Solution to Exercises - Week 7
Intelligent Systems Programming

Exercise 1

A polynomial time algorithm for checking if a CNF formula is a tautologi is given below. Here the formula F is $F = C_1 \wedge \ldots \wedge C_m$, where $C_i = x_{i1} \vee x_{i2} \vee \ldots \vee x_{in_i}$. For F to be a tautologi, each clause C_i must evaluate to TRUE no matter how the variables are assigned. In each clause, a variable together with its negation therefore must be present.

```
IsCNFTautologi(F)
  for clause i ← 1 to m
    trueFound ← FALSE
    for j ← 0 to n_i
      for k ← 0 to n_i
        if x_ij = ¬x_ik
          trueFound ← TRUE
    if trueFound = FALSE
      return FALSE
  return TRUE
```

The running time of this algorithm is $O(m \cdot \max_{i \in \{1,\ldots,m\}} n_i^2) = O(m) \cdot O(n^2)$.

A polynomial time algorithm for checking if a DNF formula is a satisfiable is given below. Here the formula F is $F = C_1 \vee \ldots \vee C_m$, where $C_i = x_{i1} \wedge x_{i2} \wedge \ldots \wedge x_{in_i}$. For F to be satisfiable, we must be able to find a clause C_i where we can choose an assignment of the variables in the clause, such that it will evaluate to TRUE. This can be done, as long as we can find a clause where none of the variables are present in both negated and unnegated form.

IsDNFSatisfiable(F)
  for clause $i \leftarrow 1$ **to** m
    $clauseGood \leftarrow$ True
    for $j \leftarrow 0$ **to** n_i
      for $k \leftarrow 0$ **to** n_i
        if $x_{ij} = \neg x_{ik}$
          $clauseGood \leftarrow$ False
    if $clauseGood =$ True
      return True
  return False

The running time of this algorithm is $O(m \cdot \max_{i \in \{1,\ldots,m\}} n_i^2) = O(m) \cdot O(n^2)$.

Exercise 2

Given an algortihm Equivalence(F_1, F_2) that returns true iff F_1 and F_2 are equivalent, we can decide whether a formula F is a tautologi by:

$$\text{Tautologi}(F) = \text{Equivalence}(F, \top)$$

Similarly we can decide if a formula F is satisfiable by:

$$\text{Satisfiable}(F) = \neg\text{Equivalence}(F, \bot)$$

Here $\top$ denotes the tautologi (the proposition that is always true) and $\bot$ denotes the proposition that is always false.

Exercise 3

Using the if-then-else operators, we get:

$$\neg x = x \rightarrow 0, 1.$$

$$x \wedge y = x \rightarrow (y \rightarrow 1, 0), 0.$$

The corresponding ROBBDs are shown in Figure 1.

$$x \vee y = x \rightarrow 1, (y \rightarrow 1, 0).$$

$$x \Rightarrow y = x \rightarrow (y \rightarrow 1, 0), 1.$$

The corresponding ROBBDs are shown in Figure 2.

$$x \Leftrightarrow y = x \rightarrow (y \rightarrow 1, 0), (y \rightarrow 0, 1).$$

The ROBBD are shown in Figure 3.

