Some Problems to be discussed/solved in CS4271 last class

Abhik Roychoudhury
abhik@comp.nus.edu.sg
Dated 17 April 2009

1. Consider a traffic light controller which initially shows green light. It senses traffic data every second. Thus, starting from time=0, the traffic is sensed at time = 1 sec, 2 sec, and so on. If there is no traffic movement, the light turns from green to yellow. If there is traffic movement, the light stays green, but it can stay green for a maximum of 3 seconds at a stretch. The light always stays yellow for exactly one second after which it turns red. Once the light is red, again traffic is sensed every second. If there is traffic, then the light becomes green after staying red for a minimum of 2 seconds. If there is no traffic, the light eventually becomes green, after staying red for 3 seconds.

   Suppose you were trying to model the above controller as a Kripke structure. What aspects of the problem can you model and what aspects you cannot ? In particular, can you model the precise timing information *e.g.* traffic data being sensed every second.

2. Model the traffic light controller as a Kripke Structure. Try to keep your model as close as possible to the informal description. Clearly state the set of atomic propositions used.

3. We want to verify that the controller will never reach a state from where it is possible to stay red/yellow forever. Let us call it the **There-is-Hope** property. Specify this property in CTL. Use the explicit-state model checking algorithm discussed in class to show that your modeled Kripke Structure satisfies **There-is-Hope**. You need to show the main steps of the model checking computation.

4. Here is a generic bit of a shift register implemented in SMV.

```
MODULE cell(left, lval)
{

   content: boolean;

   init(content) := 0;

   next(content) := case{
       left  : lval;
       1     : content;
   };

}
```

Define the value of the bit in the "next" clock cycle shown above as `next(content)`, as a boolean function $F_{next}(left, lval, content)$. You should write the function as a propositional logic formula.

5. Give a variable ordering that produces the minimum sized Reduced Ordered Binary Decision Diagram for F_{next}. How can you exploit your understanding of the SMV program to construct this ROBDD directly without trying out all possible variable orders and the reduction steps for each of these variable orders ?

6. Suppose we construct the Kripke structure model M_{bit} corresponding to our shift register bit. The state variables of this model should be *left*, *lval*, *content* – that is, we capture the bit's behavior for all possible values of *left* and *lval*. Define the boolean function that captures the set of states of M_{bit} which satisfy the CTL formula $AG(left \wedge lval \Rightarrow AX content)$. You can use the truth table notation or you can write it as a propositional logic formula. Explain your answer.

7. In class, we described the recursive characterization of the **AU** operator of Computation Tree Logic (CTL) as follows ($g1$ and $g2$ are arbitrary CTL properties).

$$\mathbf{A}(g1\mathbf{U}g2) = g2 \vee (g1 \wedge \mathbf{AX}\ \mathbf{A}(g1\mathbf{U}g2))$$

Use the duality of the **AU** and **ER** operators in Computation Tree Logic (CTL) to get the recursive characterization of $\mathbf{E}\ (g1\ \mathbf{R}\ g2)$. Again, you may assume that $g1$ and $g2$ are arbitrary CTL properties.

8. Can you also get a recursive characterization of **A** ($g1$ **R** $g2$) in a similar fashion? Explain your answer.

9. Exploit your recursive characterization of $\mathbf{E}\ (g1\ \mathbf{R}\ g2)$ to develop an algorithm for checking whether a given *finite-state* Kripke Structure M satisfies $\mathbf{E}\ (g1\ \mathbf{R}\ g2)$. You may assume that $g1$ and $g2$ are arbitrary CTL properties. Also assume that the following are available to you — (a) Set of states in M satisfying $g1$, (b) Set of states in M satisfying $g2$.

10. Convert the explicit-state model checking algorithm in question A.2 into a *symbolic* model checking algorithm. Here the search should proceed by manipulation of Binary Decision Diagrams.

11. Suppose we want to construct a Reduced Ordered Binary Decision Diagram (ROBDD) for the following boolean function $func$ which has four input variables $x1, x2, x3, x4$.

$$func(x1, x2, x3, x4) = (x1 \Rightarrow (x2 \Rightarrow x3)) \wedge (\neg x1 \Rightarrow (x2 \Rightarrow x4))$$

Choose a variable ordering which results in as small a ROBDD as possible. Clearly state and justify your choice of variable ordering without actually constructing the ROBDDs for each variable ordering.

12. In class, we discussed shift registers and how their behaviors can be encoded as transition systems. Consider a 2-bit shift register which takes in two *boolean inputs* from the environment $in1$ and $in2$. Every clock cycle, if $in1 > in2$ (i.e. in1 = 1 and in2 = 0) then a right shift is performed with $in1$ going into the leftmost bit of the register; otherwise a left shift is performed with $in2$ going into the rightmost bit.

    - List all the variables which will determine the states of this register if we want to describe its behaviors via a finite state Kripke Structure.
    - For each of the two bits of the shift register try to define $next(bit)$ (i.e., the value of each bit in the next clock cycle) as a boolean function. Explain your answer. You should do this without going through the messy task of constructing the Kripke Structure of the shift register.

13. Construct a ROBDD to describe each of the boolean functions constructed in the previous question. You should always choose a variable order which results in as small a ROBDD as possible.

14. What is the difference (if any) between the following pair of Linear-time temporal logic formulae? If you think that the two formulae are equivalent you should give a formal proof. If you think they are not equivalent, you should construct an example Kripke Structure which satisfies one of them but not the other. You may assume that p, q are atomic propositions.

    p **U** q and (p **U** (**G** q))

15. What is the difference (if any) between the following pair of Linear-time temporal logic formulae? If you think that the two formulae are equivalent you should give a formal proof. If you think they are not equivalent, you should construct an example Kripke Structure which satisfies one of them but not the other. You may assume that p is an atomic proposition.

    $\neg\ \mathbf{FG}\ \mathrm{p}$ and $\mathbf{G}(\neg\ \mathrm{p} \vee \mathbf{X}\ \mathbf{F}\ \neg\ \mathrm{p})$