National University of Singapore
School of Computing
CS3234 — Logic and Formal Systems

October 18, 2004

Assignment 2

(10% of final grade)

Deadline: October 31, 2004

This assignment will expose you to verifying systems with the Symbolic Model Verifier SMV.

Running SMV

SMV is free software and can be downloaded (together with a user's manual) from

```
http://www-2.cs.cmu.edu/ modelcheck/smv.html
```

SMV is already installed on the `sf3.comp.nus.edu` machine and can be used by passing the following command to the Unix prompt.

```
/home/course/cs3234/smv/bin/smv <yourfile.smv>
```

The output of this command will provide a report on the outcome of the verification process specified in `yourfile.smv`. If the verification has failed, the report will include a couterexample showing a computation path that does not satisfy the verification condition. You can collect this report in a file by redirecting the standard output to a file:

```
/home/course/cs3234/smv/bin/smv <yourfile.smv> 2>&1 >output.txt
```

The entire output of your command is collected now in the file `output.txt`.

A copy of the users manual can be downloaded from the course web page (the link is provided next to the link for this assignment). We highly recommend reading the manual before solving this assignment.

The Dining Philosophers Problem

The Dining Philosophers problem is a widely studied resource allocation problem in distributed computing. Traditionally, the problem is described in terms of the following informal scenario. There are n philosophers (users) seated around a table, usually *thinking.* Between each pair of philosophers is a single fork (resource). From time to time, any philosopher might become *hungry,* and attempt to *eat.* In order to eat, the philosopher needs exclusive use of the two adjacent forks. After eating, the philosopher relinquishes the two forks and resumes *thinking.* A dining philosophers problem for $n = 5$ is depicted in Figure 1.

For the system described above, it would be reasonable to expect that whenever a philosopher becomes hungry, he would eventually get to eat (progress property). However, it hapens that the informal description given above allows for many formalizations, and not all satisfy the progress property given above. In what follows, you will be asked to solve two exercises. The first one provides a specification of the philosophers' protocol and requires you to implement a system of 5 such philosophers and 5 forks in SMV, and show that the progress property is *not satisfied.* The second exercise asks you to fix the system such that it satisfies the progress property, and then show in SMV that the progress property *is indeed verified.*

Exercise 1

Consider the protocol given in Figure 2, which specifies the behaviour of a philosopher. The philosopher starts in the state where he is *thinking*, and may remain in that state as

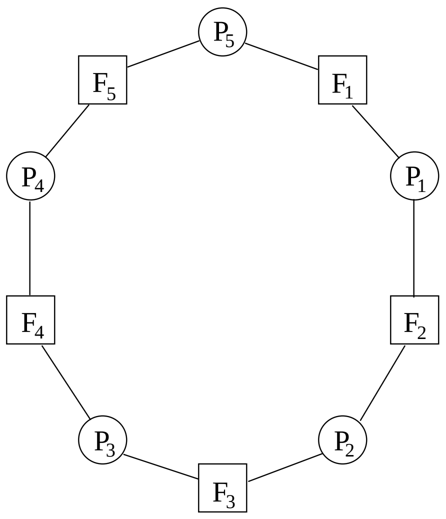

Figure 1: Dining philosophers problem for $n = 5$

long as he wants, *but not forever.* Occasionally, the philosopher may become *hungry*, and attept to acquire the forks. The philosopher will remain in the hungry state as long as the left fork is not available. Once the left fork becomes available, the philosopher *must* acquire it, and wait for the right fork. When the right fork becomes available, the philosopher must also acquire it, and begin eating. The philosopher may eat for as long as he pleases, *but not forever.* Once the philosopher has finished eating, he will relinquish the left fork first, then he will relinquish the right fork, and then resume thinking. You are required to do the follwing:

- Write the SMV specification for a dining philosophers problem of 5 philosophers and 5 forks, where the behaviour of every philosopher is the one given by the protocol in Figure 2.
- Add fairness conditions that will ensure that all philosphers are running concurrently, and that no philospher is thinking forever, or eating forever.
- Make sure that every fork is acquired by at most one philosopher at any one time. Specify safety conditions to verify this.
- Specify a progress property for every philosopher which states that once the philosopher becomes hungry, he eventually eats.

The SMV program, when run on your specification, should be able to verify that the safety property is verified, but the progress property is not. The report of a program property not being satisfied is in the form of a counterexample. Analyze the counterexample and explain informally the sequence of events that leads to a philosopher starving forever.

Exercise 2

Now, change the system you developed in your solution to your previous exercise such that progress property is satisfied. Your may pick your changes of your system only from the following set:

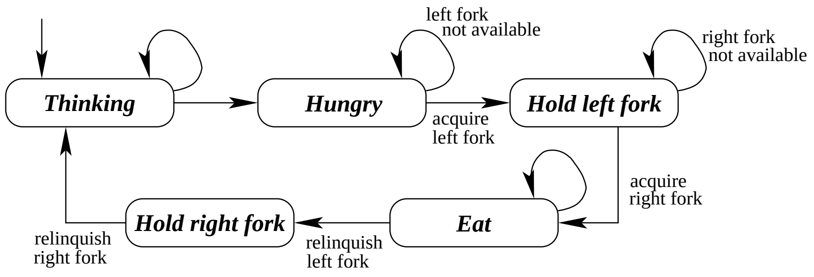

Figure 2: The behaviour of a philosopher for Exercise 1

- A philosopher may pick his forks in any order. He may choose to pick the left fork first, or the right fork first, or look for the first available fork and pick it. However, he cannot pick both forks at the same time.
- If a philosopher has acquired a fork, and the second fork is not available yet, the philosopher may choose to relinquish the fork he holds, and return to the *hungry* state.
- Each philosopher may execute a different protocol. For example, the even-ranked philosophers may try to acquire the left fork first, while the odd-ranked philosophers may try to acquire the right fork first.

You are required to do the following.

- Provide a formal specification of all the protocols of all philosophers, in the form of state graphs.
- Provide an informal explanation of why you expect that the new protocols lead to guaranteeing that the philosophers do not starve.
- Update your SMV specification to reflect the new protocols. You may need to update your fairness, safety, and progress as well, such that they would reflect the changes in the philosophers' protocols.
- Run SMV on your new specification and check that both the safety and the progress properties are verified.

What to submit

Your submission should be in the form of a small report containing the answers to the two exercises. For Exercise 1, your report should

- justify your design decisions in writing the SMV specification; remember that access to forks is exclusive, you need to explain how that is achieved;
- explain the fairness, safety, and progress conditions (i.e. why you wrote them the way you wrote them);
- include the full SMV specification (i.e. your SMV file), with comments that would make it as readable as possible;
- the output of running SMV on your file, reflecting that the safety property was verified, whereas the progress property was *not* verified, and
- a comment on the counterexample, explaining informally how the system would get to a configuration that would allow philosophers to starve.

For Exercise 2, your report should

- include the new protocols (in the form of state graphs), specifying the philosophers' behaviour;
- informal explanations on why the new protocols would guarantee that no philosopher would starve forever;
- the full SMV specification of the system, with comments that would make it as readable as possible;
- the output of running SMV on your file, reflecting the fact that all properties have been verified.

Please submit a hardcopy of your report to me before the deadline (if I'm not in the office, just slide your report under the door). Also, please submit a softcopy of your report, as well as your SMV files to me by email.