\, \forall \sigma \in \Sigma.\, \exists \sigma' \in \Sigma.\ \Vdash \langle c, \sigma \rangle \to \sigma'$$

by using structural induction.

Solution: We have to consider in turn each of the four formation rules for **while**–free programs. Here we show the case for the third formation rule only, namely sequential composition.

Case $c \equiv c_0; c_1$. Since we are applying structural induction, the induction hypotheses are: $\forall \sigma \in \Sigma.\, \exists \sigma' \in \Sigma.\ \Vdash \langle c_0, \sigma \rangle \to \sigma'$ (IH1) and $\forall \sigma \in \Sigma.\, \exists \sigma' \in \Sigma.\ \Vdash \langle c_1, \sigma \rangle \to \sigma'$ (IH2). We want to show $\forall \sigma \in \Sigma.\, \exists \sigma' \in \Sigma.\ \Vdash \langle c_0; c_1, \sigma \rangle \to \sigma'$. To this end, assume $\sigma \in \Sigma$ is an arbitrary state. By (IH1), there must be $\sigma' \in \Sigma$ so that $\Vdash \langle c_0, \sigma \rangle \to \sigma'$ (1). Then, by (IH2), there must be $\sigma'' \in \Sigma$ so that $\Vdash \langle c_1, \sigma' \rangle \to \sigma''$ (2). From (1) and (2), by rule SEQ follows that we can derive $\langle c_0; c_1, \sigma \rangle \to \sigma''$. Therefore $\exists \sigma' \in \Sigma.\ \Vdash \langle c_0; c_1, \sigma \rangle \to \sigma'$.

---

4. Consider the **IMP** program **while true do** c, where c is an arbitrary command. Execution of the program does not terminate from any initial state σ. Prove this in two ways, based on: [6p]

   (a) the denotational semantics of **IMP**;

   (b) the axiomatic semantics of **IMP**.

In both cases, as a first step express the non-termination statement accordingly.

Solution: (Sketch)

(a) In the denotational semantics of **IMP**, non–termination of **while true do** c from any initial state σ is expressed as $\forall \sigma \in \Sigma.\, \neg\exists \sigma' \in \Sigma.\, (\sigma, \sigma') \in \mathcal{C}[\![\textbf{while true do } c]\!]$, which is equivalent to $\mathcal{C}[\![\textbf{while true do } c]\!] = \emptyset$; call this equality (A). Since $\mathcal{C}[\![\textbf{while true do } c]\!]$ is defined as the least fixed–point of $\Gamma_{\textbf{true},c}$, which by the Fixed–Point Theorem is equal to $\bigcup_{n \in \omega} \Gamma^n_{\textbf{true},c}(\emptyset)$, we can prove equality (A) by showing $\Gamma_{\textbf{true},c}(\emptyset) = \emptyset$, since then all approximants (and thus also their union) are equal to the empty set. Showing $\Gamma_{\textbf{true},c}(\emptyset) = \emptyset$ is easy and simply refers to the definition of $\mathcal{C}[\![\cdot]\!]$.

(b) In the axiomatic semantics of **IMP**, non–termination of **while true do** c from any initial state σ is expressed by the Hoare triple $\{true\}$ **while true do** $c\,\{false\}$. In other words, we need to show $\forall c \in Com.\ \Vdash \{true\}$ **while true do** $c\,\{false\}$.

In class, we already showed that $\forall c \in Com.\, \forall A \in Assn.\ \Vdash \{A\}\, c\, \{true\}$. Therefore, by taking A to be $true \wedge true$, for any command c there is a derivation of the Hoare triple $\{true \wedge true\}\, c\, \{true\}$. Such a derivation is easily extended to a derivation of $\{true\}$ **while true do** $c\,\{false\}$ by applying the **while**–rule followed by the consequence rule.

---

5. Consider the following program in the light of the denotational semantics of **IMP**: [4p]

while $\neg(X \leq 0)$ **do**

  if $Y \leq X$ **then**

    $X := X - Y$

  else

    $X := X - 1$

(a) Determine the transformer Γ for the **while**–loop. Simplify it as much as possible.

Solution: After simplification, we obtain:

$$\begin{aligned}
\Gamma(F) &= \{(\sigma,\sigma') \mid \sigma(X) > 0 \wedge \sigma(Y) \leq \sigma(X) \wedge (\sigma[\sigma(X) - \sigma(Y)/X], \sigma') \in F\} \\
&\cup \{(\sigma,\sigma') \mid \sigma(X) > 0 \wedge \sigma(Y) > \sigma(X) \wedge (\sigma[\sigma(X) - 1/X], \sigma') \in F\} \\
&\cup \{(\sigma,\sigma) \mid \sigma(X) \leq 0\}
\end{aligned}$$

(b) Use Γ to compute the first two non–empty approximants of the fixed–point computation. Simplify these as much as possible.

