Binary Decision Diagrams

- Representation of Boolean Functions
- BDDs, OBDDs, ROBDDs
- Operations
- Model-Checking over BDDs

Boolean Functions

Boolean functions: $\mathcal{B} = \{0, 1\}$,
$$f : \mathcal{B} \times \cdots \times \mathcal{B} \to \mathcal{B}$$

Boolean expressions:
$$t ::= x | 0 | 1 | \neg t | t \land t | t \lor t | t \to t | \leftarrow t$$

Truth assignments: $\rho$,
$$[v_1/x_1, v_2/x_2, \cdots, v_n/x_n]$$

Satisfiable: Exists $\rho$ s.t. $t[\rho] = 1$

Tautology: For all $\rho$, $t[\rho] = 1$
What is a good representation of boolean functions?

Perfect representation is hopeless:

**Theorem 1** (Cook’s Theorem)
Satisfiability of Boolean expressions is NP-complete.

(Tautology-checking is co-NP-complete)

Good representations are
- compact and
- efficient
on real-life examples

### Truth Tables

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$x \rightarrow y, z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$2^n$ entries
Shannon Expansion

Def: \( x \rightarrow y_0, y_1 = (x \land y_0) \lor (\neg x \land y_1) \)

\( x \) is the test expression and thus this is an if-then-else.

We can represent all operators using if-then-else on unnegated variables and constants \( 0(\text{false}) \) and \( 1(\text{true}) \). This is called INF.

Shannon expansion w.r.t. \( x \):

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

Any boolean expression is equivalent to an expression in INF.

Combinatorial circuits

Are these equivalent? Do they represent a tautology? Are they satisfiable?
Example

\[ t = (x_1 \leftrightarrow y_1) \land (x_2 \leftrightarrow y_2) \]. Represent this in INF form with order \( x_1, y_1, x_2, y_2 \).

\[ t = x_1 \rightarrow t_{11}, t_0 \]
\[ t_0 = y_1 \rightarrow 0, t_{00} \quad \text{(since } x_1 = 1, y_1 = 0 \rightarrow t = 0) \]
\[ t_1 = y_1 \rightarrow t_{11}, 0 \quad \text{(since } x_1 = 0, y_1 = 1 \rightarrow t = 0) \]
\[ t_{00} = x_2 \rightarrow t_{001}, t_{000} \]
\[ t_{11} = x_2 \rightarrow t_{111}, t_{000} \]
\[ t_{000} = y_2 \rightarrow 0, 1 \quad (x_1 = 0, y_1 = 0, x_2 = 0) \]
\[ t_{001} = y_2 \rightarrow 1, 0 \quad (x_1 = 0, y_1 = 0, x_2 = 1) \]
\[ t_{110} = y_2 \rightarrow 0, 1 \quad (x_1 = 1, y_1 = 1, x_2 = 0) \]
\[ t_{111} = y_2 \rightarrow 1, 0 \quad (x_1 = 1, y_1 = 1, x_2 = 1) \]

Lots of common subexpressions:
- identify them!

BDDs – directed acyclic graph of Boolean expressions. If the variables occur in the same ordering on all paths from root to leaves, we call this OBDD.
**Example OBDD**

OBDD for \((x_1 \iff y_1) \land (x_2 \iff y_2)\) with ordering \(x_1 < y_1 < x_2 < y_2\)

If an OBDD does not contain any redundant tests, it is called ROBDD.

---

**ROBDDs**

A *Binary Decision Diagram* is a rooted, directed, acyclic graph \((V, E)\). \(V\) contains (up to) two terminal vertices, \(0, 1 \in V\). \(v \in V \setminus \{0, 1\}\) are non-terminal and have attributes \(\text{var}(v)\), and \(\text{low}(v), \text{high}(v) \in V\).

A BDD is *ordered* if on all paths from the root the variables respect a given total order.

A BDD is *reduced* if for all non-terminal vertices \(u, v\),

1) \(\text{low}(u) \neq \text{high}(u)\)

2) \(\text{low}(u) = \text{low}(v), \text{high}(u) = \text{high}(v), \text{var}(u) = \text{var}(v)\) implies \(u = v\).
**Canonicity of ROBDDs**

**Lemma 1 (Canonicity lemma)** For any function \( f : \mathcal{B}^n \rightarrow \mathcal{B} \) there is exactly one ROBDD \( b \) with variables \( x_1 < x_2 < \cdots < x_n \) such that

\[
r_b[v_1/x_1, \cdots, v_n/x_n] = f(v_1, \cdots, v_n)
\]

for all \( (v_1, \ldots, v_n) \in \mathcal{B}^n \).

**Consequences:**

- \( b \) is a tautology if and only if \( b = 1 \)
- \( b \) is satisfiable if and only if \( b \neq 0 \)
But...

