CS4052: Practical 1

120010815

October 22, 2015

Contents

1 Introduction

This report reviews the implementation of the Practical 1 in Promela. The Practical consisted of three compulsory parts that involved implementing models and specifications in Spin/Promela and an extension part which involves research about the formal verification with Spin. Spin is a tool for logical verification of concurrent systems in an automated way. Promela is a verification modelling language which allows modelling concurrent systems in order to formally verify them with Spin. Using Promela specified models Spin can check the existing model for deadlocks, unexecutable code, race conditions, unspecified receptions.

2 Part 1

The first part of the practical required to implement a model with four processes - Sender, Receiver and two Intermediate processes. In order to do send numbers from 1 to 10 to the Receiver through one of the two Intermediate processes at least three channels are required. The current implementation has a channel between each of the Intermediate processes and the Sender, as well as a channel between the Intermediate processes and the Receiver. To make the system completely symmetrical it was possible to implement two channels to the Receiver, one for each of the Intermediate processes, but it was not necessary. All of the channels are global variables, as well as the two sum variables, to make it possible to track the result easier. The sum of the numbers going through each of the intermediate processes is calculated in each of those processes.

In the init process, the sum variables are initialised to 0 and the rest of the processes are also run from there. The Sender is the only process that terminates, the rest of the processes do not end and continue doing a repetitive task. This leads to the fact that the processes time out when the program ends the execution, however, as it was stated in several sources, this is not a drawback since the process does not necessarily need o terminate. In this case, it is not sensible to make every process check the global state of the program.

After several testing cases it becomes clear that the channel capacity does not affect the order of the arriving variables or generally the behaviour of the model in any way. The capacity of all the three channels can be 0 or more, all three receiving processes (both Intermediate ones and the Receiver) wait for a number to appear in the channel and take it out immediately. In addition, the model is symmetrical, therefore, there is no reason for the numbers to take more time on one channel and less time on another channel.

3 Part 2

The objectives of the second part of the practical were to implement a model for N processes where each process wants to print infinitely often but only one process can be printing at a time. This is a mutual exclusion problem for N processes. There were two channels defined, outgoing and receiving, as well as the required array of Boolean values.

The Distributor sends out numbers through the outgoing channels and waits for the acknowledgement from the receiving channel. For the simplicity purposes and the purpose of testing the Distributor only sends numbers from 100 to 1 and terminates after that. However, it is possible to send any values without any conditions in order to make the process send the numbers without terminating.

Processes P all wait on the other end of the outgoing channel, and each time one of them (defined non-deterministically) takes the number out of the channel, prints it out and sends it back to the Distributor. This way it is ensured that no other process will print while the current one prints. In order to prove this solution formally, the model was verified using Spin. The never-claim process specifies a condition that should never happen, which is that the number of true values in the print array is more than one. The state is checked when a process prints: if oneTrue is false, mark that now there is one true value; if there is already one true value, mark the corresponding boolean value (moreThanOneTrue). After this the model was verified and no errors were found.

4 Part 3

In the third part of the practical, the goal was to express the given statements using the LTL formulae. This has been implemented and tested. Here are the results of the verification without the weak fairness enforced:

a) Assertion violated

```
pan:1: assertion violated  !( !(((x%2)==1))) (at depth 6)
```

b) Ended with 1 error, search not completed

```
Warning: Search not completed
         + Partial Order Reduction
```

c) Ended with 1 error, search not completed

```
Warning: Search not completed
         + Partial Order Reduction
```

d) Assertion violated

```
pan:1: assertion violated  !( !((y<=x))) (at depth 1530)
```

e) Passed with no errors

```
No errors found -- did you verify all claims?
```

As opposed to that, with the weak fairness enforcement, the results are as such:

a) Assertion violated

```
pan:1: assertion violated  !( !(((x%2)==1))) (at depth 6)
```

b) Ended with 1 error, search not completed

```
Warning: Search not completed
         + Partial Order Reduction
```

c) Passed with no errors

```
No errors found -- did you verify all claims?
```

d) Assertion violated

```
pan:1: assertion violated  !( !((y<=x))) (at depth 1530)
```

e) Passed with no errors

```
No errors found -- did you verify all claims?
```

Therefore, the only difference is that the statement c, "It is possible that from a certain point onwards x is infinitely often odd", is not possible with the weak fairness enforcement, and it is not possible to prove if it is possible without the weak fairness enforced.

5 Part 4

Overview. In part 4 of the practical, a research about the formal verification with Spin was suggested. Promela is a high level language for system specification. Using Promela, Spin check logical consistency of a specification without having to construct a global state graph. A Spin model is a Promela program where Spin transforms given processes into an FSA. After

that it interleaves the product of the automata. As a result, Spin generates an ANSI C verifier program. So, the model consists of processes, channels between them, and variables, of which processes are always global, and variables and channels can be either global or local. The main difference of a process from a function is that processes communicate with each other; also, after a process terminates, it is not deleted but simply marked as stopped.

Formal verification. Spin simulator supports a big variety of techniques which help to verify the given model formally. The model can be simulated using random number generator for making non-deterministic decisions, as well as interactive simulation which allows the user to make the non-deterministic decisions and the guided simulation. For verification, liveness, incorrect end states, assertion violations can be checked in different modes, for example, weak fairness can be enforced or switched off. The verification also advantages from the partial order reduction techniques which allow to reduce volume of the state-space of a model. States can also be compressed via MA (minimised automaton encoding) or COM (compression), which improves the scalability of the model.

LTL verification. The linear temporal logic requirements can be checked in two ways: either in the syntax of the formula or via never-claims. In order to support dynamic creation of processes, the rubber state vector technique is used - an addition to the original state vector model where the state vector allows for the system to change its size dynamically. Experiments show that although the rubber state vector models allows for dynamic resizing of the model, the performance of this model is approximately two times worse than the performance of the static vector.

Communication in Spin. Spin supports communication using shared memory, as well as rendezvous (a port for handling a synchronous channel of size 0) and buffered message passing (handles channels of size more than 0).

Limitations One of the biggest limitations of Promela is the so called state explosion, which is the fact that even small model expressed in Promela can have a large number of states which is difficult to store and analyse. In addition, there is always a danger that the model in Promela has nothing

to do with the real model as it is not easy to represent a concurrent system correctly as a model. Therefore, there is no guarantee that verifying the model will give the correct information about it. Another limitation is that the biggest number of processes can be 255, and also SPin has no real-time capabilities since it has no notion of time.

6 Conclusion

In summary, the practical teaches the basics of the formal verification with Spin with Promela. It teaches how to model concurrent systems, how Spin verifies the model against different specifications and how formal software verification with Spin works, including different details.

7 References

1. "Using SPIN for Feature Interaction Analysis - a Case Study" (M. Calder, A. Miller)

2. "The Spin Model Checker - Primer and Reference Manual" (Holzmann, G.J.)

3. "The Model Checker SPIN" (Holzmann, G.J.)

4. "Validating SDL Specifications: an Experiment" (G. J. Holzmann)