Deductive Verification of Reactive Systems
Fall 2007: Assignment No. 3

Due Date: 12.21.07
December 10, 2007

The solution to this assignment should be submitted as attachment to an e-mail message. The textual part should be prepared as a postscript, PDF, or Word document. All submitted SMV and PF files should be submitted as separate files. You may group all relevant files into a single file, using ZIP or TAR.

1 Peterson(N) Algorithm

In Fig. 1 we present Algorithm Peterson(N), which implements mutual exclusion among N processes, using shared variables but no semaphores.

\[
\begin{array}{l}
\text{in} \quad N : \text{integer where } N > 1 \\
\text{local} \quad y : \text{array } [1..N] \text{ of } 0..N \text{ where } \forall k : y[k] = 0 \\
\quad s : \text{array } [1..N] \text{ of } 1..N \\
\hline
\text{loop forever do} \\
\quad [\ell_0 : \text{NonCritical} \\
\quad \ell_1 : \text{while } y[i] < N \text{ do} \\
\quad \quad [\ell_2 : (y[i], s[y[i]] + 1] := (y[i] + 1, i) \\
\quad \quad \ell_3 : \text{await } s[y[i]] \neq i \lor \forall k \neq i : y[k] < y[i] \\
\quad \ell_4 : \text{Critical} \\
\quad \ell_5 : y[i] := 0]
\end{array}
\]

Figure 1: Program Peterson(N)

In file petn.smv available on the course web page, we present an SMV program that represents a finite-state version of program Peterson(N). It represents a finite-state instance obtained by taking N = 5.

In this program, each process \( P[i] \) goes through \( N \) competition levels before entering its critical section. The variable \( y[i] \) represents the level of the competition in which \( P[i] \) is currently engaged. In statement \( \ell_2 \), \( P[i] \) increments its competition level to \( L \), while saving in \( s[L] \) its identity \( i \). The signature \( s[L] \) helps to break ties in case two or more processes...
enter competition level \( L \) at about the same time. This is because \( s[L] \) records the identity of the most recent process that entered competition level \( L \). Statement \( \ell_3 \) allows a process to proceed to the level beyond \( L \) if either \( y[i] \) is greater than all \( y[j], j \neq i, \) or \( s[L] \neq i \) which provides evidence that \( P[i] \) was not the last to enter level \( L \). In the worst case, when all processes attempt to access their critical sections at about the same time, there will be one process left at every level \( L \) — the process \( P[i] \), such that \( s[L] = i \). Since there are \( N \) competition levels, at most one process can reach level \( N \) and be admitted to its critical section.

**Task 1:** Model check the invariance of the following safety assertion:

\[ \varphi : \neg(at_{-\ell_4}[1] \land at_{-\ell_4}[2]) \]

This assertion states mutual exclusion between processes \( P[1] \) and \( P[2] \). If it takes too long, you can reduce the number of processes in the checked program.

**Task 2:** Model check the following response property:

\[ \psi : at_{-\ell_1}[1] \implies \Box at_{-\ell_4}[1] \]

This formula states the property of accessibility for process \( P[1] \). If it takes too long, you may reduce the number of processes.

**Task 3:** Present a deductive proof of the invariance of the safety property \( \varphi \). In your proof you may use an auxiliary inductive assertion that includes the following elements:

1. Assertion that relates the location of process \( P[i] \) to the value of \( y[i] \). For example, it may state that while \( P[i] \) is at locations \( \ell_0, \ell_2, \ell_3, \{\ell_4, \ell_5\} \), then \( y[i] \) must satisfy the constraints \( y[i] = 0, y[i] < N, y[i] > 0, \) and \( y[i] = N \), respectively.

2. Statement \( \ell_3 \) contains the test

\[ cond[i] : s[y[i]] \neq i \lor \forall k \neq i : y[k] < y[i] \]

An additional invariant that may be included in the inductive assertion states that once this test is passed successfully, it remains true until the value of \( y[i] \) is modified. Technically, this can be stated by requiring that if \( P[i] \) is at any of the locations \( \{\ell_1, \ell_2, \ell_4, \ell_5\} \) and \( y[i] > 0 \), then \( cond[i] \) should hold.

3. We say that process \( P[i] \) is engaged in an *active competition* if \( y[i] = L > 0 \) and \( s[L] \neq i \). This implies that \( P[i] \) is not alone at level \( L \). A crucial invariant states that if \( P[i] \) is engaged in an *active competition* at level \( L > 0 \), then all levels \( m \leq L \) are occupied. That is, \( y[s[m]] = m \) for every \( m \leq L \).
Task 4: Present a deductive proof of the response property $\psi$, using rule “wellx_lex” or rule “distr”.

The major component in the progress of process $P[1]$ from $\ell_1$ to $\ell_4$ consists of steps taken by process $P[1]$. It follows that transitions $\ell_1[1], \ell_2[1], \ell_3[1]$ are helpful whenever they are enabled. To monitor the progress of $P[1]$, we can use two consecutive entries in a lexicographic tuple, where the first can be $N - y[1]$, measuring the distance of $y[1]$ from $N$. The second entry could be a small integer constant, depending on which of the three $P[1]$-transitions is currently helpful.

The situation becomes more complicated when process $P[1]$ is at location $\ell_3$ but the transition is not enabled because $cond[1]$ does not hold. This implies that $y[1] = L > 0$, $s[L] = 1$, and there are one or more processes at a level not smaller than $L$.

When this is the case we should trace the evolution until all processes at a level not smaller than $L$ visit their critical section and drop out of the competition. The helpful transitions for this phase are any enabled transition of process $P[j]$, for some $j \neq 1$, such that $y[j]$ is (weakly) maximal among the $y$-values. That is, there does not exist a process $P[k]$, such that $y[k] > y[j]$. Note that the helpfulness condition for a transition $\tau_{(j, \ell)}$ should include, in addition to the enableness of $\tau_{(j, \ell)}$ and the maximality of $y[j]$, also the requirement $at.\, \ell_3[1] \land \neg cond[1]$.

According to this analysis, we will have two types of ranking functions, both ranging over 5-tuples of natural numbers. Corresponding to each of the three transitions of process $P[1]$, the ranking function will be of the form

$$\Delta_{(1, \ell)} : (N - y[1], c_{\ell}, 0, 0, 0)$$

For all other processes $P[j], j \neq 1$, we will have ranking functions of the form

$$\Delta_{(j, \ell)} : (N - y[1], C, \sum_k (y[k] \geq y[1]), N - y[j], d_{(j, \ell)})$$

where $C$ is a fixed constant, $\sum_k (y[k] \geq y[1])$ is the number of processes $y[k]$ whose $y$-value is not smaller than $y[1]$. The two last components $(N - y[j], d_{(j, \ell)})$ measure the progress of $P[j]$ until it drops out of the competition and returns to $\ell_0$. 

2