Consider the following transition system \( T = (S, R, I) \) with the set of states \( S = \{0, 1, 2, 3\} \), the initial set of states \( I = \{0\} \), the transition relation \( R = \{(0, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3)\} \), the set of atomic propositions \( AP = \{p\} \), and the labeling function \( L : S \rightarrow 2^{AP} \) where \( L(0) = \emptyset \), \( L(1) = \emptyset \), \( L(2) = \emptyset \), \( L(3) = \{p\} \). We will use two boolean variables \( x, y \) to encode the states of this transition system as follows: \( \{0\} \equiv \neg x \land \neg y \), \( \{1\} \equiv x \land \neg y \), \( \{2\} \equiv \neg x \land y \), \( \{3\} \equiv x \land y \).

(a) Write the Boolean logic formulas that represent the transition relation \( R \) and the set of initial states \( I \) of the above transition system for this encoding.

(b) Compute \( \text{EX}(p) \) using the boolean encoding described above (show the steps of your computation).

(c) Given the variable ordering \( x < x' < y < y' \) draw the BDD for the transition relation \( R \).

2. Given the variable ordering \( x_1 < x_2 < x_3 \):

(a) Construct the BDDs for the formulas \((x_1 \land x_2) \lor x_3\) and \(\neg x_1 \land x_3\).

(b) Using the BDDs from part (a) show the recursive calls for the apply algorithm while computing the conjunction of the above two formulas. Show the resulting BDD.

3. In this problem you are going to write some C/C++ code to implement the basic functionality of a BDD-based symbolic model checker. You will use the CUDD BDD package (http://vlsi.colorado.edu/~fabio/CUDD/).

(a) Assuming that the transition relation \( R \) is represented as a BDD and property \( p \) is represented as a BDD, implement the EX function using the CUDD package.

(b) Assuming that the transition relation \( R \) is represented as a BDD and property \( p \) is represented as a BDD, using the EX function implementation from the previous part, implement the EF fixpoint computation using the CUDD package.

(c) Construct the BDDs for the transition relation \( R \) and the property \( p \) from Problem 1 and use the EF fixpoint computation you implemented to compute \( \text{EF}(p) \).

Output the BDDs that are constructed by the functions you implemented using CUDD’s output functions. Put your code and the outputs of the resulting BDDs in a directory (use
your last name as the name of the directory) and email the tar ball to the instructor.

Some of the CUDD functions that you will need for implementing this are: Cudd_bddAnd, Cudd_bddOr, Cudd_Not, Cudd_bddPermute, Cudd_bddAndAbstract, Cudd_Init, Cudd_RecursiveDeref, Cudd_Ref.

4. Consider Dekker’s mutual exclusion algorithm given below for two processes:

```plaintext
boolean a, b;
integer k;
initial: a = false and b = false and 1 <= k and k <= 2;

while true do
    a := true;
    while b do
        if (k = 2) then
            a := false;
            while (k = 2) do skip; endwhile;
            a := true;
        endif;
    endwhile;
    // critical section
    k := 2;
    a := false;
endwhile
||

while true do
    b := true;
    while a do
        if (k = 1) then
            b := false;
            while (k = 1) do skip; endwhile;
            b := true;
        endif;
    endwhile;
    // critical section
    k := 1;
    b := false;
endwhile
```

(a) Specify Dekker’s algorithm in SMV’s language.

(b) State the mutual exclusion property for your model in CTL and check if Dekker’s algorithms satisfies the mutual exclusion property using SMV. Provide a counter-example (with an explanation) if the property fails.

(c) State the starvation freedom property for your model in CTL and check if Dekker’s algorithms satisfies the mutual exclusion property. Provide a counter-example (with an explanation) if the property fails.

(d) Add the fairness constraint FAIRNESS running and check the starvation freedom property again. Provide a counter-example (with an explanation) if the property fails.