An Introduction to Binary Decision Diagrams

Henrik Reif Andersen

Department of Information Technology
Technical University of Denmark
Lyngby
http://www.it.dtu.dk/~hra

(Cand.scient. DAIMI 1990,
Ph.D. DAIMI 1993,
Adjunkt DTU 1993,
Lektor DTU 1996)

Boolean Functions

Boolean functions: $\mathbb{B} = \{0, 1\}$,

$$f : \mathbb{B} \times \cdots \times \mathbb{B} \to \mathbb{B}$$

Boolean expressions:

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

Truth assignments: ρ,

$$[v_1/x_1, v_2/x_2, \ldots, v_n/x_n]$$

Satisfiable: Exists ρ such that $t[\rho] = 1$

Tautology: Forall ρ, $t[\rho] = 1$

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

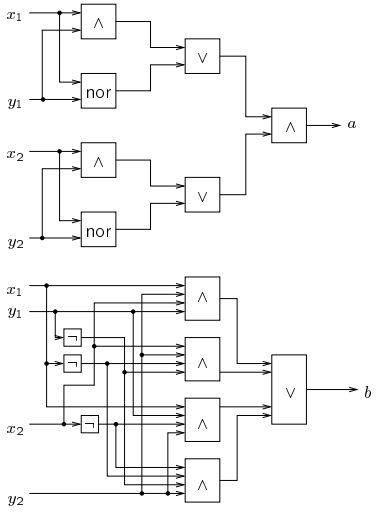
$\Leftrightarrow$	0	1
0	1	0
1	0	1

xyz	$x \to y, z$
000	0
001	1
010	0
011	1
100	0
101	0
110	1
111	1

$x_1 \cdots x_n$	$f(x_1, \ldots, x_n)$
$0 \cdots 0$	1
$0 \cdots 1$	0
$\vdots$	$\vdots$
$1 \cdots 1$	0

2^n entries

Combinatorial circuits



Equivalent?

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 has attributes *var*(v), and *low*(v), *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) *low*$(u) \neq$ *high*(u)
2) *low*$(u) =$ *low*(v), *high*$(u) =$ *high*(v), *var*$(u) =$ *var*(v) implies $u = v$

What's a good representation of Boolean functions?

Perfect representations are 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.

Canonicity of ROBDDs

$t_0 = 0$
$t_1 = 1$
$t_u = x \rightarrow t_h, t_l$, if u is a node (x, l, h)

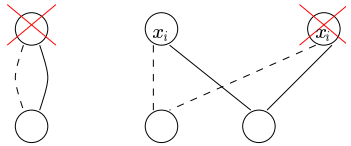
Lemma 1 (Canonicity lemma) *For any function $f : \mathbb{B}^n \rightarrow \mathbb{B}$ there is exactly one ROBDD b with variables $x_1 < x_2 < \ldots < x_n$ such that*

$$t_b[v_1/x_1, \ldots, v_n/x_n] = f(v_1, \ldots, v_n)$$

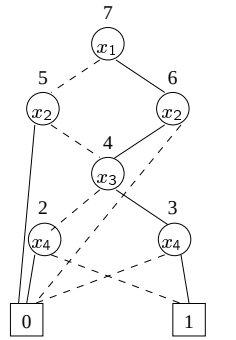
for all $(v_1, \ldots, v_n) \in \mathbb{B}^n$.

Consequences: b is a tautology, if and only if, $b = \boxed{1}$
b is satisfiable, if and only if, $b \neq \boxed{0}$

ROBDD Examples

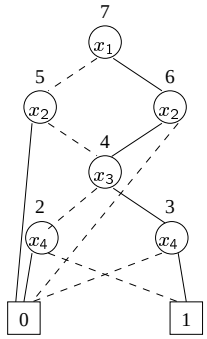


reducedness



$(x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4)$

Array Implementation



root = 7

	var	*low*	*high*
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

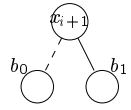
Canonicity Proof

Ordering: $x_n < \cdots < x_i < \cdots < x_1$.
Induction hypothesis
$P(i)$: All f over the variables $x_i, \ldots, x_1$ has a unique ROBDD

$P(0)$: $f = 0$ or $f = 1$. Only one ROBDD in both cases.

