CS 6110 – Formal Methods – Spring 2011

January 27, 2011

Assignment 2, Handed out: January 20 – NEW due dates noted below

New Due Dates

- Question 1: Due Jan 24th monday midnight
- Questions 2 and 5: Jan 26th wednesday midnight
- Question 6: Due Feb 2nd (Wed) midnight.
- Remaining: Due Jan 31st midnight

1. (5 points) Read Chapter 21 from my book. Also read the DDHY92 paper linked at http://www.eng.utah.edu/~cs6110/week2/DDHY92.pdf. Finally, Skim through many-promela-exs that will be the subject of the lecture on Jan 25th. These are linked at http://www.eng.utah.edu/~cs6110/week3/many-promela-exs.pdf. Write a one-page summary covering these readings.

2. (5 points) Develop your own example to understand invariants and inductive invariants geometrically, as was presented in class. Draw these invariants out as sets of points as I showed in class.

3. (10 points) There is a large urn with a fixed but arbitrary amount of black and white balls (“initial state”). There is a supply of extra balls whenever you want. Someone executes this loop:

   - If there are fewer than two balls in the urn, terminate.
   - Takes two balls out of the urn.
   - Now, if both of the same color, throw back a black (could be the one you took out; could be from the extra supply).
   - Else throw back a white.

Answer these questions briefly (a sentence in most cases):

   - Will this procedure terminate?
   - For all initial states with at least two balls, what is the final state of the urn?
   - Write this as a loop. Write down an inductive invariant for the loop. Show that it is inductive.
4. (10 points) An iterative GCD is roughly as follows:

function gcd(x : int, y : int): int;
begin
  while (x!=y) do
    if (x>y) then x := x-y;
    else // (x<y)
      y := y-x;
  endwhile;
  return x;
end;

Annotate this pseudocode with suitable pre- and post conditions. Then, write a loop invariant that is inductive for the loop. Show that your loop invariant is “powerful enough” to infer the answer assertion (postcondition). Show your work.

5. (25 points) The rules for the man, wolf, goat, cabbage problem were described in the slides. Formulate this problem as a Murphi transition system and solve it (again as described in the slides).

6. (45 points) Model Lamport’s Bakery Protocol in Murphi for a fixed number (say, 4) of processes. See if you can employ scalar sets for some of the indices. Come up with the necessary invariants to prove mutual exclusion. Show that these invariants hold. Develop a readable model that generates a low number of states. Present a typescript session of your work and a brief (one page) writeup of your observations.

7. Extra (say 20 points): See if you can establish the “extra” features of this protocol using Murphi. Lamport mentions two such properties: one being that a node failure still preserves the overall workings; the second being that a concurrent read and write of a location may return “garbage” values and yet the protocol will work. I don’t know how long these extra credit parts will take, hence the points are open-ended for now. Bound your work.