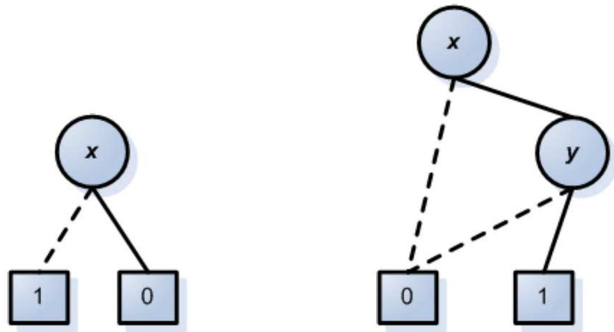

Figure 1: The ROBDDs for $\neg x$ and $x \wedge y$ respectively

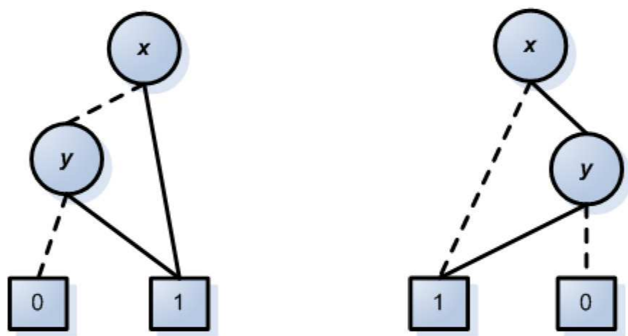

Figure 2: The ROBDDs for $x \vee y$ and $x \Rightarrow y$ respectively

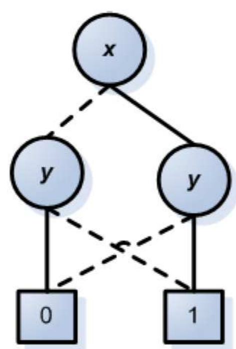

Figure 3: The ROBDD for $x \Leftrightarrow y$

Exercise 4

The ROBBD for $(x_1 \wedge y_1) \vee (x_2 \wedge y_2)$ using the ordering x_1, x_2, y_1, y_2 is shown in Figure 4.

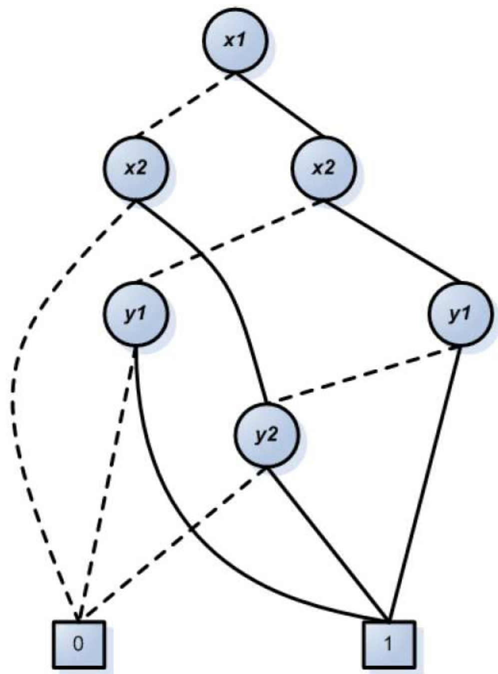

Figure 4: The ROBDD for $(x_1 \wedge y_1) \vee (x_2 \wedge y_2)$

Exercise 5

The ROBDD for $(x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2) \wedge (x_3 \Leftrightarrow y_3)$ using the ordering $x_1, y_1, x_2, y_2, x_3, y_3$ and $x_1, x_2, x_3, y_1, y_2, y_3$ are shown in Figues 5 and 6 respectively.

Exercise 6

For every internal node we can choose a variable in n ways, a low child in $g + 2$ ways (including two terminal nodes), and a high child in $g + 2$ ways. Therefore, we can completely specify one node in at most $n(g + 2)^2$ ways. Since we have to do this for each of the g internal nodes, all the nodes are completely specified in at most $(n(g + 2)^2)^g = n^g \cdot (g + 2)^{2g}$ ways. This is an upper bound on the total number of ROBDDs.

To estimate a fraction of ROBDDs that have a polynomial size we limit g to a polynomial number $O(n^k)$. Now, it is easily seen that the fraction of polynomial-size ROBDDs goes to 0, as n approaches infinity:

$$\frac{n^{O(n^k)} \cdot O(n^k)^{O(n^k)}}{2^{2^n}} \to 0, \text{ when } n \to \infty$$