$P(i) \Rightarrow P(i+1)$: $f = x_{i+1} \rightarrow \underbrace{f[1/x_{i+1}]}_{b_1}, \underbrace{f[0/x_{i+1}]}_{b_0}$

$b_1 = b_0$: $f[1/x_{i+1}] = f[0/x_{i+1}] = f$. The unique ROBDD for f is b_1.

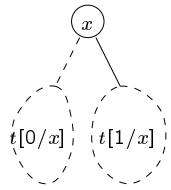
$b_1 \neq b_0$: 

unique due to ordering and reducedness

Shannon Expansion

Shannon expansion:

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



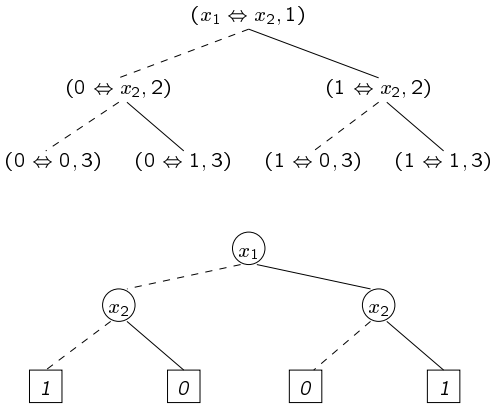
Makenode and Hashing

Makenode ensures reducedness using a hash table

$$H : (i, l, h) \mapsto u$$

Makenode(H, max, b, i, l, h)
1: **if** $l = h$ **then return** l
2: **else if** *member*(H, i, l, h) **then return** *lookup*(H, i, l, h)
3: **else** $max \leftarrow max + 1$
4: $b.var(max) \leftarrow i$
5: $b.low(max) \leftarrow l$
6: $b.high(max) \leftarrow h$
7: *insert*(H, i, l, h, max)
8: **return** *max*

Build Example



Build

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

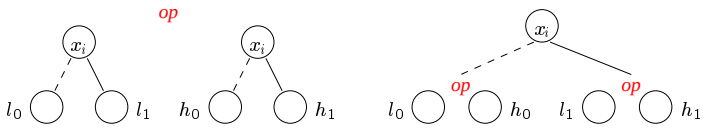
Build(t)

```
3:   function build'(t, i) =
4:        if i > n then if t is false then return 0 else return 1
5:        else l ← build'(t[0/x_i], i + 1)
6:             h ← build'(t[1/x_i], i + 1)
7:             return makenode(H, max, b, i, l, h)
8:   end build'
9:
1:   H ← emptytable    max ← 1
10:  b.root ← build'(t, 1)
11:  return b
```


Boolean Operations on ROBDDs

Ordering: $x_1 < \cdots < x_n$

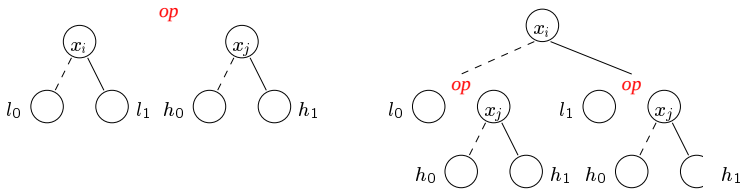
$$(x_i \to l_1, l_0)\ op\ (x_i \to h_1, h_0) = x_i \to (l_1\ op\ h_1), (l_0\ op\ h_0)$$



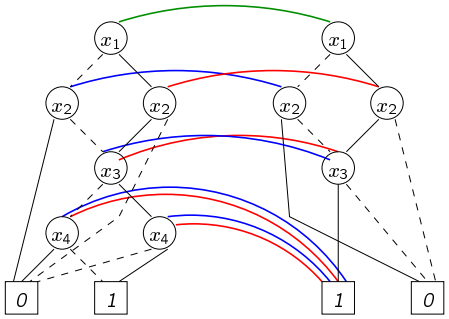
Boolean Operations on ROBDDs (cont'd)

