NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1, 2004/2005
CS6214 - AUTOMATED SOFTWARE VALIDATION

27 September 2004 Time Allowed: 1 Hour 30 minutes

INSTRUCTIONS TO CANDIDATES

1. Answer **ALL** questions.
2. Answer **ALL** questions in the space provided in this booklet.
3. This is an **OPEN BOOK** examination.
4. **Please write your Matriculation Number below.**

MATRICULATION NO.:

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

Question	Marks	Remark
Question A 10		
Question B 11		
Question C 9		
Total 30		

A. (4 + 6) = 10 marks

1. Consider the program fragment `x = 5; x = x +1 ; x = x - 1; y = x`

   Suppose we want to prove that $y = 5$ at the end of the program. Construct a predicate abstraction that is insufficient to prove this property; also construct a predicate abstraction that is sufficient to prove the property. You are only allowed to abstract the data store of the program via predicates, but the control flow should not be abstracted. You can assume the model extraction scheme discussed in class.

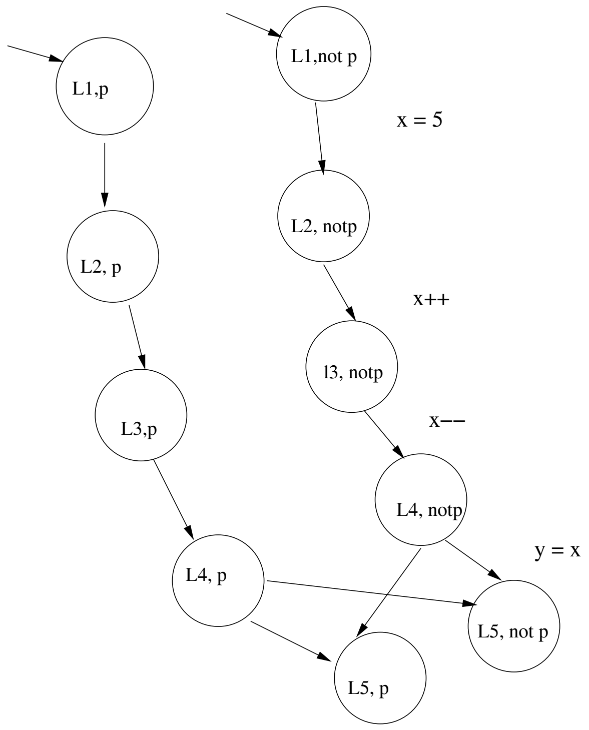
   ANSWER: The abstraction $\{y == 5\}$ is insufficient to prove the property. The abstraction $\{x == 5, x == 6, y == 5\}$ is sufficient.

2. Construct the finite state transition systems resulting from the two predicate abstractions in part 1 of this question.

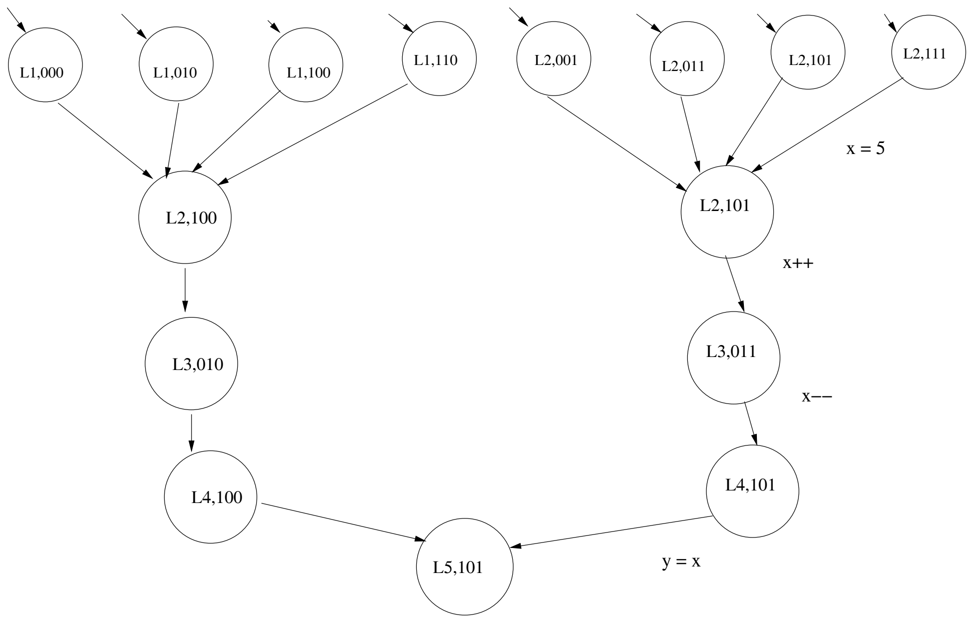
   Answer: We number the control locations as follows

   $$(L1 : x = 0); (L2 : x = x = 1); (L3 : x = x - 1); (L4 : y = x); L5 :$$

   For the first part, we have only one predicate monitoring the data store i.e. y== 5; we call it p. The finite state transition system is



   For the second part, there are 3 predicates x==5, x==6, y == 5. Call them p1, p2, p3. The finite state transition system is



B. (3 + 2 + 3 + 3) = 11 marks

1. Consider the two LTL formula $\mathbf{G}(\varphi \Rightarrow \mathbf{X}\varphi)$ and $\mathbf{G}(\varphi \Rightarrow \mathbf{G}\varphi)$. Are these two formulae equivalent ? If so, provide a proof; if not, provide an example to illustrate why these formulae are not equivalent. Assume that φ is an arbitrary LTL formula.