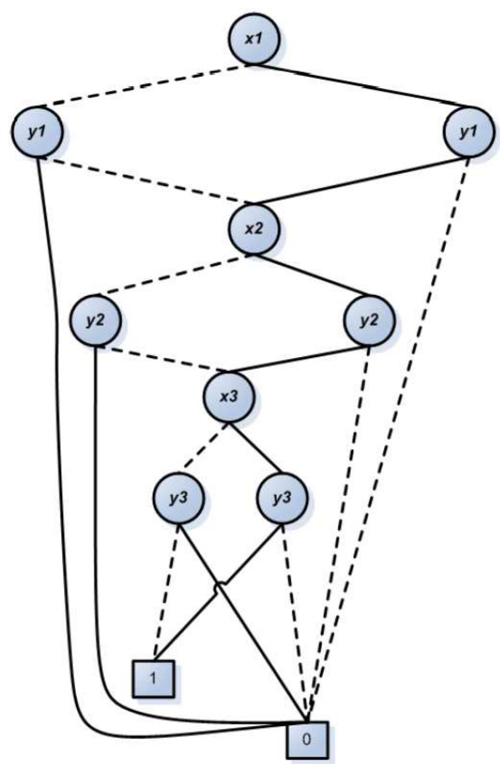

Figure 5: Using the ordering $x_1, y_1, x_2, y_2, x_3, y_3$

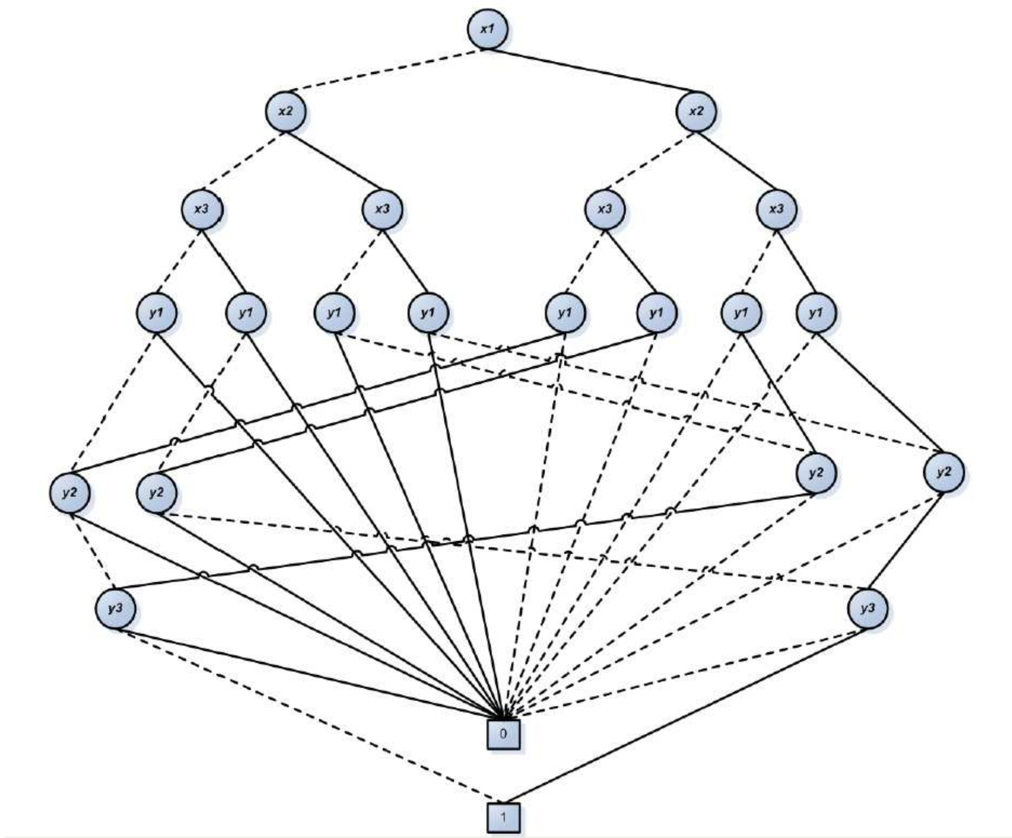

Figure 6: Using the ordering $x_1, x_2, x_3, y_1, y_2, y_3$