Binary Decision Diagrams
Binary Decision Diagrams (BDDs) [Bryant 86]

- Reduced Ordered Binary Decision Diagrams (BDDs)
  - An efficient data structure for representing Boolean functions (or truth sets of Boolean formulas) and manipulating them
  - There are BDD packages available: (for example CUDD from Colorado University)

- BDDs are a canonical representation for Boolean functions
  - given two Boolean logic formulas F and G, if F and G are equivalent (i.e. if their truth sets are the same), then their BDD representations will be the same
BDDs for Symbolic Model Checking

- BDD data structure can be used to implement the symbolic model checking algorithm we discussed earlier

- BDDs support all the operations we need for symbolic model checking
  - take conjunction of two BDDs
  - take disjunction of two BDDs
  - test equivalence of two BDDs
  - test subsumption between two BDDs
  - negate a BDD
  - test if a BDD satisfiable
  - test if a BDD is a tautology
  - existential variable elimination
Binary Decision Trees

Given a variable order, in each level of the tree, branch on the value of the variable in that level.

- Examples for boolean formulas on two variables
  Variable order: x, y

```
\begin{array}{cc}
\text{x } \lor \text{ y} & \text{x } \land \text{ y} \\
\begin{array}{cccc}
\text{F} & \text{T} & \text{F} & \text{T} \\
\text{T} & \text{F} & \text{T} & \text{T} \\
\end{array} & \\
\begin{array}{cccc}
\text{F} & \text{F} & \text{F} & \text{T} \\
\text{T} & \text{T} & \text{T} & \text{T} \\
\end{array} \\
\text{x } \land \text{ y} & \text{x } \lor \text{ y} \\
\begin{array}{cccc}
\text{F} & \text{T} & \text{F} & \text{T} \\
\text{T} & \text{F} & \text{T} & \text{T} \\
\end{array} & \\
\begin{array}{cccc}
\text{F} & \text{F} & \text{F} & \text{T} \\
\text{T} & \text{T} & \text{T} & \text{T} \\
\end{array} \\
\text{False} & \\
\begin{array}{cccc}
\text{F} & \text{T} & \text{F} & \text{T} \\
\text{T} & \text{T} & \text{T} & \text{T} \\
\end{array} & \\
\begin{array}{cccc}
\text{F} & \text{F} & \text{F} & \text{F} \\
\text{F} & \text{F} & \text{F} & \text{F} \\
\end{array}
\end{array}
```
Reduced and Ordered Binary Decision Diagrams

- We are interested in **Reduced** and **Ordered** Binary Decision Diagrams

- Reduced:
  - Merge all identical sub-trees in the binary decision tree (converts it to a directed-acyclic graph)
  - Remove redundant tests (if the false and true branches for a node go to the same place, remove that node)

- Ordered
  - We pick a fix order for the Boolean variables:
    \[ x_0 < x_1 < x_2 < \ldots \]
  - The nodes in the BDD are listed based on this ordering
BDDs

• Repeatedly apply the following transformations to a binary decision tree:

1. Remove duplicate terminals
2. Remove duplicate non-terminals
3. Remove redundant tests

• These transformations transform the tree to a directed acyclic graph
Binary Decision Trees vs. BDDs

$x \lor y$

$x \land y$

$x$

False
Good News About BDDs

• Given BDDs for two boolean logic formulas $F$ and $G$

  – The BDDs for $F \land G$ and $F \lor G$ are of size $|F| \times |G|$ (and can be computed in that time)

  – The BDD for $\neg F$ is of size $|F|$ (and can be computed in that time)

  – $F \equiv G$ can be checked in linear time

  – Satisfiability of $F$ can be checked in constant time
    • No, this does not mean that you can solve SAT in constant time
Bad News About BDDs

• The size of a BDD can be exponential in the number of boolean variables

• The sizes of the BDDs are very sensitive to the variable ordering. Bad variable ordering can cause exponential increase in the size of the BDD

• There are functions which have BDDs that are exponential for any variable ordering (for example binary multiplication)
BDDs are Sensitive to Variable Ordering

Identity relation for two variables: \((x' \leftrightarrow x) \land (y' \leftrightarrow y)\)

Variable order: \(x, x', y, y'\)

Variable order: \(x, y, x', y'\)

For \(n\) variables, \(3n+2\) nodes

For \(n\) variables, \(3 \times 2^n - 1\) nodes
BDDs from Another Perspective

• Any Boolean formula $f$ on variables $x_1, x_2, \ldots, x_n$ can be written as (called Shannon expansion):

$$f = x_i \land f[\text{True}/x_i] \lor \neg x_i \land f[\text{False}/x_i]$$

   (this is an if-then-else)

• BDDs use this idea

This node corresponds to the formula $y$, which comes from the Shannon expansion:

$$y \equiv x \land y[\text{True}/x]$$

This node corresponds to the formula False, which comes from the Shannon expansion:

$$\text{False} \equiv x \land y[\text{False}/x]$$
Model counting with BDDs

• Once you construct a BDD, you can count the number of models by counting paths of the BDD
• Count the paths that reach from the root to the “True” leaf node
• You need to take into account the variables that are not represented in the BDD
  – they are removed as redundant tests but we need to keep track of them to count
• Count the number of paths that reach True
  – keep track of missing (redundant) variables on a path, and add $2^k$ to the count for each path that has $k$ missing variables
• Can compute the count in linear time by traversing the nodes from leaves towards the root node