Midterm : CS 4271: Critical Systems and their Verification

February 20 2004, 10-11:30 AM

Instructions to Candidates

- Answer **ALL** questions. This exam has five questions.
- All answers **MUST** come with the correct explanations. There is no credit for guessing. A correct answer without the correct explanation will receive no marks.
- Answers must be written in the space provided in this booklet;
  otherwise they will not be graded.
- This is an **OPEN BOOK** examination. You are allowed to bring in any books/lecture notes etc., but not a laptop.
- You can ask for extra sheets for rough work.
- **PLEASE WRITE YOUR MATRICULATION NUMBER BELOW**.

MATRICULATION NO.:

(This portion is reserved for the examiner's use only)

Question		Marks
Question A	5	
Question B	5	
Question C	4	
Question D	3	
Question E	3	
TOTAL	20	

A. 5 marks

Two students are taking the CS4271 exam. We must ensure that they cannot leave the exam hall at the same time. To prevent this, each student reads a shared token n before leaving the hall. The shared token is an arbitrary natural number. The global state of the system is given by $\langle s_1, s_2, n\rangle$ where s_1 and s_2 are the local states of students 1 and 2 respectively. Note that $s_1 \in \{in, out\}$, $s_2 \in \{in, out\}$. The pseudo-code executed by the two students is given below. The two student processes are executed asynchronously. Every time one process is scheduled, it *atomically* executes one iteration of its loop. Initially s1 =in and s2 = in.

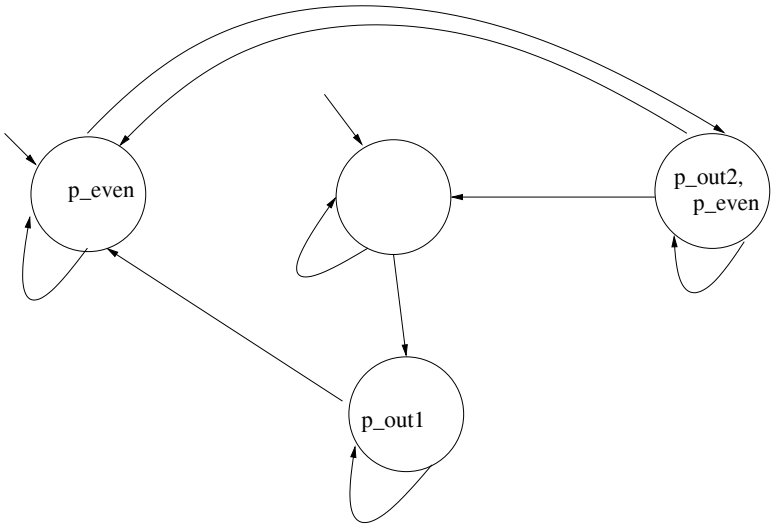
```
do forever{
   if (s1 = in and n is odd) then {s1 := out}
   else if (s1=out) then {s1:=in; n:=3*n+1}
   else {do nothing}
}
```

```
do forever{
   if (s2 = in and n is even) then  {s2 := out}
   else if (s2=out and n is even) then {s2:=in; n:=n/2}
   else {do nothing}
}
```

To represent the above program as a Kripke Structure, we need to introduce atomic propositions corresponding to s1, s2 and maintain approximate information about n. I suggest using an atomic proposition p_even which is true when n is even and false otherwise. Thus, the global state will be given by the valuation of three atomic propositions. Draw the Kripke Structure for the above program.

Answer:

The set of all atomic propositions AP is $AP = \{p\_out1, p\_out2, p\_even\}$. The meaning of p_even is given in the question. The meaning of p_out1 (p_out2) is as follows: it is true if the first (second) process is in the `out` state and false otherwise. The Kripke Structure is given below. Note that there are two initial states, since nothing is known about the initial value of n.



B. 5 marks

State the mutual exclusion property (both students cannot leave the exam hall at the same time) in Computation Tree Logic (CTL). Prove it using the explicit-state CTL model checking algorithm discussed in class.

