

***BDD, OBDD, ROBDD ...
OMG! WTF!?!?***

***a introduction to
Binary Decision Diagrams***

David “Dave” Dagon

dagon@cc.gatech.edu

College o’ Computing
Georgia Institute of Technology

- Motivation
- Review of Propositional Logic
- Normal Forms
- Binary Decision Diagrams
- Operations on BDDs

- What's the most commonly cited paper?
- Google this: “most cited paper on internet”
- What comes up?

Propositional Logic

- Propositional vs Categorical Logic
- Boolean expressions. Abstract syntax

$$t ::= x \mid 0 \mid 1 \mid \neg t \mid t \wedge t \mid t \vee t \mid t \Rightarrow t \mid t \Leftrightarrow t$$

Where x ranges over set of boolean vars, and t is a term

- Concrete syntax includes $(,)$.
- Associativity (*highest* \rightarrow *lowest*)

$$\neg, \wedge, \vee, \Leftrightarrow, \Rightarrow$$

A Note on Notation

- Example assignment: $[0/x_1, 1/x_2, 0/x_3, 1/x_4]$
Equivalent of

$$x_1 = 0 \quad x_2 = 1$$

$$x_3 = 0 \quad x_4 = 1$$

- (In)frequent Notation:
 - \perp false
 - \top true
 - A valuation on Φ is a function $v : \Phi \rightarrow \{0, 1\}$
 - For propositions $\phi \in \Phi$, we define
 $v \models \phi \Leftrightarrow v(\phi) = 1$

- Familiar truth tables

	\neg
0	1
1	0

\wedge	0	1
0	0	0
1	0	1

\vee	0	1
0	0	1
1	1	1

\Rightarrow	0	1
0	1	1
1	0	1

\Leftrightarrow	0	1
0	1	0
1	0	1

- The set of truth values often written as $\mathbb{B} = \{0, 1\}$
- All operators can be encoded using only \neg, \wedge .

- You are perhaps familiar with conjunctive and disjunctive normal forms (CNF, DNF).
- Example:

$$DNF : (x \wedge \neg y) \vee (\neg x \wedge y)$$

$$CNF : (x \vee \neg y) \wedge (\neg x \vee y)$$

- In CNF and DNF,
 - each term is either a variable or negated variable.
 - All terms joined by \wedge, \vee

Why Does Form Matter?

- Satisfiability of CNF boolean expressions is NP-complete (Cook).
- Satisfiability for DNF is poly-time (Consider short-circuiting!)
- Conversion from CNF to DNF is exponential in size.

DNF : n size

CNF : $n2^n$ size

- This sucks!

Binary Decision Diagrams

- Let's introduce a new operator, called "if-then-else"
- Defined as:

$$x \rightarrow y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- $t \rightarrow t_0, t_1$ is true if t and t_0 are true, or if t is false and t_1 is true.

Binary Decision Diagrams - Example

- By using the if-then-else operator, we can express *all* other operators.

<i>old</i>	<i>new</i>
$\neg x$	$(x \rightarrow 0, 1)$
$x \Leftrightarrow y$	$x \rightarrow (y \rightarrow 0, 1), (y \rightarrow 0, 1)$

If-then-else Normal Form

- An expression in If-then-else Normal Form (INF)
 - build entire of if-then-else operators, and
 - the numbers 0, 1
- Shannon expansion of t with respect to x :

$$t = x \rightarrow t[1/x], t[0/x]$$

- Key: Every boolean expression has an equivalent in INF.

Example Shannon Expansion

- Given $t = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$, the Shannon expansion is:

