Final Exam, Reactive Systems 2003

Saturday, March 15, 10.00 - 15.00. Polacksbacken.

For assistance, you may bring and consult any notes, course material, books, etc. that you like. In Swedish:

Alla hjälpmedel tillåtna

This exam consists of 5 problems. The problems are formulated in English. Start solution of each problem on a new sheet of paper. Write your name on each sheet. You may write in English or in Swedish, whichever you prefer.

Before starting to work on any problem, read first through the entire exam.

Problem 1. (8 p) Let p be a boolean variable (of some transition system). The temporal property

> p is true at all even-numbered states and can have any value in odd-numbered states, i.e., in a sequence $s_0\ s_1\ s_2\ s_3\ s_4\ \ldots$ of states, p is true in states $s_0\ s_2\ s_4\ \ldots$ and has arbitrary values in the other states.

cannot be expressed in temporal logic. The problem is the following

a) Construct a Büchi automaton which accepts exactly all behaviors that satisfy the above property.

b) Construct a Büchi automaton which accepts exactly all behaviors that do **not** satisfy the above property.

c) Construct a Büchi automaton which accepts exactly all behaviors that have no suffix which satisfies the property. In other words, for any suffix $s_n\ s_{n+1}\ s_{n+2}\ s_{n+3}\ s_{n+4}\ \ldots$ it should **not** be the case that p is true in all of the states $s_n\ s_{n+2}\ s_{n+4}\ \ldots$.

Problem 2. (14 p) Consider the following action system.

$$
\begin{array}{l}
\textbf{Action System}\ \ \textit{Program 1} \\
\quad \textbf{variables} \\
\qquad x\ , y\ :\ \text{integer} \\
\quad \textbf{initially} \\
\qquad x = y = 0 \\
\quad \textbf{actions} \\
\qquad x \quad := \quad x + 2 \\
\quad [\!] \\
\qquad x > y - 4 \ \longrightarrow\ y \quad := \quad y + 2 \\
\quad [\!] \\
\qquad x \geq y \ \longrightarrow\ x, y \quad := \quad x - 1, y + 1 \\
\quad [\!] \\
\qquad x \leq y \ \longrightarrow\ x, y \quad := \quad x + 1, y - 1 \\
\textbf{end}
\end{array}
$$

Prove, using the techniques of *inductive invariants*, that

a) $x \neq y + 1$ is invariant.

b) $y \leq x + 10$ is invariant.

Let us introduce fairness sets into the action system. In which of the following cases is it guaranteed that the value of y will grow without bound? (i.e., for any integer K, the value of y will at some point become at least K)

c) There are four fairness sets: one fairness set for each action.

d) There is one fairness set containing all actions

e) There are two fairness sets: one containing the two first actions, and one containing the two last actions.

Problem 3. (8 p) Each of the following properties in temporal logic can be expressed by a **simpler** formula, which expresses exactly the same property. For each of the properties, find such a formula, which is as simple as possible. (here "simple" means "small" or "having few symbols")

a) $\Box \Diamond (p \,\mathcal{U}\, q)$

b) $\Box (p \,\mathcal{U}\, (\Diamond q))$

c) $\Box ((p \,\mathcal{U}\, q) \,\mathcal{U}\, p)$

Problem 4. (10 p) This problem is loosely based on the Promela model of the Alternating bit protocol in the SPIN documentation, which is included on the next page. The protocol is intended to provide reliable error-free transmission of messages from process A to process B, in spite of the possibility that the medium may lose or duplicate messages.

In the model of the next page, the losses and duplications of channels are not modeled. Your problem is to explain concretely how to modify the Promela model so that it is possible for each message to be either lost, or duplicated an arbitrary number of times in the channel. Supply **two** different ways to do this, and give a **short** comparison of the relative advantages of them. In the model, the channel capacity is only 2, meaning that there is not so much room for duplications; ignore this problem (i.e., your solution should make sense if the capacity would have been, say, 17). You do not have to write a complete new model: for instance, if your suggestion is to change each send or each receive statement, it is enough to show how one such statement is changed.

```
#define MAX 4                /* file ex.2 */
proctype A(chan in, out)
{   byte mt; /* message data */
    bit  vr;
S1: mt = (mt+1)%MAX;
    out!mt,1;
    goto S2;
S2: in?vr;
    if
    :: (vr == 1) -> goto S1
    :: (vr == 0) -> goto S3
    :: printf("MSC: AERROR1\n") -> goto S5
    fi;
S3: out!mt,1;
    goto S2;
S4: in?vr;
    if
    :: goto S1
    :: printf("MSC: AERROR2\n"); goto S5
    fi;
S5: out!mt,0;
    goto S4
}
proctype B(chan in, out)
{   byte mr, lmr;
    bit ar;
    goto S2; /* initial state */
S1: assert(mr == (lmr+1)%MAX);
    lmr = mr;
    out!1;
    goto S2;
S2: in?mr,ar;
    if
    :: (ar == 1) -> goto S1
    :: (ar == 0) -> goto S3
    :: printf("MSC: ERROR1\n"); goto S5
    fi;
S3: out!1;
    goto S2;
S4: in?mr,ar;
    if
    :: goto S1
    :: printf("MSC: ERROR2\n"); goto S5
    fi;
S5: out!0;
    goto S4
}
init {
    chan a2b = [2] of { bit };
    chan b2a = [2] of { byte, bit };
    atomic {
            run A(a2b, b2a);
            run B(b2a, a2b)
    }
}
```