Answer:

The mutual exclusion property is $AG\neg(s1 = out \wedge s2 = out) \equiv AG\varphi$. For conveninece we use $s1 = out$ ($s1 = in$) to denote p_{out1} ($\neg p_{out1}$) and $s2 = out$ to denote p_{out2} ($\neg p_{out2}$).

Now $AG\varphi = \neg\neg AG\varphi = \neg EF\neg\varphi = \neg E(trueU\neg\varphi) = \neg E(trueU(s1 = out \wedge s2 = out))$

The model checking algorithm now proceeds as follows. We use St_ψ to denote the set of states satisfying ψ.

1. Compute $St_{s1=out}$
2. Compute $St_{s2=out}$
3. Compute $St_{s1=out \wedge s2=out} = St_{s1=out} \cap St_{s2=out}$
4. Compute St_{true} = S, set of all states
5. Compute $St_{trueU(s1=out \wedge s2=out)}$. These are all the states form which the violating states can be reached.
6. Compute $S - St_{trueU(s1=out \wedge s2=out)}$ and check whether the initial state of the global automata belongs to this set.

C. 4 marks

Consider the following program with two processes, which are composed *asynchronously*. Assume that initially `x = y = 0`, and each assignment is executed atomically.

```
y := 1          a := x
x := 1          b := y
```

What are the possible values of `a` and `b` when the program terminates? For each of these possible values draw a trace that will generate it. For any value that is not possible, you must also argue why it is not possible.

Answer:

- a = 0, y = 0 : Can be generated by the trace a:=x; b:= y; y := 1; x := 1
- a = 1, b = 1 : Can be generated by the trace y := 1; x := 1; a:=x; b := y
- a=0, b = 1: Can be generated by the trace y:=1; a:=x; b := y; x := 1
- a=1, b = 0: This is not possible for the following reason:
  - since a = 1, therefore $x := 1 < a := x$ where $<$ denotes "happens-before".
  - since b= 0, therefore $b := y < y := 1$
  - but as per the program order of the right-hand thread we have $a := x < b := y$ that is

$$x := 1 < a := x < b := y < y := 1$$

  - This violates the program order of the left thread since y:= 1 should happen before x := 1

D. 3 marks

Specify the following as a linear time temporal logic formula: "*Always, between processor A writing a value and processor B reading the value, the value is cleared from processor A's cache.*" You can assume the following atomic propositions:

- wr_A : true exactly when processor A writes a value, false at all other times.
- rd_B : true exactly when processor B reads a value, false at all other times.
- clr_A : true exactly when a value is cleared from processor A 's cache, false at all other times.

Your answer should contain these three atomic propositions and LTL operators as well as boolean operators. Explain your answer.

Answer:

$G((wr_A \ \wedge \ F \ rd_B) \Rightarrow (F \ clr_A \wedge (clr_A R \neg rd_B))$

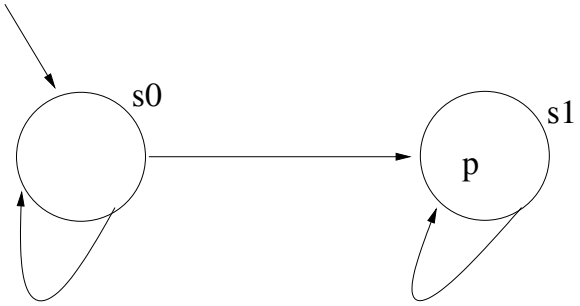
This means that if wr_A is followed by rd_B then rd_B is de-asserted until clr_A.

E. 3 marks

Consider the LTL formula GFp and the CTL formula AGEFp where p is an atomic proposition. Give an example of a Kripke Structure which satisfies AGEFp but does not satisfy GFp. You may assume that p is the only atomic proposition for constructing the labeling function. You should explain why your constructed example does not satisfy GFp.

Answer:

Here is an example Kripke Structure.



It does not satisfy GFp because the path $s0^{\omega}$ does not satisfy GFp (even the property Fp is not true for this path since p does not even hold once in this path).