15-413: Introduction to Software Engineering

Jonathan Aldrich

Assignment 9: Model Checking

Due: Monday, November 14, 11:30am (hardcopy at beginning of class)
40 points

This assignment is a group assignment. Each project group should turn in one response to each part, with all the names of the group members.

Important note: This assignment asks you to build two Promela models. Because the models are for standard systems, it is possible that there are existing solutions. While you may use the web as a general resource, you may not look at or use any Promela code that is specifically modeling the examples in this homework. This is just a specific instance of the general rule that your work must be your own.

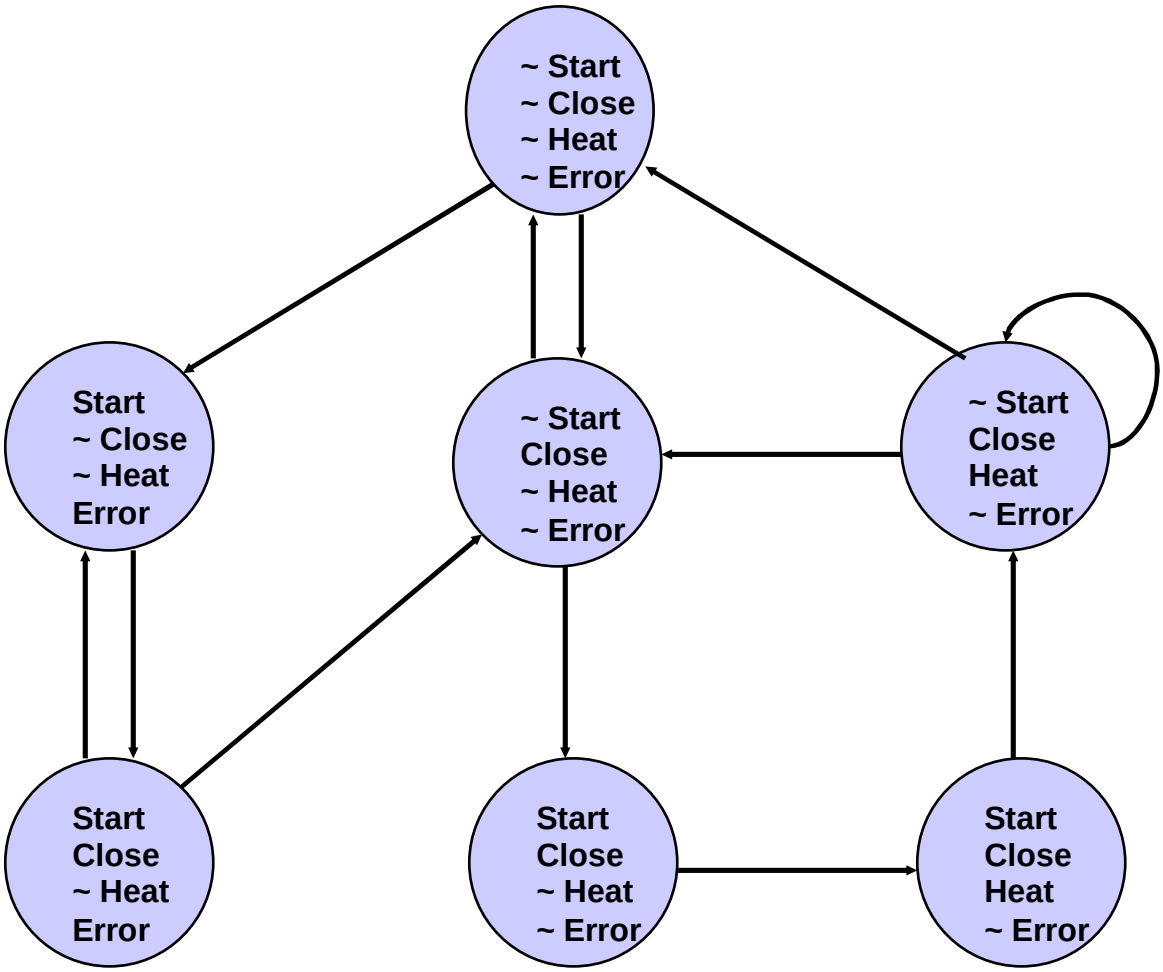
1. Temporal Logic (8 points)

Formally state each of the following properties for the system on the next page using CTL temporal logic:

a) If the heat is on, and then the door is opened, the heat immediately goes off
b) It is possible to eventually get out of the error condition
c) If the door is open, it will eventually close
d) If we are in an error condition, the door will be closed before we get out of the error condition

2. Model Checking (4 points)

Pick two of the properties described in part 1. Use the CTL model checking algorithm on the system on the next page, to show in which states the property holds. Show your work by (a) converting each formula to use only the temporal operators EX, EU, and EG, (b) assigning a letter to each subformula of the converted formulas, and (c) showing which nodes of the graph are marked with (the letter of) each subformula.


~ Start
~ Close
~ Heat
~ Error
Start
~ Close
~ Heat
Error
~ Start
Close
~ Heat
~ Error
~ Start
Close
Heat
~ Error
Start
Close
~ Heat
Error
Start
Close
~ Heat
~ Error
Start
Close
Heat
~ Error

3. Promela warm-up (8 points)

If two or more concurrent processes execute the same code and access the same data, there is a potential problem that they may overwrite each others results and corrupt the data. The mutual exclusion problem is the problem of restricting access to a critical section in the code to a single process at a time, assuming *only* the indivisibility of read and write instructions. (The problem disappears if one can assume an indivisible test-and-set instruction.) The problem and a first solution were first published by Dijkstra.

The following `improved' solution appeared one year later in the same journal (*Comm. of the ACM*, Vol. 9, No. 1, p. 45) by another author. It is reproduced here as it was published (in pseudo Algol).

```
 1 Boolean array b(0;1) integer k, i, j,
 2 comment process i, with i either 0 or 1 and j = 1-i;
 3 C0: b(i) := false;
 4 C1: if k != i then begin
 5 C2: if not (b(j) then go to C2;
 6     else k := i; go to C1 end;
 7     else critical section;
 8     b(i) := true;
 9     remainder of program;
10     go to C0;
11     end
```

Model the solution in *Promela*, and prove or disprove the correctness of the algorithm. Note: you're on your honor not to find a solution on the web for this one.

Email your Promela program to the TA, and turn in a printout of the program and the output of SPIN.

4. Modeling Two-Phase Commit (20 points)

There is a nice description of the Two-Phase Commit problem in the following paper (just use the standard algorithm, not the variants):

http://citeseer.ist.psu.edu/188038.html

Model a distributed system with Two-Phase Commit (2PC). You should model the system as performing one commit after another in a loop. There should be at least two participants plus a coordinator. You must model the possibility of failure at *any* point in the protocol, by any participant. After a failure, the process should restart and consult its "log file" (which you may represent in any way of your choice) to determine where it was in the protocol and continue.

a) Write a temporal logic property for your system that represents liveness, i.e. that the system does not get into deadlock and stop processing transactions. Turn in the property, along with the output of spin showing whether it holds or not.

   In general, this kind of property should hold of 2PC assuming that processes eventually recover and the system is modeled with some kind of "fairness." However, modeling fairness is beyond the scope of this course, so your property may not hold in this assignment.

b) Write a temporal logic property that states that different processes do not disagree about the results of a commit, even if they fail at some point. Turn in the property, along with the output of spin showing whether it holds or not.

   If you correctly implemented 2PC and wrote the right property, it should hold.

c) Email the Promela code for your basic model to the TA.