CSC3304 Software verification technology

The exam paper will be divided into 3 sections.

- The Laws of Programming: Floyd/Hoare logic and the challenge of proof (10/50).
- **Model Checking (Promela/Spin) (15/50) .**
- Theorem Proving (Symbolic reasoning tools) & advanced static analysis in practice (JML) (25/50).

Model Checking Revision Guide

There will be 4 questions.

Question A

A Promela model with 2 or 3 processes is presented, and you must describe the operation of one of the processes.

Reading:

Lecture notes (week 3).

The concise Promela reference:

http://spinroot.com/spin/Man/Quick.html

Message Channels:

http://www.informit.com/articles/article.aspx?p=169103&seqNum=6

To look out for:

len(q) predefined function to determine the number of messages stored in channel "q".

The role played by eval, and [ ]. For example:

```
(a > b && qname?[msg0])
```

The expression `qname?[msg0]` is true precisely when the receive statement `qname?msg0` would be executable at the same point in the execution, but the actual receive is not executed, only its precondition is evaluated. Any receive statement can be turned into a side effect free expression in a similar way, by placing square brackets around the list of message parameters. The channel contents remain undisturbed by the evaluation of such expressions.

Previous Exam Example:

Consider the following PROMELA model

```
#define top 3
#define floors 4
#define ground 0
chan opendoor[floors] = [0] of {byte};
chan closedoor[floors] = [0] of {byte};
chan opend[floors] = [0] of {byte};
chan closed[floors] = [0] of {byte};
chan call = [0] of {byte};
chan doorbutton = [0] of {byte};
byte floor=ground;
bool calls[floors];

proctype door(byte i)
{ byte any;
  do
  :: opend[i]?any -> {opendoor[i]!any; closedoor[i]!any; closed[i]?any}
  od
}

proctype lift()
{
  byte x;

  bool uptag=true;
  do
  :: call?x -> calls[x]=true
  :: calls[floor] -> {opendoor[floor]?x;
                      do
                      :: doorbutton?x -> calls[x]=true
                      :: true -> break
                      od;
                      closedoor[floor]?x;
                      calls[floor]=false
                      }
  :: !calls[floor] -> if
                      :: (floor!=top) && uptag -> floor++
                      :: (floor!=ground)&& !uptag -> floor--
                      :: (floor==top) -> uptag=false
                      :: (floor==ground) -> uptag=true
                      fi
   od
}

proctype user(byte f; byte t)
  {call!f;opend[f]!f; doorbutton!t; closed[f]!f}

init {
       run door(ground);
       run door(1);
       run door(2);
       run door(top);
       run lift();
       run user(ground,top);
       run user(1,top);
       run user(2, ground)
     }
```

Question: Explain how the `lift` process works.

Looking for in answer: Explanation of the role played by any guards, describe the relationship between the process and the other process, and explain how the channels are used. If there are any **assertions** within the process, then you will be asked to describe the role that they play within the model.

Question B

Model checking, definition of key concepts:

Model checking, **Safety, Liveness**, **Deadlock, Livelock, Progress Cycles, Acceptance cycles, Never Claims, Fairness**.

Reading:

Lecture Notes (Blackboard: week3 & week 4)

Previous Exam Example:

Question: Define the term *deadlock* in the context of a PROMELA model.

Looking for in answer: Definition of deadlock generally, **and** what deadlock is in the context of a Promela model. (Similar question could be asked for any of the above concepts)

Possible answer: *Deadlock: A deadlock is a situation where two or more actions are each waiting for the other to finish, and thus neither ever does. In the context of PROMELA deadlock happens when a process in a model does not terminate. Non-termination means reaching an invalid end state.*

Other Possible Questions:

If there are any deadlocks within the model presented, then you may be **asked to describe when the deadlock will occur** (Coursework should have prepared you well for this!). When describing this, state where in the code the deadlock will occur, and what sequence of events lead to the deadlock.

F***or example****, in the model checking coursework, deadlock may happen in the chef process when the chef is waiting for a message that may never arrive. Sequence of events leading to this? There are no customers making orders in the restaurant.*

Question C

Linear Temporal Logic Expressions.

Reading:

Lecture Notes (Blackboard: week 4 & week 5)

Previous Exam Examples:

Question: Give an informal but precise description of the LTL formula: <>(p^[]q)

Answer: *Eventually p is true and q continues to be true for all future states.*

Question: Write an English translation of the LTL formula: <>[]p

Answer: It will eventually be the case that p will be true for every state in the path.

Question D

Linear Temporal Logic Negation.

Reading:

Examples in Lecture Notes (Blackboard: week 4 & week 5)

Previous Exam Examples:

Question: Produce the negated form of <>(p^[]q) that could be easily translated into a finite state automaton showing how you transformed one formula into the other.

Answer:

!<>(p^[]q) ⇔

[]!(p^[]q)⇔

[](!pV![]q)⇔

[](!pV<>!q)⇔

Questions? Ellis.solaiman@ncl.ac.uk