COMP424/524-06A
Topics in Software Engineering

Part I – *Model Checking Algorithms*

15. Binary Decision Diagrams

Robi Malik

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato
DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Representing Boolean Functions

Formula:

$(a \vee c) \wedge (b \Rightarrow d)$

Normal forms:

$(a \vee c) \wedge (\neg b \vee d)$

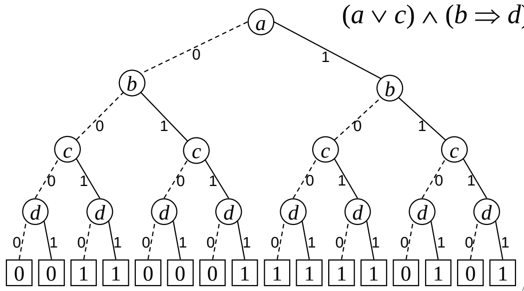
$(a \wedge \neg b) \vee (a \wedge d) \vee$
$(c \wedge \neg b) \vee (c \wedge d)$

Truth table:

a	b	c	d	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Binary Decision Tree

$(a \vee c) \wedge (b \Rightarrow d)$



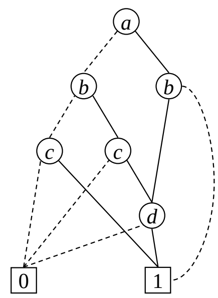
Reducing Decision Trees

Two ways of simplifying decision trees:

1. Identify and share identical subtrees.
2. Remove nodes whose left and right child nodes are identical.

Results in a Reduced Ordered Binary Decision Diagram (OBDD).

Ordered Binary Decision Diagram



$(a \vee c) \wedge (b \Rightarrow d)$

OBDD

Properties

Storage Efficiency

Many common Boolean functions have small OBDD representations.

Canonicity

If the order in which the variables are tested is fixed, then there exists only one OBDD for each Boolean formula.

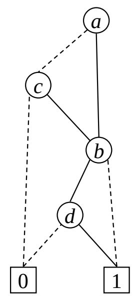
The Variable Ordering

On every branch in an OBDD, the variables must be tested in the same order, e.g.,

$$a < b < c < d$$

Different variable orderings yield different OBDDs.

Ordered Binary Decision Diagram



$$(a \vee c) \wedge (b \Rightarrow d)$$

$$a < c < b < d$$

Exercise

Find OBDDs for the Boolean formula

$$(x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2) \wedge (x_3 \Leftrightarrow y_3)$$

using two different variable orderings:

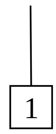
a) $x_1 < x_2 < x_3 < y_1 < y_2 < y_3$

b) $x_1 < y_1 < x_2 < y_2 < x_3 < y_3$

Tautology Checking Using OBDDs

To check whether a formula φ is a tautology, i.e., true for all truth assignments:

1. Compute the OBDD for φ.
2. Compare it to the OBDD representing true.



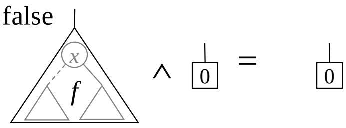
Operations on OBDDs

- **Comparison**
  Two OBDDs represent the same function if and only if they are identical.
- **Negation**
  To obtain an OBDD for ¬φ from an OBDD for φ, just swap its 0-leaf and its 1-leaf.
- **Conjunction**, **Disjunction**, etc.
  By recursive algorithms …

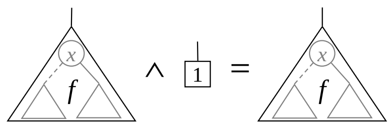
Conjunction of OBDDs (1)

Base cases:

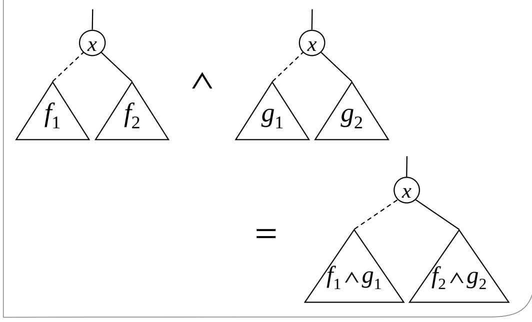
$f \wedge \text{false} = \text{false}$



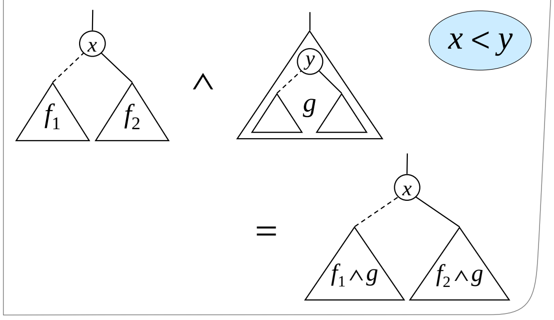
$f \wedge \text{true} = f$



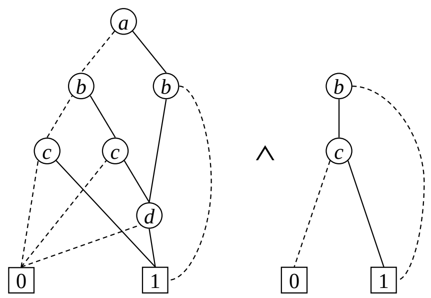
Conjunction of OBDDs (2)