$$t = a_1 \rightarrow t_1, t_0$$

$$t_0 = b_1 \rightarrow 0, t_{00}$$

$$t_1 = b_1 \rightarrow t_{11}, 0$$

$$t_{00} = a_2 \rightarrow t_{001}, t_{000}$$

$$t_{11} = a_2 \rightarrow t_{111}, t_{110}$$

$$t_{000} = b_2 \rightarrow 0, 1$$

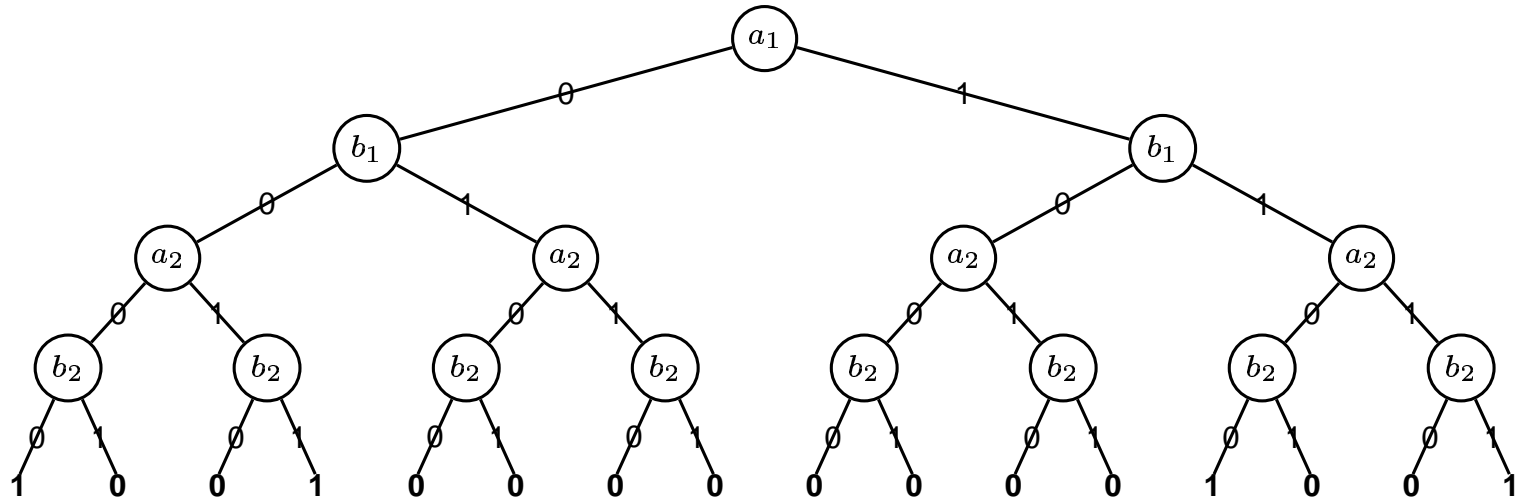
$$t_{001} = b_2 \rightarrow 1, 0$$

$$t_{110} = b_2 \rightarrow 0, 1$$

$$t_{111} = b_2 \rightarrow 1, 0$$

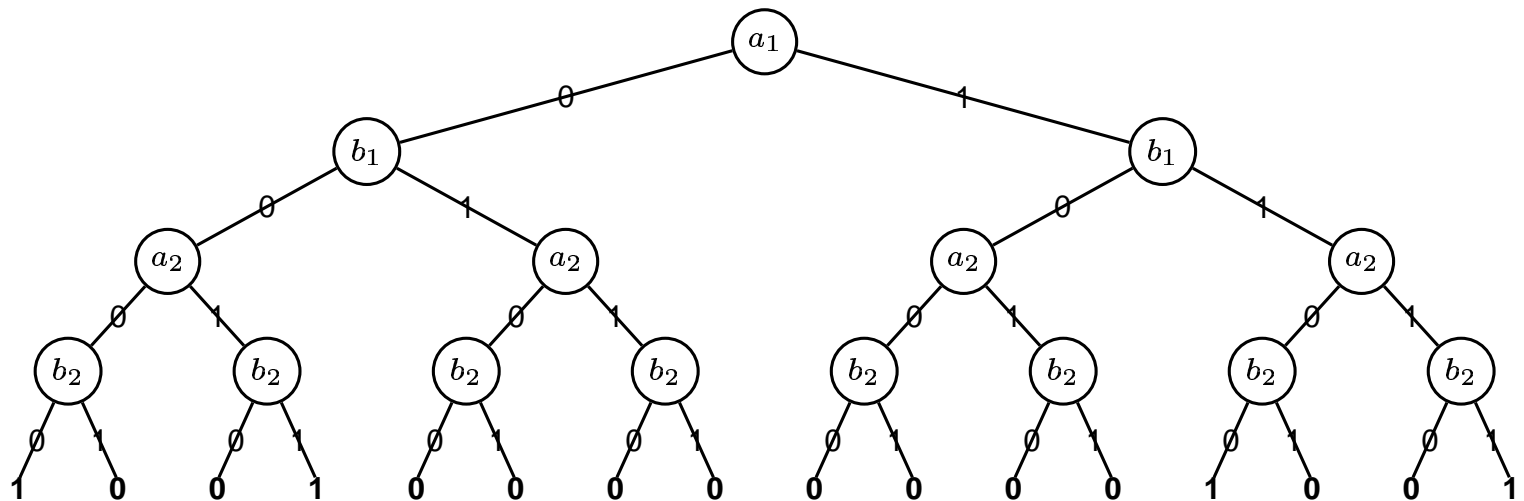