If $x_i < x_j$:

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



Apply Example



Apply

```
Apply(op, b_1, b_2)
4:   function app(u_1, u_2) =
6:            if u_1 ∈ {0,1} and u_2 ∈ {0,1} then res ← op(u_1, u_2)
7:            else if u_1 ∈ {0,1} and u_2 ≥ 2 then
8:                 res ← makenode(var(u_2), app(u_1, low(u_2)), app(u_1, high(u_2)))
9:            else if u_1 ≥ 2 and u_2 ∈ {0,1} then
10:                res ← makenode(var(u_1), app(low(u_1), u_2), app(high(u_1), u_2))
11:           else if var(u_1) = var(u_2) then
12:                res ← makenode(var(u_1), app(low(u_1), low(u_2)),
                                             app(high(u_1), high(u_2)))
13:           else if var(u_1) < var(u_2) then
14:                res ← makenode(var(u_1), app(low(u_1), u_2), app(high(u_1), u_2))
15:           else (* var(u_1) > var(u_2) *)
16:                res ← makenode(var(u_2), app(u_1, low(u_2)), app(u_1, high(u_2)))
18:           return res
20:
21:  b.root ← app(b_1.root, b_2.root)
22:  return b
```


Other Operations on ROBDDs

Restrict	$b[v/x]$
Size (satcount)	$size(b) = \lvert\{\rho \mid b[\rho] = 1\}\rvert$
Anysat	$anysat(b) = \rho$, for some ρ with $b[\rho] = 1$
Allsat	$allsat(b) = \{\rho \mid b[\rho] = 1\}$
Compose	$compose(b, x, b') = b[x/b']$
Existential quantification	$\exists x.b = b[x/0] \vee b[x/1]$

Apply with Dynamic Programming

```
Apply(op, b_1, b_2)
4:   function app(u_1, u_2) =
5:        if G(u_1, u_2) ≠ empty then return G(u_1, u_2)
6:        else if u_1 ∈ {0,1} and u_2 ∈ {0,1} then res ← op(u_1, u_2)
7:             else if u_1 ∈ {0,1} and u_2 ≥ 2 then
8:                  res ← makenode(var(u_2), app(u_1, low(u_2)), app(u_1, high(u_2)))
9:             else if u_1 ≥ 2 and u_2 ∈ {0,1} then
10:                 res ← makenode(var(u_1), app(low(u_1), u_2), app(high(u_1), u_2))
11:            else if var(u_1) = var(u_2) then
12:                 res ← makenode(var(u_1), app(low(u_1), low(u_2)),
                                              app(high(u_1), high(u_2)))
13:            else if var(u_1) < var(u_2) then
14:                 res ← makenode(var(u_1), app(low(u_1), u_2), app(high(u_1), u_2))
15:            else (* var(u_1) > var(u_2) *)
16:                 res ← makenode(var(u_2), app(u_1, low(u_2)), app(u_1, high(u_2)))
17:            G(u_1, u_2) ← res
18:            return res
20:
2:   forall i ≤ max(b_1), j ≤ max(b_2) :  G(i, j) ← empty
21:  b.root ← app(b_1.root, b_2.root)
22:  return b
```

Further Reading Material

[Bryan'ro] Randal e. Bryan, Graphic Models for Reasoning in Computer Science, Carnegie Mellon University, 2013, 2014, 2015, 2016-2018

[Bernstein'ho] A.K. Bernstein, A.K. Chandra, A.L. Wendelin, D.L. Hill, L.J. Kenney, Computer Science Laboratory, 1979 Hardware Research, Lucent Tech, Inc. Murray

[Kaminski-Lagunas'ro] A.K. Kaminski, A. Lagunas, Computer Science Laboratory, Lucent Tech, Inc. Murray

> Introduction ...

Use of RODDDS

- Constraint Circuits
- Sensor Circuits
- Automata
- Constraint Simulation
- Temporal Disk Model Decision
- Program Analysis