CS4350: Logic in computer Science

Normal Forms

1

Logical Operators

Do we need all these?

- $\vee$ - Disjunction
- $\wedge$ - Conjunction
- $\neg$ - Negation
- $\rightarrow$ - Implication: $A \rightarrow B \equiv \neg A \vee B$
- $\oplus$ - Exclusive or: $A \oplus B \equiv (A \wedge \neg B) \vee (\neg A \wedge B)$
- $\leftrightarrow$ - Biconditional: $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$
- $\uparrow$ - Nand: $A \uparrow B \equiv \neg (A \wedge B)$
- $\downarrow$ - Nor: $A \downarrow B \equiv \neg (A \vee B)$

2

Logical Operators

- There are two nullary Boolean operators, 1, which is interpreted as true, and 0, which is interpreted as false.
- There are 4 unary operators:
  - $f_1(x) = 0$; $f_2(x) = 1$; $f_3(x) = x$; and $f_4(x) = \neg x$.
- There are 16 binary Boolean operators:
  - $\wedge$, $\vee$, $\rightarrow$, $\oplus$, $\leftrightarrow$, $\uparrow$, and $\downarrow$ are some of them.
- There are 64 trinary Boolean operators:
  - ite(x, y, z) = if x then y else z   is one of them
  - ite(x, y, z) $\equiv x \wedge y \vee \neg x \wedge z$
- How many n-ary Boolean operators? 2^{2^n}

3

Functionally Sufficient

- A set of logical operators is called (functionally) sufficient if every formula is logically equivalent to a formula involving only this set of logical operators.
- $\wedge$, $\vee$, and $\neg$ form a sufficient set of operators.
- Are there other sufficient sets?
- YES, because $A \wedge B \equiv \neg(\neg A \vee \neg B)$, we may drop $\wedge$ and leave $\vee$ and $\neg$ as one.

4

Functionally Sufficient

- { ∨, ¬ } is sufficient.
- { ∧, ¬ } is sufficient.
- { ite, 0, 1 } is sufficient:
  - ¬ *x* ≡ ite(x, 0, 1)
  - x ∧ y ≡ ite(x, ite(y, 1, 0), 0)
- ITE operator can implement any two variable logic function. There are 16 such functions corresponding to all subsets of B^2:

5

ITE Operator: $\text{ite}(f, g, h) = fg + \overline{f}h$

Output of o(x, y) on (0,0), (0,1), (1,0), and (11)

y = ite(y, 1, 0); $\overline{y}$ = ite(y, 0, 1)

Subset	Name	Expression	Equivalent Form
0000	constant 0	0	0
0001	AND(x, y)	x ∧ y	ite(x, y, 0)
0010	x > y	x ∧ ¬y	ite(x, $\overline{y}$, 0)
0011	1st projection	x	ite(x, 1, 0)
0100	x < y	¬ x ∧ y	ite(x, 0, y)
0101	2nd projection	y	ite(y, 1, 0)
0110	XOR(x, y)	x ⊕ y	ite(x, $\overline{y}$, y)
0111	OR(x, y)	x ∨ y	ite(x, 1, y)
1000	NOR(x, y)	x ↓ y	ite(x, 0, $\overline{y}$)
1001	EQ(x, y)	x ↔ y	ite(x, y, $\overline{y}$)
1010	NOT(y)	¬ y	ite(y, 0, 1)
1011	x ≥ y	x ∨ ¬ y	ite(x, 1, $\overline{y}$)
1100	NOT(x)	¬ x	ite(x, 0, 1)
1101	x ≤ y	¬ x ∨ y	ite(x, y, 1)
1110	NAND(x, y)	x ↑ y	ite(x, $\overline{y}$, 1)
1111	constant 1	1	1

6

6

Are $\neg(p \lor (\neg p \land q))$ and $(\neg p \land \neg q)$ equivalent?

$\neg(p \lor (\neg p \land q))$	
$\equiv \neg p \land \neg(\neg p \land q)$	DeMorgan
$\equiv \neg p \land (\neg\neg p \lor \neg q)$	DeMorgan
$\equiv \neg p \land (p \lor \neg q)$	Double Negation
$\equiv (\neg p \land p) \lor (\neg p \land \neg q)$	Distribution
$\equiv (p \land \neg p) \lor (\neg p \land \neg q)$	Commutative
$\equiv 0 \lor (\neg p \land \neg q)$	And Contradiction
$\equiv (\neg p \land \neg q)$	Identity

7

Normal Form

- So $\neg(p \lor (\neg p \land q))$ and $(\neg p \land \neg q)$ are equivalent, even though both are expressed with only $\land$, $\lor$, and $\neg$.
- It is still hard to tell without doing a proof.
- What we need is a standard format of a formula that uses a small set of operators.
- This unique representation is called a *Normal Form*.

8

Normal Forms

A restricted set of formulas such that other equivalent formulas can be converted to.

- NNF: Negation Normal Form
  - Use { ¬, ∨, ∧ }
- CNF: Conjunctive Normal Form
  - Use { ¬, ∨, ∧ }
- DNF: Disjunctive Normal Form
  - Use { ¬, ∨, ∧ }
- INF: ITE Normal Form
  - Use { ite, 0, 1}

9

Negation Normal Form

- A *literal* is either a propositional variable (*positive literal*) or the negation of a proposition variable (negative literal).
- A formal is in *negation normal form (NNF)* if the negation symbol appears only in literals.
- Example:
  - $\neg p \wedge \neg q$, $(\neg p \wedge \neg q) \vee r$ are in NNF.
  - $\neg(p \vee (\neg p \wedge q))$ is not in NNF.

