CS 6110 – Formal Methods – Spring 2011

January 14, 2011

Assignment 1, Handed out: January 13 – due dates noted below

These are mainly for class discussions which will be “graded” through questions and interactions with you:

1. Go through the protocols `2_peterson.m` and `n_peterson.m` in the Murphi examples directory. Be ready to discuss these next week when called upon. Run these protocols and tell me what you observed. Understand all Murphi constructs in it.
2. Read through the User Manual of Murphi. Be ready to ask me questions. I'll ask a few too.
3. Be sure you understand the workings of `two-bytes.m`.
4. I will give you hints on how my coding of bubble sort is going. You can take these hints and see if you can finish up coding. Look online for `illustration2`.
5. Please read Chapter 2 of Stern's dissertation. We can all have a discussion around the items there.

This is a problem you must all work on individually.

In an old algorithms textbook, the following Bubblesort routine was (actually) given. I give it in pseudo-code. I also give it in Promela, another language like Murphi. Your task is to code up this model suitably in Murphi. Then write an invariant to claim that this protocol sorts.

Tell me whether the invariant is true, and if not why not. Hint: I am not claiming that this Bubble sort is correct. So if the Bubble sort is not correct, your final invariant should help you debug it.

Also my pseudo-code in `illustration2` is just a rough sketch of some work—it is not a complete piece of code yet. It won't even compile.

Schedule for your presentation:

- Initial Murphi coding due by next week Monday midnight (1/17). You need not have fully tested it, but I must see enough Murphi code to know that you have made progress. Submit it as `version1_bubsort_yourname.m`.
- More working Murphi code to be submitted Wednesday midnight (1/19). Name your submission `version2_bubsort_yourname.m`. We will discuss your work then. Document your progress since previous version.
- Final working code to be submitted by Monday midnight of 1/24. We will set up a handin directory. Call it `final_bubsort_yourname.m`.

- **EXTRA CREDIT: Model the rules of a correct Bubble sort program and verify it using Murphi.** You don't need to follow the same code pattern as below – you can write your own Bubble sort pseudo-code and verify that using Murphi.

```
procedure bubblesort;
 var j, t: integer;
 begin
 repeat
   t:=a[1];
   for j:=2 to N do
     if a[j-1]>a[j] then
     begin t:=a[j-1]; a[j-1]:=a[j]; a[j]:=t end
 until t=a[1];
 end;

/* -- This happens to be a Promela model for the above -- */

#define Size 5
#define aMinIndx 1
#define aMaxIndx (Size-1)

/* Gonna "waste" a[0] because Sedgewick uses 1-based arrays */
active proctype bubsort()
{ byte j, t;   /* Init to 0 by SPIN */
  bit a[Size]; /* Use 1-bit abstraction */

  /* Nondeterministic array initialization */
  do ::break ::a[1]=1 ::a[2]=1 ::a[3]=1 ::a[4]=1 od;

  t=a[aMinIndx];
  j=aMinIndx+1;

  do /* First ‘‘repeat’’ iteration */
  :: (j >(aMaxIndx)) -> break /*-- For-loop exits --*/
  :: (j<=(aMaxIndx)) ->
      if
      :: (a[j-1] > a[j]) -> t=a[j-1]; a[j-1]=a[j]; a[j]=t
      :: (a[j-1] <= a[j])
     fi;
     j++
  od;

  do /* Subsequent ‘‘repeat’’ iterations */
  :: t!=a[1] ->
     t=a[aMinIndx];
     j=aMinIndx+1;
```

```
      do
      :: (j >(aMaxIndx)) -> break /*-- For-loop exits --*/
      :: (j<=(aMaxIndx)) ->
         if
         :: (a[j-1] > a[j]) -> t=a[j-1]; a[j-1]=a[j]; a[j]=t
         :: (a[j-1] <= a[j])
         fi;
         j++ /*-- for-index increments --*/
      od         /*-- end of for-loop --*/

   :: t==a[1] -> break
   od;

   t=1; /*-- Comb from location-1 to look for sortedness --*/
   do
   :: t < aMaxIndx-1 -> t++
   :: t > aMinIndx   -> t--
   :: a[t] > a[t+1]  -> assert(0) /*- announce there is a bug! -*/
   od
 }
```