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

Submission deadline: June 7th, 2010

Please submit your solution before **9:15am** on the submission date specified above. Solutions can be submitted via e-mail or by using the boxes to the left of **RZ F1**. Make sure that the first page always contains your name, the exercise sheet number as well as your tutor's name. Don't forget to staple your pages if you submit more than one page. *Your marked solutions can be collected from the same boxes from Tuesday 8th June onwards.*

Assignment 1

In the lecture, you have seen parts of the modeling language *Promela*. In this and later assignments you will use Promela and the model checker *Spin* for modeling and analyzing concurrent programs. You can find detailed information about how to install the model checker Spin on your computer at the webpage `http://spinroot.com/spin/Man/README.html`.

Some useful information for running Spin:

- If Spin is invoked without any options, it performs a *random simulation*. For example,
  ```
  $ spin foo.pr
  ```
  performs a random simulation for the Promela model specified in the file `foo.pr`.
- The command line option `-a` generates a protocol *specific analyzer*. The output is written into a C file pan.c, which you can, e.g., compile with the GNU C compiler. Running the compiled file will explore the state space of the Promela model and check whether the model might deadlock. The assertions in the model are also checked.
  ```
  $ spin -a foo.pr
  $ gcc -o pan pan.c
  $ ./pan
  ```

Note that there is also a graphical interface for Spin, called XSpin, which you can also download from the Spin webpage `http://spinroot.com`. We recommend that you also install XSpin on your computer. See `http://spinroot.com/spin/Man/Manual.html#S` for more information.

The purpose of this assignment is to get familiar with Promela and the model checker Spin.

(a) Consider again the following statement (which you already saw on previous exercise sheets).

```
y := 0;
while x > 0 do
  y := y + x;
  x := x - 2
end
```

Write a model in Promela to check if the statement starts in a state σ with $\sigma(\mathtt{x}) = 3$, it will reach a state σ' with $\sigma'(\mathtt{y}) = 4$.

(b) Write a model in Promela to verify that the following program will result in a state in which x has either the value 1 or 4.

$$x := 1 \,[]\, x := 2; x := x + 2$$

(c) Write a model in Promela to verify that the following program will result in a state in which x has one of the values 1 or 3 or 4.

$$x := 1 \texttt{ par } x := 2; x := x + 2$$

(d) Consider the following program.

```
x := 5;
y := 1;
(while x > 1 and y < 5 do
   (x := x - y [] y := y + 1)
 end
par
 while x > 0 do
   y = y + 1;
   x = x - 1
 end)
```

Assume that we start the program in some state. Can we reach a final state in which the variable x stores the integer -7? What is the minimal value of the variable x after executing the program?

Assignment 2

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.

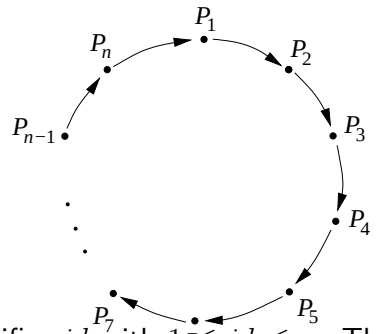
(b) The deadlock can be avoided when each philosopher behaves more deterministically to pick up the forks (not all philosophers have to behave the same). Model your solution in Promela and use Spin to check that it is indeed deadlock free. Is your model also starvation free (i.e., each philosopher will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

Assignment 3

In the lecture, you have seen a Promela program that models parts of the Needham-Schroeder protocol, namely agent Alice and an intruder. Complete the Promela program for agent Bob according to the description of the protocol. Use the template from the course webpage. Simulate your model in (X)Spin. Can you find an attack?

Assignment 4

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 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.