DD2457 Program Semantics and Analysis

Examination Problems
3 June 2015, 14:00 - 19:00

Dilian Gurov
KTH CSC
tel: 08-790 8198

Give solutions in English or Swedish, each problem beginning on a new sheet. Write your name on all sheets. The maximal number of points is given for each problem. The total number of points is 30. Up to two bonus points per section will be taken into account. The course book, the handouts, own notes taken in class, as well as reference material are admissible at the exam.

1 Level E

For passing level E you need 6 (out of 8) points from this section.

1. Consider the *Natural semantics* of the **While** language. The rules of the semantics do not impose a particular order of execution. However, we discussed in class that the language is deterministic, and that a strategy can be imposed on applying the rules that guarantees that for any given initial configuration $\langle S, s\rangle$ the exploration will terminate whenever there exists a derivation. 4p

   (a) Suggest a procedure or function that executes **While** programs in Natural semantics from a specified initial state, yielding the final state if there is one. Provide the pseudo-code of the procedure in a suitable notation.

   (b) Explain briefly the workings of your procedure/function, and argue why it yields the correct final state whenever it exists.

2. Consider the following (not particularly useful) **While** program: 4p

$$
\begin{array}{l}
x := 17; \\
\textbf{while } 0 \leq y \textbf{ do} \\
\quad \textbf{if } x \leq y \textbf{ then} \\
\quad\quad y := y - x; \\
\quad\quad \textbf{while } 0 \leq x \textbf{ do } x := x - y \\
\quad \textbf{else} \\
\quad\quad x := x - 1;
\end{array}
$$

   Your task is to construct for this program a *test suite*, i.e., a set of tests, each being represented simply by an initial state. The *coverage criterion* for the suite is to traverse every possible simple path of the program, i.e., no edge of the control flow graph needs to be visited more than once along the same path. Apply *symbolic execution* (see handouts) to construct such a test suite. Illustrate and explain your construction step-by-step.

Please Turn Over

2 Level C

For grade D you need to have passed level E and obtained 5 (out of 12) points from this section. For passing level C you need 8 points from this section.

Consider the extension of **While** with the **thread** S **end** statement discussed in class and in Assignment 2. Extend the *Abstract Machine* of Chapter 4 of the course book to handle this extension in a way that you still can argue for correctness of the implementation of **While**. That is:

1. Propose an operational semantics for the extended language that is suitable as a *specification* for the implementation. [3p]
2. In this operational semantics, show an execution of the program: [2p]

   $$x := 0;\ \textbf{while true do}\ (x := x + 1;\ \textbf{thread}\ x := x - 1\ \textbf{end})$$

   from an arbitrary state s up to a repeating configuration.
3. Extend suitably the abstract machine language **AM** and the translation of **While** into **AM**. [2p]
4. Describe the necessary extensions of the (operational semantics of the) abstract machine, so that you can still argue for correctness of the implementation (this you will do in part A below). [3p]
5. Translate the program from item 2 above to **AM** and execute the resulting code from an arbitrary state s up to a repeating configuration, matching the execution from item 2. [2p]

3 Level A

For grade B you need to have passed level C and obtained 4 (out of 10) points from this section. For grade A you need 7 points from this section.

1. For the extension of **While** with multi-threading and its implementation developed in part C above, formalize the statement of *correctness of the implementation* (just the statement itself, not its proof). Argue as formally as you can that your correctness statement holds. [5p]
2. When we developed the *Denotational semantics* of **While**, we discussed a possible definition for **while** loops as follows: [5p]

   $$\begin{aligned}\mathcal{S}'_{ds}[\![\textbf{while}\ b\ \textbf{do}\ S]\!]\,(s) = s' \quad \overset{\mathsf{def}}{\Longleftrightarrow} \quad & \exists s_0, \ldots, s_n \in \textbf{State}. \\ & s = s_0 \wedge s' = s_n \\ & \wedge\ \forall i < n.\ (\mathcal{B}[\![b]\!]\,(s_i) = \textbf{tt} \wedge \mathcal{S}_{ds}[\![S]\!]\,(s_i) = s_{i+1}) \\ & \wedge\ \mathcal{B}[\![b]\!]\,(s_n) = \textbf{ff}\end{aligned}$$

   Instead, we chose a definition based on fixed points (i.e., as a specific solution to a recursive semantic equation). Still, argue as formally as you can that the above definition is equivalent to the chosen definition. (Hint: Refer to the meaning of the i-th approximant of the semantic transformer corresponding to the loop.)

Good luck!