Binary Decision Diagrams
Part 1

15-414 Bug Catching: Automated Program Verification and Testing

Sagar Chaki
September 12, 2011
BDDs in a nutshell

Typically mean Reduced Ordered Binary Decision Diagrams (ROBDDs)

Canonical representation of Boolean formulas

Often substantially more compact than a traditional normal form

Can be manipulated very efficiently
  • Conjunction, Disjunction, Negation, Existential Quantification

Running Example: Comparator

\[ f = 1 \iff a_1 = b_1 \land a_2 = b_2 \]
Conjunctive Normal Form

\[ f = (\neg a_1 \lor b_1) \land (\neg b_1 \lor a_1) \land (\neg a_2 \lor b_2) \land (\neg b_2 \lor a_2) \]

\[ \neg a_1 \Rightarrow b_1 \land b_1 \Rightarrow a_1 \land \neg a_2 \Rightarrow b_2 \land b_2 \Rightarrow a_2 \]

Not Canonical
Truth Table (1)

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$b_1$</th>
<th>$a_2$</th>
<th>$b_2$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Still Not Canonical
Truth Table (2)

<table>
<thead>
<tr>
<th>a&lt;sub&gt;1&lt;/sub&gt;</th>
<th>a&lt;sub&gt;2&lt;/sub&gt;</th>
<th>b&lt;sub&gt;1&lt;/sub&gt;</th>
<th>b&lt;sub&gt;2&lt;/sub&gt;</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Canonical if you fix variable order.

But always exponential in # of variables. Let’s try to fix this.
Representing a Truth Table using a Graph

Binary Decision Tree (in this case ordered)
Binary Decision Tree: Formal Definition

Balanced binary tree. Length of each path = # of variables

Leaf nodes labeled with either 0 or 1

Internal node $v$ labeled with a Boolean variable $\text{var}(v)$
  • Every node on a path labeled with a different variable

Internal node $v$ has two children: $\text{low}(v)$ and $\text{high}(v)$

Each path corresponds to a (partial) truth assignment to variables
  • Assign 0 to $\text{var}(v)$ if $\text{low}(v)$ is in the path, and 1 if $\text{high}(v)$ is in the path

Value of a leaf is determined by:
  • Constructing the truth assignment for the path leading to it from the root
  • Looking up the truth table with this truth assignment
Binary Decision Tree

\[ \text{var}(v) = a_1 \]

\[ \text{low}(v) \]

\[ \text{high}(v) \]
The truth assignment corresponding to the path to this leaf is:

\[ a_1 = ? \quad b_1 = ? \quad a_2 = ? \quad b_2 = ? \]
The truth assignment corresponding to the path to this leaf is:

\[ a_1 = 0 \quad b_1 = 0 \quad a_2 = 1 \quad b_2 = 0 \]
The truth assignment corresponding to the path to this leaf is:
\[ a_1 = 0 \quad b_1 = 0 \quad a_2 = 1 \quad b_2 = 0 \]

### Truth Table

<table>
<thead>
<tr>
<th>(a_1)</th>
<th>(b_1)</th>
<th>(a_2)</th>
<th>(b_2)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The truth assignment corresponding to the path to this leaf is:

\[ a_1 = 0 \quad b_1 = 0 \quad a_2 = 1 \quad b_2 = 0 \]
Binary Decision Tree (BDT)

Canonical if you fix variable order (i.e., use ordered BDT)

But still exponential in # of variables. Let’s try to fix this.
Reduced Ordered BDD

Conceptually, a ROBDD is obtained from an ordered BDT (OBDT) by eliminating redundant sub-diagrams and nodes.

Start with OBDT and repeatedly apply the following two operations as long as possible:

1. Eliminate duplicate sub-diagrams. Keep a single copy. Redirect edges into the eliminated duplicates into this single copy.
2. Eliminate redundant nodes. Whenever low(v) = high(v), remove v and redirect edges into v to low(v).

• Why does this terminate?

