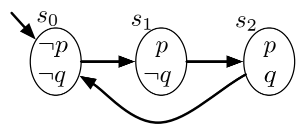
CSC2108: Automated Verification
Assignment 4. Part 1

Problem 1 is due on November 21. The rest are due in the last class.

1. Using the Bounded Model Checking technique, prove that the Kripke structure below satisfies the following properties



   (a) $\varphi_1 = E[EXpUq]$

   (b) $\varphi_2 = EG(q \Rightarrow p)$

   For each property φ_i ($i = 1, 2$), you need to only construct a propositional formula that is satisfiable iff φ_i holds over the above Kripke structure. You do not need to convert the resulting propositional formulas to CNF or perform the resolution proof.

2. Suppose we are given the following 4-state concrete model with variables x and y:

state	variables
s_0	x, y
s_1	$\neg x$, y
s_2	x, $\neg y$
s_3	$\neg x$, $\neg y$

   with the transition relation (s_0, s_0), (s_0, s_1), (s_1, s_0), (s_1, s_2), (s_2, s_0), (s_3, s_3), (s_0, s_3), (s_3, s_1).

   We define a 3-state abstraction of this system with the following *concretization* function:

$$\begin{aligned} \gamma(a_0) &= \{s_0, s_1\} \\ \gamma(a_1) &= \{s_1, s_2\} \\ \gamma(a_2) &= \{s_3\} \end{aligned}$$

   (a) Build this abstract system as a 3-valued Kripke structure.

   (b) Use the resulting structure to check the following properties:

      i. $AG(x)$

      ii. $AG(x \Rightarrow EFy)$

3. Give an example of two finite automata, which are not equivalent as automata on finite words, but are equivalent as Buchi automata.

4. Let a language that consists of all strings over the alphabet $\{a, b\}$ that end with a^ω or $(ab)^\omega$ be given. This language can be represented by a regular expression $(a + b)^*a^\omega + (a + b)^*(ab)^\omega$.

   (a) Give a parity automaton accepting this language.

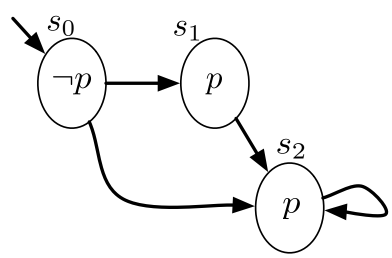
   (b) Now change its acceptance conditions to be:

      i. Buchi

      ii. Rabin

      iii. Muller

5. Let the following Kripke structure be given:



   Check whether the two LTL properties:

   - $p_1 = Fp$ and
   - $p_2 = Gp$

   hold on this Kripke structure. That is,

   (a) Convert each of the properties into the corresponding Buchi automata, named P_1 and P_2.

   (b) Convert the Kripke structure into an automaton K.

   (c) Computer intersection between K and P_1, and K and P_2.

   (d) Run emptiness decision algorithm on each intersection.

   (e) Draw conclusions.

6. Let the following SMV program be given:

```
MODULE main
VAR

s1:  {n1, t1, c1};
s2:  {n2, t2, c2};
d1 :  boolean;
```

```
d2 :  boolean;
turn: boolean;
r1 :  boolean;
r2 :  boolean;

ASSIGN

init(s1)   := n1;
init(s2)   := n2;
init(turn) := 0;
init(d1) := 0;
init(d2) := 0;

next(d1) :=
case
   r1 : 1;
   1 : {0, 1};
esac;
next(d2) :=
case
   r2 : 1;
   1 : {0, 1};
esac;

next(s1) :=
case
   (s1 = n1) : {n1, t1};
   (s1 = t1) & (s2 = n2): c1;
   (s1 = t1) & (s2 = t2) & (!turn):  c1;
   (s1 = c1): n1;
   1: s1;
esac;

next(s2) :=
case
   (s2 = n2) : {n2, t2};
   (s2 = t2) & (s1 = n1): c2;
   (s2 = t2) & (s1 = t1) & (turn):  c2;
   (s2 = c2): n2;
   1: s2;
esac;

next(turn) :=
case
```

```
   next(s1=c1) : 1;
   next(s2=c2) : 0;
   1: turn;
esac;

SPEC

EF((s1 = c1) & (s2 = c2))

SPEC

AG((s1 = t1) -> AF (s1 = c1))

SPEC

AG((s2 = t2) -> AF (s2 = c2))
```

Further, let variables `r1`, `r2`, `d1` and `d2` be abstracted away.

(a) Produce the resulting abstract SMV model.

(b) Check the properties on this model.

(c) Explain why you can trust the answer. Hint: Read Chapter 13.1 of the textbook.

7. Let the following program (written in some procedural language) be given:

```
x, y: integer;

x = 0; y = 4;
while (x <= 10) {
   x = x+y;
}
y--;
if (y == x) {
  while (x != 0) {
    x --;
  }
}
else {
  x = x*2;
}
```

Further, suppose we are interested in checking the property

$$P = AG(x\%2 == 0)$$

that is, x is always even.

(a) We now abstract variables x and y using the POS/NEG/ZERO abstraction. For operations of the above program, build abstract operations on the abstract domain. Make your answers as precise as possible.

(b) Build the abstract program.

(c) Our goal now is to check P on the abstract model. What is the abstract property that corresponds to P?

(d) Check (by hand) the value of P.

(e) Now, let's choose a better abstraction of x: EVEN/ODD. Build the abstract program under this abstraction.

(f) What is the abstract version of P under this abstraction?

(g) Check P.