CS511 – Endterm – Topic 4 – December 9, 2020

Exercise 1 (*Message Passing in Erlang, 5 pts*)

The following program in Erlang implements the Producers/Consumers synchronization pattern using message passing. It is missing the implementation of `loopPC/4` which you are asked to complete. The arguments to `loopPC` are as follows:

- `Cs`: Number of consumers that started consuming
- `Ps`: Number of producers that started producing
- `MaxBufferSize`: Buffer size (constant, greater than 0)
- `OccupiedSlots`: Number of currently occupied slots in the buffer

Note: Multiple producers and consumers should be allowed.

Deliverable: file pc.erl

```erlang
consumer(Id,Buffer) ->
    timer:sleep(200),
    io:fwrite("Consumer ~p trying to consume~n",[Id]),
    Ref = make_ref(),
    Buffer!{start_consume, self(), Ref},
    receive
        {ok_to_consume, Ref} ->
            io:fwrite("Consumer ~p consuming~n",[Id]),
            Buffer!{end_consume},
            io:fwrite("Consumer ~p stopped consuming~n",[Id]),
            consumer(Id,Buffer)
    end.

producer(Id,Buffer) ->
    timer:sleep(1000),
    io:fwrite("Producer ~p trying to produce~n",[Id]),
    Ref = make_ref(),
    Buffer!{start_produce, self(), Ref},
    receive
        {ok_to_produce, Ref} ->
            io:fwrite("Producer ~p producing~n",[Id]),
            Buffer!{end_produce},
            io:fwrite("Producer ~p stopped producing~n",[Id]),
            producer(Id,Buffer)
    end.

loopPC(Cs, Ps,MaxBufferSize,OccupiedSlots) ->
       %% implement me

startPC() ->
    Buffer = spawn (fun() -> loopPC(0,0,10,0) end),
    [ spawn(fun() -> producer(Id,Buffer) end) || Id<- lists:seq(1,10)],
    [ spawn(fun() -> consumer(Id,Buffer) end) || Id<- lists:seq(1,10)].
```

This page is intentionally left blank. You may use it for writing your answers.

Exercise 2 (*Spin, 5 pts*)

Below is exercise 2 from eb5 (reformulated in terms of a feeding lot instead of a restroom):

> In a zoo there is a common feeding area for exotic mice and exotic felines. The feeding area has a common feeding lot that holds up to 2 animals. The feeding area can be used by both mice and felines, but cannot be used by mice and felines at the same time for obvious reasons.

Here is a proposed solution in Promela where we have 3 felines and 3 mice.

```
byte mice = 0;
byte felines = 0;
byte mutexMice  = 1;
byte mutexFelines =1;
byte mutex = 1;

inline acquire(sem) {
  atomic {
    sem>0;
    sem--
  }
}

inline release(sem) {
  sem++
}

active [3] proctype Mouse() {
  acquire(mutex)
  acquire(mutexMice);
  mice++;
  if
    :: mice==1 -> acquire(mutexFelines);
    :: else -> skip;
  fi
  release(mutexMice);
  release(mutex);

  // access feeding lot

  acquire(mutexMice);
  mice--;
   if
      :: mice==0 -> release(mutexFelines);
      :: else -> skip;
  fi
  release(mutexMice);
}



active [3] proctype Feline() {
  acquire(mutex);
  acquire(mutexFelines);
  felines++;
  if
    :: felines==1 -> acquire(mutexMice);
    :: else -> skip;
  fi
  release(mutexFelines);
  release(mutex);

  // use feeding lot

  acquire(mutexFelines);
  felines--;
  if
    :: felines==0 -> release(mutexMice);
    :: else -> skip;
  fi
  release(mutexFelines);
}
```

1. Show, using assertions in Spin, that the solution is correct in the sense that there cannot be mice and felines at the same time in the feeding lot.

2. Unfortunately, the proposal is not quite right because it is possible for three felines (or three mice) to be in the feeding lot at the same time. This is incorrect since the feeding lot only has capacity for two animals. Show that the solution is incorrect in this regard, using assertions.

3. Propose a fix to the problem from the previous item and use spin to show that your fix is indeed correct.

Deliverables:

1. Item 1. File zoo1.pml (including any assertions) + output1.txt (output from jpsin indicating that there are no errors)

2. Item 2: File zoo2.pml (including any assertions) + output2.txt (output from jpsin indicating offending trail)

3. Item 3: File zoo3.pml (including any assertions) + output2.txt (output from jpsin indicating that there are no errors)

This page is intentionally left blank. You may use it for writing your answers.