ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

D. Basin and P. Müller

Formal Methods and Functional Programming

Exercise Sheet 13: Modelling and LTL

Submission deadline: -

Assignment 1

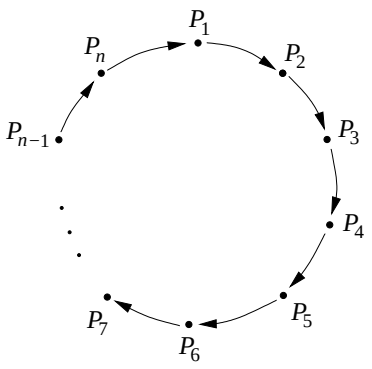
The problem of the dining philosophers illustrates common issues for concurrent programs. N philosophers sit around a circular table. Each philosopher spends her/his life thinking and eating. In the center of the table is a large platter of spaghetti. Because the spaghetti are long and tangled a philosopher must use two forks to eat them. Since the philosophers can only afford N forks, a single fork is placed between each pair of philosophers, which they have to share. Each philosopher can only reach the forks to her/his immediate left and right with her/his left and right hand, respectively.

(a) You find a skeleton (`philosopher_skeleton.pr`) of a Promela model of the dining philosophers at the course webpage. Complete the model such that each philosopher chooses nondeterminstically to pick up her/his left or right fork first. Use Spin to find a deadlock and explain its source. How does the falsification time change when increasing the number N of philosophers.

(b) The deadlock can be avoided when each philosopher behaves more deterministically to pick up the forks (not all philosophers have to behave the same). Model your solution in Promela and use Spin to check that it is indeed deadlock free. Is your model also starvation free (i.e., each philosopher will eventually eat)?

Assignment 2

Consider the following leader election protocol. For $n \geq 1$, the processes $P_1, \ldots, P_n$ are located in a ring topology, where each process is connected by an unidirectional channel to its neighbor as outlined in the figure to the right.



To distinguish the processes, each process has a unique identifier id with $1 \leq id \leq n$. The aim is to elect the process with the highest identifier as the leader within the ring. Therefore, each process executes the following algorithm:

send message id
loop
    receive message m
    if $m = id$ then stop
    if $m > id$ then send message m
end loop

(a) Model this leader election protocol for n processes in Promela.
Hint: Use an array of n channels of length 1, i.e.,

```
#define N 5  /* number of processes in the ring */
#define L 1  /* length of a channel */
chan c[N] = [L] of { byte };
```

Model a process in Promela as

```
proctype pnode(chan in, out; byte id) {
  /* algorithm for electing the leader ... */
}
```

(b) Assume that the channels are of length $n + 1$ instead of length 1 in your Promela model. Is there a state in some execution in which a channel stores more than n messages? Use Spin to verify your claim for some fixed values of n. What happens if the channels have length 0?

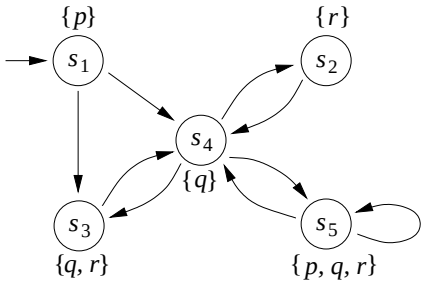
Assignment 3

Consider the following game between a mole (*Maulwurf*) and a hunter. The mole has five holes as in the above figure. At the beginning of the game, the mole hides in one of these holes. Now the game proceeds in rounds of the following form: the hunter checks one hole to see whether the mole is inside. If it is, the hunter wins the game. If not, the mole must move one hole to the left or right (if it is in the leftmost hole already, it must move to the right and conversely for the rightmost hole). After the mole has moved, the next round starts. It turns out that the hunter has a strategy to win the game, no matter how the mole moves. One sure winning strategy for the hunter is to check holes in the sequence 2-3-4-2-3-4.

- Your task is to model this game in Promela. Your model has to contain the data structure for the holes, the set-up of the initial state (the hiding of the mole), the behavior of the mole, the implementation of the hunter's winning strategy, and an assertion that ensures that after executing the strategy, the hunter has definitely caught the mole.

Assignment 4

Consider the transition system T over the set of atomic propositions $P = \{p, q, r\}$:



That is, T is the transition system $(\Gamma, s_1, \rightarrow, L)$ with $\Gamma = \{s_1, s_2, s_3, s_4, s_5\}$,

$$\rightarrow = \{(s_1, s_3), (s_1, s_4), (s_2, s_4), (s_3, s_4), (s_4, s_2), (s_4, s_3), (s_4, s_5), (s_5, s_4), (s_5, s_5)\},$$

and $L(s_1) = \{p\}$, $L(s_2) = \{r\}$, $L(s_3) = \{q\}$, $L(s_4) = \{q, r\}$ and $L(s_5) = \{p, q, r\}$.

- Which of the following LTL formulas are satisfied in T, i.e., $T \models \varphi_i$? Justify your answer. If $T \not\models \varphi_i$, provide a computation γ of T such that for the corresponding trace t, $t \not\models \varphi_i$.

$$\begin{array}{lcl}
\varphi_1 & = & \Diamond \Box\, r \\
\varphi_2 & = & \Box \Diamond\, r \\
\varphi_3 & = & \bigcirc \neg r \rightarrow \bigcirc \bigcirc\, r \\
\varphi_4 & = & \Box\, p \\
\varphi_5 & = & p \,\mathsf{U}\, \Box\, (q \vee r) \\
\varphi_6 & = & (\bigcirc \bigcirc\, q) \,\mathsf{U}\, (q \vee r) \\
\varphi_7 & = & \Diamond \Box\ \bigcirc\, q \\
\varphi_8 & = & (\Diamond \Box p) \rightarrow (\Diamond \Box r)
\end{array}$$

- Formalize the following properties in LTL:
  - Eventually, it will not be possible for the system to go to state s_1.
  - Whenever the system is in a state that satisfies r, then the next state satisfies q.
  - p always implies r except perhaps in the initial state.
  - Whenever the system is in state s_5, it will remain there until r becomes false.
  - q will be true at least twice.
  - q will be true infinitely often.
  - If p is true only at the initial state of a trace, then r is false infinitely many times in that trace.
  - s_4 can never be repeated (there is no transition from s_4 to itself).

Assignment 5 - Headache of the week

On a chessboard (an 8x8 grid), a *knight's tour* is a sequence of consecutive legal moves which can be made by a knight, which visits each square of the board exactly once. A *closed knight's tour* is one in which, at the end of the sequence, the knight is left in a position where it could move back to its original starting point (and therefore start the tour over again). Any other tour is called an *open knight's tour*. Many closed knights tours exist for a standard 8x8 board. Closed (and open) knights tours also exist using smaller grids.

Write a Promela program to model a search for knights tours (you should be able to choose whether these have to be open or closed tours) on an M x N grid (where M and N can be defined as integer constants).

Use your program to demonstrate:

1. No knight's tour (closed or open) exists for a 4x4 board (can you see why?).
2. An open knight's tour is possible on a 5x5 board, but a closed knights tour does not exist.
3. An open knight's tour is possible on a 5x6 board. If you can, try to find a closed one too.. (it does exist but you may run out of memory! One also exists for a 6x6...).
4. ...any other examples you feel like trying out (unfortunately the state space of an 8x8 search will probably require too much memory, unless you've done better than we have).

As a hint - try to avoid using extra variables for e.g., printing purposes. The more variables you have, the greater the state space of your model. Try using small types for variables too (bit, byte, etc.) where you can.