(BDD) A BDD (Binary Decision Diagram) is a binary expression that has one of the following 3 forms: \( T, \bot, \text{if variable then BDD else BDD} \). For example,

\[
\text{if } x \text{ then if } a \text{ then } T \text{ else } \bot \text{ else if } y \text{ then if } b \text{ then } T \text{ else } \bot \text{ else } \bot \text{ fi}
\]

is a BDD. An OBDD (Ordered BDD) is a BDD with an ordering on the variables, and in each if then else fi, the variable in the if-part must come before any of the variables in its then- and else-parts (“before” means according to the ordering). For example, using alphabetic ordering for the variables, the previous example is not an OBDD, but

\[
\text{if } a \text{ then if } c \text{ then } T \text{ else } \bot \text{ else if } b \text{ then if } c \text{ then } T \text{ else } \bot \text{ else } \bot \text{ fi}
\]

is an OBDD. An LBDD (Labeled BDD) is a set of definitions of the following 3 forms:

\[
\text{label } = T
\]
\[
\text{label } = \bot
\]
\[
\text{label } = \text{if variable then label else label fi}
\]

The labels are separate from the variables; each label used in a then-part or else-part must be defined by one of the definitions; exactly one label must be defined but unused. The following is an LBDD.

\[
\text{true } = T
\]
\[
\text{false } = \bot
\]
\[
\text{alice } = \text{if } b \text{ then true else false fi}
\]
\[
\text{bob } = \text{if } a \text{ then alice else false fi}
\]

An LOBDD is an LBDD that becomes an OBDD when the labels are expanded. The ordering prevents any recursive use of the labels. The previous example is an LOBDD. An RBDD (Reduced BDD) is a BDD such that, in each if then else fi, the then- and else-parts differ. An RLOBDD is reduced, labeled, and ordered. The previous example is an RLOBDD.

(a) Express \(-a, a \land b, a \lor b, a \Rightarrow b, a=b, a \lor b\), and if \(a\ then b\ else c\ fi\) as BDDs.

(b) How can you conjoin two OBDDs and get an OBDD?

(c) How can you determine if two RLOBDDs are equal?

(d) How can we represent an RLOBDD in order to determine efficiently if an assignment of values to variables satisfies it (solves it, gives it value \(T\))?