Technische Universität München **Summer Semester 2015**
I7
J. Esparza / A. Durand-Gasselin / D. Suwimonteerabuth **30.04.2015**

Model Checking – Exercise sheet 3

Exercise 3.1

Consider the following C function:

```
1 void syra(char n) {
2   while (n > 1) {
3     if (n % 2)
4         n = 3*n +1;
5     else
6         n = n/2;
7   }
8 }
```

1. Write down a Promela process for the above function. Use Spin to print out values of `n`, starting from e.g. `n` = 19.
2. How can you use Spin to check termination of the function?
3. Write down another Promela model containing a global variable `n` and two processes: one for handling even values and one for handling odd values. Use Spin to simulate your result.

Exercise 3.2

The sieve of Eratosthenes is an algorithm for finding all prime numbers up to a given integer n. The method works as follows:

- Create an array with the range $2..n$.
- Initially, let p equal to 2, the first prime number.
- Mark all multiples of p in the array, i.e. mark all $2p, 3p, 4p, \ldots$ up until n.
- Find in the array the next smallest unmarked number. If no such number can be found, stop. Otherwise, let p be this new number and repeat the previous step.

When the algorithm terminates, all unmarked numbers in the array are prime.

1. Write down a C function for the above algorithm.

2. Translate your C function into a Promela model. Use Spin to simulate your result.

3. How would you modify your model to support parallel executions?

Exercise 3.3

Consider the following program `mutex.c`:

```
1  #include <pthread.h>
2  #include <assert.h>
3
4  pthread_t x, y, z;
5  int cnt;
6
7  void lock(pthread_t Pid) {
8  busywait:
9    x = Pid;
10   if (y != 0
11   && !pthread_equal(Pid, y))
12     goto busywait;
13
14   z = Pid;
15   if (!pthread_equal(x, Pid))
16     goto busywait;
17
18   y = Pid;
19   if (!pthread_equal(z, Pid))
20     goto busywait;
21 }
22
23 void unlock() {
24   x = 0; y = 0; z = 0;
25 }
26
27 void *thread(void *arg) {
28   lock(pthread_self());
29   cnt++;
30   assert(cnt == 1);
31   cnt--;
32   unlock();
33 }
34
35
36 int main(void) {
37   pthread_t t[2];
38
39   pthread_create(&t[0], 0, thread1, 0);
40   pthread_create(&t[1], 0, thread2, 0);
41
42   pthread_join(t[0], 0);
43   pthread_join(t[1], 0);
44
45   return 0;
46 }
```

1. Write down a Promela model for the above program. Use Spin to verify the assertion at line 30. Does the assertion always hold? Explain the output of Spin.

2. Use Modex (via the script `verify`) to confirm your finding.