Decision Tree

- Expansions can also be shown as a tree.
- Given $t = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$



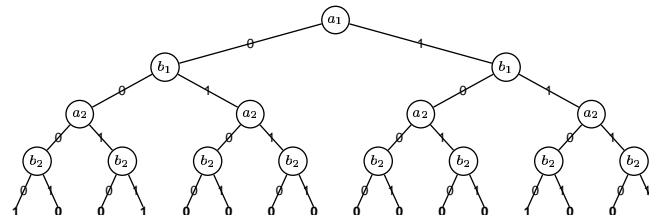
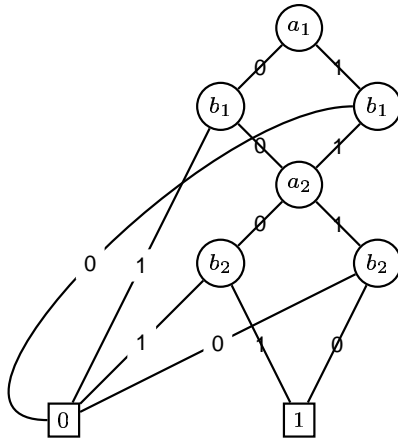
Analysis of BDD

- Fully expanded, *each* state is a separate leaf.
- The size of the tree equals a truth table ...
- ... but note the redundant states in
 $t = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$



