CS 511: Assignment #6
Model Checking with Erigone

Due Tuesday, December 8 at 11:59pm.

1 Assignment

Use the Erigone model checker at
`https://code.google.com/p/erigone/downloads/list` to learn about the value of model checking for testing (models of) concurrent programs. Erigone is intended for educational use. It implements a large subset of the functionality of the popular Spin model checker, but Erigone is much easier to use than Spin.

1.1 Part 1

Here is the "extreme interleaving" counting program that was distributed in class:

```
Two threads, T1 and T2, execute this program concurrently:

for i = 1 to 5 {
    temp = n;
    n = temp + 1;
}

Each thread has its own copy of i and temp.  n is shared between the
threads, temp is not (each thread has its own temp).  Initially n=0.
Each body statement is atomic.  The threads do not know and cannot
influence the behavior of the scheduler; any interleaving of the body
statements is possible.

There is no locking to implement a critical section, so n's final
value is uncertain.  When both threads have finished executing, what
are the possible final values of n?
```

Use Erigone/Promela to show that the final value of `n` may be outside the range 5-10. Write a Promela model that includes an assertion that is violated if `n` is out of range when the two processes are finished, then run Erigone in verification mode. The verification run will end with an assertion-failed report.

This task requires figuring out how to do a counting loop in Promela. Also, it may be necessary to raise the limit on the number of simulation steps; the default limit of 10,000 may not be enough.

Submit:

1a. Promela model source code file.

1b. Erigone command line for the verification run.

1c. Erigone verification output that shows that `n` may finish outside the range 5-10.

1.2 Part 2

We saw in class that it is possible for the "extreme interleaving" program to finish with `n=2`. Write an assertion that is violated if `n` is 2 at the finish then run Erigone in verification mode. The run will end with an assertion-failed report, leaving a "trail" file. Then use the trail file to run Erigone again, this time in "guided simulation" mode that indicates step-by-step how the program executes to finish with `n=2`.

Submit:

2a. Promela model source code file.

2b. Erigone command line for the verification run.

2c. Erigone verification output that shows that n may finish as 2.

2d. Erigone command line for the guided simulation run.

2e. File that contains Erigone guided simulation output showing the step-by-step actions that finish with `n=2`.

1.3 Part 3

The "extreme interleavings" program is a buggy program that behaves unpredictably because it doesn't access the shared variable `n` in a critical section. Correct the program by placing each access of `n` inside a critical section then run the corrected program in verification mode to confirm that `n` always finishes as 10.

Submit:

3a. Promela model source code file.

3b. Erigone command line for the verification run.

3c. Erigone verification output that shows that `n` always finishes as 10.

1.4 Part 4

C code for a simplified Assignment 3's "cart crossing problem" is posted in Canvas. This code doesn't implement the "next cart on the right" rule that was required for that assignment. Instead, in this simplified code the next cart to enter the intersection is chosen by chance, based on thread scheduling.

Develop a Promela model of this code then verify its safety. (In this case, safety means: no more than one cart in the intersection at any moment.) Each thread should have an array of cart numbers. Keep the array size small to make verification tractable.

Submit:

4a. Promela model source code file.

4b. Erigone command line for the verification run.

4c. Erigone verification output that shows that all carts pass through the intersection in order.

2 Submission Instructions

You must work alone on this assignment. Use Canvas to submit all necessary files by the indicated date and time. Moodle will shut you out at the deadline. Since your clock and Moodle's clock may differ by a few minutes, be sure to submit at least several minutes before the deadline.

Place all single-line submission products – like the Erigone command lines and the single-line simulation outputs – into a single text file. Clearly label which lines correspond to which submission products using the numbering above; e.g., 2b, 2c, etc. If the TAs have trouble understanding your submission then your grade will decline.

Please note: *late work is not accepted; late submissions will not be graded and will receive a score of zero.*