Solution: After simplification, we obtain:

$$\begin{aligned}
\Gamma^1(\emptyset) &= \{(\sigma,\sigma) \mid \sigma(X) \leq 0\} \\
\Gamma^2(\emptyset) &= \{(\sigma,\sigma[0/X]) \mid \sigma(X) > 0 \wedge \sigma(Y) = \sigma(X)\} \\
&\cup \{(\sigma,\sigma[0/X]) \mid \sigma(X) = 1 \wedge \sigma(Y) > \sigma(X)\} \\
&\cup \{(\sigma,\sigma) \mid \sigma(X) \leq 0\}
\end{aligned}$$

(c) Argue for correctness of your answers based on the intuitive understanding of what fixed-point approximants correspond to in terms of execution of a **while**–loop.

Solution: As explained in class, the i–th approximant of Γ contains exactly the state pairs (σ, σ') for which the while loop, when executed from σ, terminates in σ' by executing the body of the loop at most $i - 1$ times.

The above sets $\Gamma^1(\emptyset)$ and $\Gamma^2(\emptyset)$ indeed capture this for $i = 1$ and $i = 2$: the loop terminates without executing the body, in the start state σ, exactly when $\sigma(X) \leq 0$, and terminates by executing the body just once, in state $\sigma[0/X]$, whenever $\sigma(X) > 0 \wedge \sigma(Y) = \sigma(X)$ (that is, when the then–branch is taken) or $\sigma(X) = 1 \wedge \sigma(Y) > \sigma(X)$ (that is, when the else–branch is taken).

---

6. Consider the **IMP** program MED for computing the average value of two integers: 5p

if $X \leq Y$ **then**
  while $\neg(X = Y)$ **do**
    $X := X + 1;$
    $Y := Y - 1$
else
  while $\neg(X = Y)$ **do**
    $X := X - 1;$
    $Y := Y + 1$

Notice that the program does not terminate from all initial states.

(a) Verify that the program meets the specification

$$\{X = m \wedge Y = n\}\ \textsc{Med} \left\{X = \frac{m+n}{2}\right\}$$

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

Solution: The annotations are easily obtained after choosing suitable loop invariants. For both loops, $X + Y = m + n$ is a suitable choice.

(b) Identify and justify the resulting proof obligations.

(c) Improve the specification by strengthening the pre-condition to describe the set of all states from which MED terminates.

Solution: The program terminates exactly for all initial states in which the values of X and Y differ by an even number. This could be formalized for example as follows:

$$\{X = m \wedge Y = n \wedge \exists k \in \omega.\, m = n + 2k\}\ \textsc{Med} \left\{X = \frac{m+n}{2}\right\}$$

---

7. Consider the axiomatic semantics of **IMP**. Recall that validity of Hoare triples $\{A\}\, c\, \{B\}$ is defined as: [4p]

$$\models \{A\}\, c\, \{B\} \overset{def}{\Leftrightarrow} \forall \sigma, \sigma' \in \Sigma.\, (\sigma \models A \;\wedge\; (\sigma, \sigma') \in \mathcal{C}[\![c]\!] \;\Rightarrow\; \sigma' \models B)$$

where for simplicity we assume that no meta-variables are used (and hence no interpretations I are needed). Now, prove that $\models \{A\}\, \textbf{while}\ b\ \textbf{do}\ c\, \{B\}$ implies $\models A \Rightarrow B \vee b$.

Solution: Proof by contradiction. Assume $\models \{A\}\, \textbf{while}\ b\ \textbf{do}\ c\, \{B\}$ (1), and assume (for the sake of arriving at a contradiction) that $\not\models A \Rightarrow B \vee b$. Then, there must be a state σ such that $\sigma \models A$ (2) but $\sigma \not\models B$ (3) and $\sigma \not\models b$ (4). From (4), since $\sigma \models b$ if and only if $\mathcal{B}[\![b]\!]\,(\sigma) = \mathit{true}$, we obtain that $\mathcal{B}[\![b]\!]\,(\sigma) = \mathit{false}$. By the definition of the denotational semantics of while loops, we then have $(\sigma, \sigma) \in \mathcal{C}[\![\textbf{while}\ b\ \textbf{do}\ c]\!]$ (5). Then, by the definition of (1), assumption (2) and from (5), it follows that $\sigma \models B$. But this contradicts assumption (3).