Exam Automated Reasoning

Thursday, 3 February 2005, 9 - 12 h.

NB. The exam will be corrected and graded by humans, not by a computer. Therefore, you need not to bother too much about the syntactical peculiarities of PVS and Promela.

1. Let R be a binary symmetrical relation over some set X, so $R \subseteq X^2$ and $\forall x, y \in X((x,y) \in R \rightarrow (y,x) \in R)$. Then the following property holds:

   $(R^{-1})^* \subseteq R^*$

   Here $^{-1}$ denotes the inverse, so $R^{-1} = \{(y,x) \mid (x,y) \in R\}$, and * denotes transitive closure:

   $R^* = \{(f(0), f(n)) \mid n \in N \wedge f : N \rightarrow X \wedge \forall i(i \leq n \rightarrow (f(i), f(i+1)) \in R)\}$,

   Formulate a PVS-theory and a lemma that expresses this property. (You may use predicates to model relations.) Then sketch how this lemma can be proved with PVS.

2. Consider three processes, defined by

```
process Thread (self := 0 to 2)
   var priv: int := 0
   do :: true ->
      RW
      priv ++
      synch.self
   od
```

   The idea is that the processes do relevant work in `RW`, but need to be synchronized in such a way that

   $$\begin{aligned} &priv.0 \leq priv.1 + 1\ , \\ &priv.0 + priv.1 \leq priv.2 + 2\ , \\ &priv.2 \leq priv.0 + priv.1 + 1\ . \end{aligned}$$

   always hold.

   (a) Implement the three commands `sych.0`, `sych.1`, `sych.2` to realize this, under the following conditions:

      i. unnecessary waiting is to be avoided;
      ii. the commands are not allowed to modify *priv*;
      iii. they may only contain assignments, atomic waits and busy-waiting loops;
      iv. you may only use atomic commands that are *simply shared*, i.e. they may refer to at most one shared variable at most once.

   (b) Show that your solution admits an execution where the value of *priv.2* becomes arbitrary large.

   (c) Indicate how you may check the correctness of your solution with Spin. Pay attention to correctness, deadlock and progress. Do not forget to reduce your solution to a finite state space.

   Hint: introduce auxiliary variables, e.g. shared variables *sh* and a private variable *own*.