Conjunction of OBDDs (3)



Exercise



Implementation Issues

Node hash table

Nodes are uniquely identified by
- variable number
- right and left subtrees

and stored in a hash table for quick look-up.

Caching

Store intermediate results of OBDD operations for some time, so repeated operations can be done quickly.

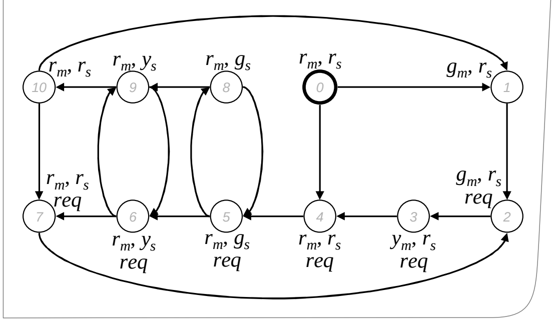
OBDD Packages

CUDD
`http://vlsi.colorado.edu/~fabio`

Buddy
`http://buddy.sourceforge.net`

JDD (pure Java)
`http://javaddlib.sourceforge.net`

Representing Automata with OBDDs



Representing States

	g_m	y_m	r_m	g_s	y_s	r_s	req
0	0	0	1	0	0	1	0
1	1	0	0	0	0	1	0
2	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1
4	0	0	1	0	0	1	1
5	0	0	1	1	0	0	1
6	0	0	1	0	1	0	1
7	0	0	1	0	0	1	1
8	0	0	1	1	0	0	0
9	0	0	1	0	1	0	0
10	0	0	1	0	0	1	0

2 g_m, r_s, req

$g_m \wedge \neg y_m \wedge \neg r_m \wedge \neg g_s \wedge \neg y_s \wedge r_s \wedge req$

Representing Transitions

r_m, r_s 0 $\rightarrow$ 4 r_m, r_s, req

Primed variables for next state

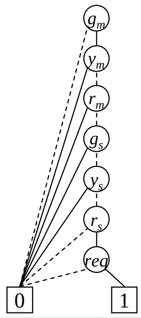
Current State								Next State							
	g_m	y_m	r_m	g_s	y_s	r_s	req		g'_m	y'_m	r'_m	g'_s	y'_s	r'_s	req'
0	0	0	1	0	0	1	0	1	1	0	0	0	0	1	0
0	0	0	1	0	0	1	0	4	0	0	1	0	0	1	1
1	1	0	0	0	0	1	0	2	1	0	0	0	0	1	1
2	1	0	0	0	0	1	1	3	0	1	0	0	0	1	1
3	0	1	0	0	0	1	1	4	0	0	1	0	0	1	1
…								…							

$\neg g_m \wedge \neg y_m \wedge r_m \wedge \neg g_s \wedge \neg y_s \wedge r_s \wedge \neg req \wedge \neg g'_m \wedge \neg y'_m \wedge r'_m \wedge \neg g'_s \wedge \neg y'_s \wedge r'_s \wedge req'$

Representing States

2 g_m, r_s, req

$g_m \wedge \neg y_m \wedge \neg r_m \wedge \neg g_s \wedge \neg y_s \wedge r_s \wedge req$



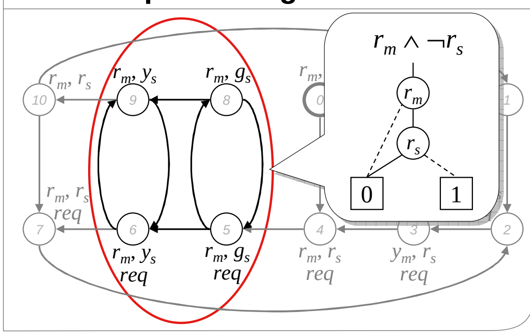
Representing NuSMV Code

```
MODULE switch(toggle)
VAR
  state:  {straight, curved}
ASSIGN
  init(state) := straight;
  next(state) := case
    !toggle:          state;
    state = straight: curved;
    state = curved:   straight;
  esac;
```

Translated into an OBDD with variables

- *toggle*
- *straight*
- *straight'*

Representing State Sets



Reading

Bérard et. al.:

Chapter 4 –
Symbolic Model Checking