   Answer: Consider a path π satisfying $\mathbf{G}(\varphi \Rightarrow \mathbf{X}\varphi)$. Let k be an index ≥ 0 such that $\pi^k \models \varphi$ where π^k denotes the suffix of π starting from position k. Then, clearly $\pi^{k+1} \models \varphi$. Again this means $\pi^{k+2} \models \varphi$. A simple induction on i is able to establish that for any natural number i we must have $\pi^{k+i} \models \varphi$. This means $\pi \models G(\varphi \Rightarrow G\varphi)$.

   The proof in the other direction is trivial and follows from the definition of G and X operators.

2. Explain in English the meaning of the following LTL property

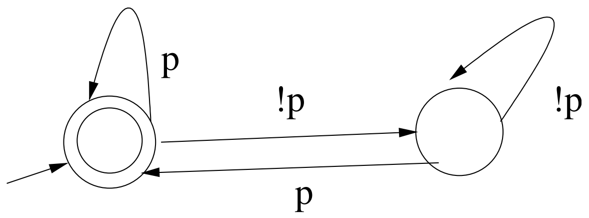
$$p \wedge \mathbf{X}\neg p \wedge \mathbf{G}((p \Rightarrow \mathbf{XX}p) \wedge (\neg p \Rightarrow \mathbf{XX}\neg p))$$

   Assume that p is an atomic proposition.

   Answer: Any path π will satisfy the above formula if and only if all odd numbered positions of π satisfy p and all even numbered positions of π do not satisfy p. Note that p is an atomic proposition, and hence a state formula.

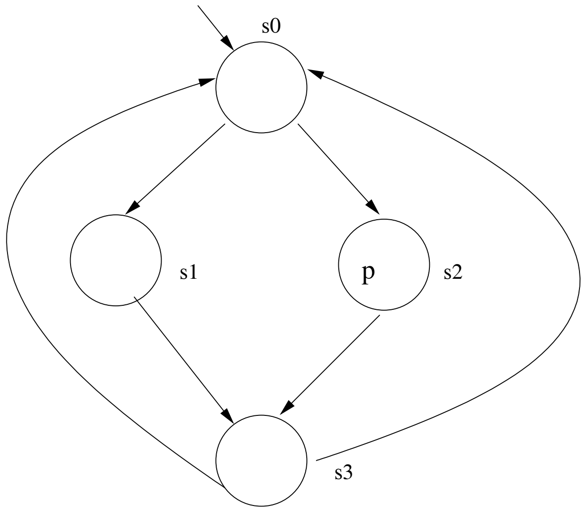
3. Suppose we want to verify $\neg\mathbf{GF}p$ for a Promela program P, where p is an atomic proposition. What is the Buchi automata (internally constructed by SPIN) that is composed with the state transition system of P, as per the model checking algorithm of SPIN ?

   Answer: SPIN will construct the Buchi automata for GFp, *i.e.* an automata which accepts all infinite strings with infinitely many occurrences of p. Here it is:



4. Give an example to show that the LTL formula **GF**p is not equivalent to the CTL formula **AGEF**p.

   Answer: The following Kripke Structure has only one state satisfying p. It satisfies AGEFp since this state satisfying p (s_2) is always reachable. It does not satisfy GFp since the trace $(s_0, s_1, s_3)^\omega$ does not even contain finitely many occurrences of p.



C. (4 + 5) = 9 marks

1. Consider a concurrent program consisting of one producer and one consumer process. Both the processes access a shared data structure. Design a protocol to ensure mutual exclusion of the shared data structure access. Encode your protocol in Promela language.

   Answer:

   We consider a simple protocol with a single `turn` variable. The producer (consumer) is allowed to access the shared buffer if and only if `turn = P` (`turn = C`). This can be encoded in Promela as follows

```
toktype = {P, C};
toktype turn = P;

active proctype producer()                active proctype consumer()
{                                         {
  do                                        do
  :: (turn == P) -> /*produce 1 item*/      :: (turn == C) -> /* consume 1 item */
                    turn = C                                   turn = P
  od                                        od
}                                         }
```

2. Apart from mutual exclusion, what other desirable temporal properties should be followed by such a protocol for shared data access ? Explain whether these properties are satisfied by your designed protocol. Are there any undesirable features specific to your designed protocol which can be captured as a temporal property ?

   Answer: We need to make sure that neither the producer nor the consumer is starved. This can be made sure by the LTL properties $GF(turn == P)$ and $GF(turn == C)$. These properties are satisfied by the simple protocol we have constructed since we simply do a round-robin scheduling of the producer and consumer processes.

   The round-robin scheduling policy of our protocol is itself undesirable This is because if one of the processes is hungry for access and the other process is not hungry we will still alternate the access leading to obvious inefficiency. This round-robin nature of our protocol can in fact be captured as a temporal property as follows.

   ```
   G( ( p1  ==>  ( p1  U  ( not p1 /\ (not p1 U p2) ) ) )    /\
      ( p2  ==>  ( p2  U  ( not p2 /\ (not p2 U p1) ) ) )
    )
   ```

   In the above, p1 is (turn == P) and p2 is (turn == C).