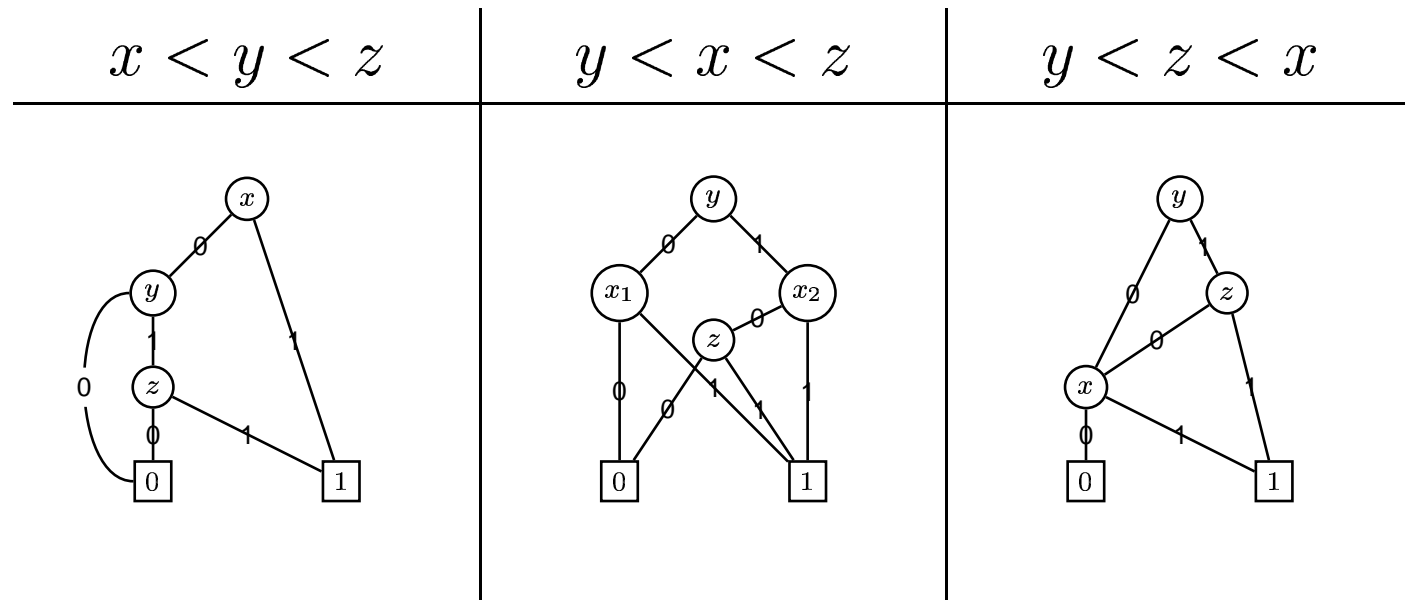
Assumptions

- Our expansion of $(a_1 \vee b_1) \wedge (a_2 \vee b_2)$ arbitrarily started with the root a_1 .
- The order $a_1 < b_1 < a_2 < b_2$ was important.
- By reducing (eliminating duplicate states) and ordering the variables, we've created an ROBDD.



The Importance of Ordering ORBDD

- The ordering of the variables determines the number of states
- Given: $x \vee (y \wedge z)$



- One can apply operations on BDDs

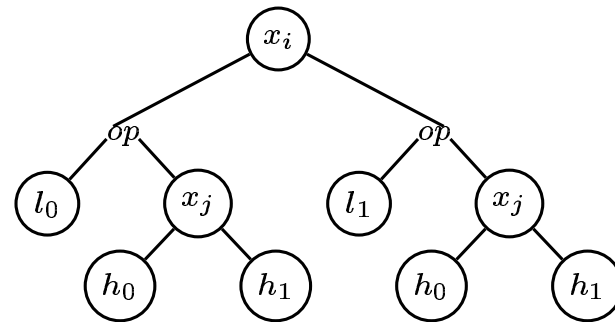
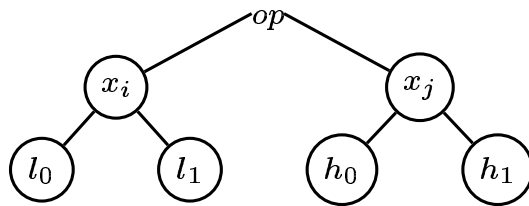
operation	notation	meaning
sum	$f + g$	$(f + g)(x) = f(x) + g(x)$
product	$f \cdot g$	$(f \cdot g)(x) = f(x) \cdot g(x)$
maximization	$\max(f, g)$	$(\max(f, g))(x) = \max(f(x), g(x))$

- One can apply operations on BDDs

operation	notation	meaning
sum	$f + g$	$(f + g)(x) = f(x) + g(x)$
product	$f \cdot g$	$(f \cdot g)(x) = f(x) \cdot g(x)$
maximization	$\max(f, g)$	$(\max(f, g))(x) = \max(f(x), g(x))$

Operations on BDDs

- More examples of function application to BDDs:



- Assumes $x_i < x_j$

Apply Operation

- Any binary argument can be applied to BDDs.
- The Apply function Implementation:
 - For terminal nodes with a dominant value, just return the terminal index
 - Use a hash table to avoid multiple recursive calls on identical vertices.
 - Generate a reduced graph through reductions

- Boolean expressions can take CNF, DNF, and INF forms, among others.
- INF BDDs are powerful because;
 - Common states may be reduced
 - Equality of BDDs checked in $O(1)$
 - Quantification is polynomial to operands:
 $O(\phi_1 \otimes \phi_2) = O(|\phi_1||\phi_2|)$
- Draw backs:
 - Quantification exponential to variables
 - Efficient ordering (to avoid state explosion) is an NP problem

- Useful resources:

- Randal Bryant, Graph-Based Algorithms for Boolean Function Manipulation
<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/bryant/www/pubdir/ieeetc86.ps>
(most cited paper on the internet!)
- JADE - A BDD Visualization Tool
http://www.informatik.uni-bremen.de/grp/ag-ram/public/software/index_d.html
- Jussi Rintanen, Lecture Notes on AI Planning (includes BDD)
<http://www.informatik.uni-freiburg.de/ki/lehre/ws0203/aip/>
(a rather complete set of notes)