ROBDD is often exponentially smaller than the corresponding OBDT.
OBDT to ROBDD
OBDD to ROBDD

Duplicate sub-diagram
OBDDT to ROBDD
OBDD to ROBDD

Diagram showing the conversion of OBDD to ROBDD.
OBDD to ROBDD
OBDD to ROBDD
OBDT to ROBDD
OBDT to ROBDD
OBDT to ROBDD

Redundant node
OBDT to ROBDD
OBDT to ROBDD

Binary Decision Diagrams – Part 1
Sagar Chaki, Sep 12, 2011
© 2011 Carnegie Mellon University
OBDT to ROBDD
If $a_1 = 0$ and $b_1 = 1$ then $f = 0$ irrespective of the values of $a_2$ and $b_2$. 
OBDT to ROBDD
OBDT to ROBDD
OBDD to ROBDD
OBDT to ROBDD
OBDT to ROBDD
OBDT to ROBDD
Let’s move things around a little bit so that the BDD looks nicer.
Bryant gave a linear-time algorithm (called Reduce) to convert OBDT to ROBDD.

In practice, BDD packages don’t use Reduce directly. They apply the two reductions on-the-fly as new BDDs are constructed from existing ones. Why?
ROBDD (a.k.a. BDD) Summary

BDDs are canonical representations of Boolean formulas

• $f_1 = f_2 \iff ?$
ROBDD (a.k.a. BDD) Summary

BDDs are canonical representations of Boolean formulas

1. \( f_1 = f_2 \iff \text{BDD}(f_1) \text{ and BDD}(f_2) \text{ are isomorphic} \)
2. \( f \text{ is unsatisfiable} \iff \) ?
ROBDD (a.k.a. BDD) Summary

BDDs are canonical representations of Boolean formulas

- \( f_1 = f_2 \iff \text{BDD}(f_1) \text{ and } \text{BDD}(f_2) \text{ are isomorphic} \)
- \( f \) is unsatisfiable \( \iff \) BDD\((f)\) is the leaf node “0”
- \( f \) is valid \( \iff \) ?
ROBDD (a.k.a. BDD) Summary

BDDs are canonical representations of Boolean formulas
• $f_1 = f_2 \iff \text{BDD}(f_1)$ and $\text{BDD}(f_2)$ are isomorphic
• $f$ is unsatisfiable $\iff$ $\text{BDD}(f)$ is the leaf node “0”
• $f$ is valid $\iff$ $\text{BDD}(f)$ is the leaf node “1”
• BDD packages do these operations in constant time

Logical operations can be performed efficiently on BDDs
• Polynomial in argument size
• More details in next lecture

BDD size depends critically on the variable ordering
• Some formulas have exponentially large sizes for all ordering
• Others are polynomial for some ordering and exponential for others
ROBDD and variable ordering
ROBDD and variable ordering
ROBDD and variable ordering
ROBDD and variable ordering
ROBDD and variable ordering
ROBDD and variable ordering
ROBDD and variable ordering

Let’s move things around a little bit so that the BDD looks nicer.
ROBDD and variable ordering

a_1 < b_1 < a_2 < b_2

8 nodes

b_1

a_1

b_1

a_2

b_2

b_2

0

1

a_1 < a_2 < b_1 < b_2

11 nodes

b_1

a_2

b_1

b_1

b_1

b_2

b_2

1

0
ROBDD and variable ordering

\[ a_1 < b_1 < \ldots < a_n < b_n \]

\[ a_1 < \ldots < a_n < b_1 < \ldots < b_n \]
ROBDD and variable ordering

\[ a_1 \leq b_1 \leq \ldots \leq a_n \leq b_n \]

\[ 3 \times n + 2 \text{ nodes} \]

\[ a_1 \leq \ldots \leq a_n \leq b_1 \leq \ldots \leq b_n \]

\[ 3 \times 2^n - 1 \text{ nodes} \]
Next Class

BDD recap

BDD operations

BDD applications

Next homework

See you then …
Questions?

Sagar Chaki
Senior Member of Technical Staff
RTSS Program
Telephone: +1 412-268-1436
Email: chaki@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu/staff/chaki

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257