The size of ROBDD depends significantly on the chosen variable ordering!

Example: ROBDD for \((x_1 \Leftrightarrow y_1) \land (x_2 \Leftrightarrow y_2)\) with ordering \(x_1 < x_2 < y_1 < y_2\)

![ROBDD Diagram](image.png)

Under ordering \(x_1 < y_1 < x_2 < y_2\) had 6 nodes.

Furthermore...

- The size according to one ordering may be exponentially smaller than another ordering.
- Figuring out the optimal ordering of variables is co-NP-complete.
- Some functions have small size independent of ordering, e.g. parity.
- Some functions have large size independent of ordering, e.g., multiplication
Implementing BDDs

\{ \text{root: integer; var, low, high: array of integer;} \}

\[ \begin{array}{|c|c|c|}
\hline
\text{var} & \text{low} & \text{high} \\
\hline
0 & ? & ? & ? \\
1 & ? & ? & ? \\
2 & 4 & 1 & 0 \\
3 & 4 & 0 & 1 \\
4 & 3 & 2 & 3 \\
5 & 2 & 4 & 0 \\
6 & 2 & 0 & 4 \\
7 & 1 & 5 & 6 \\
\hline
\end{array} \]

Helper Functions: Makenode and Hashing

Makenode ensures reducedness using a hash table

\[ H : (i, l, h) \rightarrow u \]

Makenode\((H, \text{max}, b, i, l, h)\)

1: \text{if } l = h \text{ then return } l
2: \quad \text{else if } \text{member}(H, i, l, h)
3: \quad \text{then return } \text{lookup}(H, i, l, h)
4: \quad \text{else } \text{max} \leftarrow \text{max} + 1
5: \quad b.\text{var}(\text{max}) \leftarrow i
6: \quad b.\text{low}(\text{max}) \leftarrow l
7: \quad b.\text{high}(\text{max}) \leftarrow h
8: \quad \text{insert}(H, i, l, h, \text{max})
9: \quad \text{return } \text{max}
Build

Build: Maps a Boolean expression $t$ into an ROBDD.

function $Build(t)$
1: \[ H \leftarrow \text{emptytable}; \max \leftarrow 1 \]
2: \[ b.\text{root} \leftarrow build'(t, 1) \]
3: \[ \text{return } b \]

function $build'(t, i)$
1: \[ \text{if } i > n \text{ then} \]
2: \[ \text{if } t \text{ is false then return } 0 \]
3: \[ \text{else return } 1 \]
4: \[ \text{else } l \leftarrow build'(t[0/x_i], i + 1) \]
5: \[ h \leftarrow build'(t[1/x_i], i + 1) \]
6: \[ \text{return } \text{makenode}(H, \max, b, i, l, h) \]

Build Example

$(x_1 \leftrightarrow x_2, 1)$

$(0 \leftrightarrow x_2, 2)$

$(1 \leftrightarrow x_2, 2)$

$(0 \leftrightarrow 0, 3)$

$(0 \leftrightarrow 1, 3)$

$(1 \leftrightarrow 0, 3)$

$(1 \leftrightarrow 1, 3)$

$x_1$

$x_2$

$x_2$

$1$

$0$

$0$

$1$
Boolean Operations on ROBDDs

Ordering: $x_1 < \cdots < x_n$

$(x_i \rightarrow l_1, l_0) \ op (x_i \rightarrow h_1, h_0) = x_i \rightarrow (l_1 \ op h_1), (l_0 \ op h_0)$

If $x_i < x_j$:

$(x_i \rightarrow l_1, l_0) \ op (x_j \rightarrow h_1, h_0) =

x_i \rightarrow (l_1 \ op (x_j \rightarrow h_1, h_0)), (l_0 \ op (x_j \rightarrow h_1, h_0))$
Other Operations on ROBDDs

**Restrict** – \( b[v/x] \)
given a truth assignment for \( x \), compute ROBDD for \( b \)

**Size** – \( \text{size}(b) = |\{ p \mid b[p] = 1 \}| \)
"number of valid truth assignments"

**Anysat** – \( \text{anysat}(b) = \rho \), for some \( \rho \) with \( b[\rho] = 1 \)
"give a satisfying assignment"

**Compose** – \( \text{compose}(b, x, b') = b[x/b'] \)
"substitute \( b' \) for all free occurrences of \( x \)"

**Existential quantification** – \( \exists x. b = b[x/0] \lor b[x/1] \)

Using dynamic hash-table implementation, can get amortized cost for operations to be \( O(1) \).
Uses of ROBDDs

Symbolic reasoning about:

- Combinatorial circuits
- Sequential circuits
- Automata
- Program analysis (theorem-proving)

and

- Temporal logic model checking