Technische Universität München (I7) **Summer Semester 2017**
J. Křetínský / T. Meggendorfer / P. Ashok **22.03.2017**

Model Checking – Exercise sheet 3

Exercise 3.1

Consider the following C function:

```
1 void collatz(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. Insert a `do` loop that non-deterministically generates a value between 1 and 30,000 for `n` before calling the function.
3. Write down an LTL formula to check whether the function, when given a non-deterministic value generated above, always terminates. Use Spin to check the formula.
4. Write down another Promela model containing a global variable `n` and two processes: one for handling even values and one for handling odd values. Name the processes `even` and `odd`, respectively. Use Spin to simulate your result.
5. Use Spin to check the following properties:
   - Whenever the `odd` process updates the value of `n`, it never updates it again until the `even` process does it.
   - Whenever the `even` process updates the value of `n`, it never updates it again until the `odd` process does it.
   - The value of `n` is never more than the square of its original value.
   - The value of `n` is always even before it is `1`.

   Which properties are not satisfied? Use Spin to read violated trails.

Exercise 3.2

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.