

## CIS 540 Fall 2009: Homework 3, Due November 9

For this homework, if you wish, you can work in groups of 2. The project involves modeling, simulation, and LTL model checking in the popular software SPIN (see [spinroot.com](http://spinroot.com) for all the details about the tool). The software is already installed on **eniac**. But you may want to install it on your own PC/laptop, see the webpage for instructions to download and install. The tool **xspin** has an easy-to-use interface (click on **GettingStarted** for instructions to run a demo of **xspin**). You need to get familiar with the tool by browsing through the manual, tutorial, and examples. Here are a few notes to relate it to what we have discussed in the course.

### Modeling

The modeling language is called **Promela**. The model consists of processes executing asynchronously and communicating by sending/receiving messages on channels. Thus, it is very similar to Chapter 3. You will have to learn the syntax though. The directory **Test** contains many examples. In particular, you probably need to understand the model in **leader** which describes a protocol for leader election similar to ours.

### Simulation

You can either simulate or verify the model. In simulation mode, a single execution is generated, and illustrated using a format called *Message Sequence Charts*, a commonly used standardized notation from UML.

### Specifying Requirements

You can insert **assert** statements within the model: if an assertion fails this is reported as an error. You can also write LTL formulas. Read **leader.lt1** for sample specifications. SPIN compiles these formulas into what it calls **never** claims, which are exactly the monitors in Section 4.2.1 (you can write them directly as **Promela** models also).

### Model Checking

The basic verification algorithm in SPIN is enumerative nested depth-first-search. A large number of optimizations have been implemented to make it scale to complex models. Performance of the verifier crucially depends on setting of parameters.

### Project Description

After learning the basics of the tool, you need to model and analyze the leader election protocol described in Section 3.3.1. You may find it more interesting and useful to model some other distributed algorithm; this is perfectly fine, just let me know in advance which algorithm you wish to model.

1. Construct **Promela** model of the leader election protocol. Note that the model should capture the essential details of the protocol as described in the text of Section 3.3.1, but does not have to match the formal description of Figure 3.9 exactly. Your model should be parameterized by the number of processes and their initial identifiers so that you can instantiate different

configurations easily. One aspect you should think about is: how large should the buffers be for the links between each pair of nodes?

2. For some specific configurations (e.g, the one shown in Figure 3.8 of notes, and the one in Problem 4(a) of Homework 2), simulate the model and observe its behavior. Running the simulation repeatedly should change the order in which messages accumulate in various queues and get processed, but hopefully should not change the elected leader.
3. Specify a variety of requirements in LTL (e.g. no two processes are elected leaders, and eventually a decision is made). Run the verifier to check that the model satisfies these specifications for the initial configurations chosen in part (2) above.
4. Set up an experiment to check how the number of reachable states of the model, and time and memory used by the verifier grows with the size of the ring. For this purpose, change the number of processes from 3 to 8, and plot the above quantities (you will need to assign initial identifiers in some manner in each case).
5. Modify the basic protocol by making some local change to the model. For example, suppose when a node finds that  $id_1$  is the highest among  $id$ ,  $id_1$ , and  $id_2$ , it continues to the next phase, but does not update its  $id$  to  $id_1$  (you may want to think of a more interesting variation which you think would improve the efficiency without affecting the correctness). We want to know if this change makes the protocol incorrect. Run the verifier for previously chosen initial configurations, and check if it detects the error (and if it gives you useful feedback to locate the source of error).

You should submit (a) the Promela model of the protocol constructed in part 1, (b) the LTL specifications and results of verification in part 3, (c) the tables/charts showing experiments in part 4, (d) the modified model and results of verification in part 5, and (e) a brief description of your experience with the tool.