10

Obtaining NNF

- Use the following relations to get rid of $\rightarrow$, $\oplus$, $\leftrightarrow$, $\uparrow$, and $\downarrow$:
  - $A \rightarrow B \equiv \neg A \vee B$
  - $A \oplus B \equiv (A \wedge \neg B) \vee (\neg A \wedge B)$
  - $A \leftrightarrow B \equiv (\neg A \vee B) \wedge (\neg B \vee A)$
  - $A \uparrow B \equiv \neg (A \wedge B)$
  - $A \downarrow B \equiv \neg (A \vee B)$
- Use De Morgan's Laws to push $\neg$ down:
  - $\neg (A \wedge B) \equiv \neg A \vee \neg B$
  - $\neg (A \vee B) \equiv \neg A \wedge \neg B$

11

Obtaining NNF

- Simplify formulas with the following axioms:
  - $p \vee 1 \equiv 1$
  - $p \wedge 0 \equiv 0$
  - $p \vee 0 \equiv p$
  - $p \wedge 1 \equiv p$
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
  - $\neg 0 \equiv 1$
  - $\neg 1 \equiv 0$
  - $\neg \neg p \equiv p$
  - $p \vee \neg p \equiv 1$
  - $p \wedge \neg p \equiv 0$
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

12

Conjunctive Normal Form (CNF)

- A disjunctions of literals is called a *clause*.
  - Duplicate literals are removed from a clause
- A CNF is a conjunction of disjunctions of literals (product of sums (POS))
  - Duplicate clauses are removed from CNF

Example:

- $\neg p \wedge \neg q$, $\neg p \vee \neg q \vee r$ are in CNF.
- $(p \wedge \neg q) \vee (\neg p \wedge q)$ is not in CNF.

13

Obtaining CNF

- At first, convert the formula in NNF.
- Use the distribution law to push $\vee$ down:
  - $X \vee (A \wedge B) \equiv (X \vee A) \wedge (X \vee B)$
  - $(A \wedge B) \vee X \equiv (X \vee A) \wedge (X \vee B)$
- Simplify formulas as we do for NNF

- **Theorem**: Every formula has an equivalent formula in CNF.

14

Disjunctive Normal Form (DNF)

- A conjunctions of literals is called a *minterm* (or *product*).
  - Duplicate literals are removed from a minterm
- A DNF is a disjunction of conjunctions of literals (sum of products (SOP))
  - Duplicate minterms are removed from DNF

Example:

- $\neg p \wedge \neg q$, $(p \wedge \neg q) \vee (\neg p \wedge q)$ are in DNF.
- $(\neg p \vee \neg q) \wedge r$ is not in DNF.

15

Obtaining DNF

- At first, convert the formula in NNF.
- Use the distribution law to push $\vee$ down:
  - $X \wedge (A \vee B) \equiv (X \wedge A) \vee (X \wedge B)$
  - $(A \vee B) \wedge X \equiv (X \wedge A) \vee (X \wedge B)$
- Simplify formulas as we do for NNF

- **Theorem**: Every formula has an equivalent formula in DNF.

16

Full CNF and Full DNF

- A CNF is a *full CNF* if every clause contains every variable exactly once.
  - If a clause C does not contain y, replace C by
    $(C \vee y) \wedge (C \vee \neg y)$
- A DNF is a *full DNF* if every minterm contains every variable exactly once.
  - If a midterm A does not contain y, replace A by
    $(A \wedge y) \vee (A \wedge \neg y)$

17

Normal Form vs Canonical Form

- A *canonical form* is a normal form which has a unique representation for all equivalent formulas.
- Advantage: Easy to tell equivalent formulas.
- To show A is valid, check if A's canonical form is 1.
- CNF and DNF are not canonical forms:
  $(p \vee r) \wedge (q \vee \neg r) \equiv (p \vee q) \wedge (p \vee r) \wedge (q \vee \neg r)$
  $(p \wedge r) \vee (q \wedge \neg r) \equiv (p \wedge q) \vee (p \wedge r) \vee (q \wedge \neg r)$
- Full CNF and Full DNF are canonical forms (up to the associativity and commutativity of $\wedge$ and $\vee$).

18

Get Full DNF & CNF from Truth Table

- Truth table is popular for defining Boolean functions.

a	b	c	out	minterms	clauses
0	**0**	**0**	**0**	$m_0=\bar{a}\,\bar{b}\,\bar{c}$	$M_0=a+b+c$
0	**0**	**1**	**1**	$m_1=\bar{a}\,\bar{b}\,c$	$M_1=a+b+\bar{c}$
0	**1**	**0**	**0**	$m_2=\bar{a}\,b\,\bar{c}$	$M_2=a+\bar{b}+c$
0	**1**	**1**	**1**	$m_3=\bar{a}\,b\,c$	$M_3=a+\bar{b}+\bar{c}$
1	**0**	**0**	**0**	$m_4=a\,\bar{b}\,\bar{c}$	$M_4=\bar{a}+b+c$
1	**0**	**1**	**1**	$m_5=a\,\bar{b}\,c$	$M_5=\bar{a}+b+\bar{c}$
1	**1**	**0**	**0**	$m_6=a\,b\,\bar{c}$	$M_6=\bar{a}+\bar{b}+c$
1	**1**	**1**	**1**	$m_7=a\,b\,c$	$M_7=\bar{a}+\bar{b}+\bar{c}$

- i in m_i and M_i is the decimal value of abc (in binary)
- $\neg\, m_i \equiv M_i$

- DNF $f_1 = \bar{a}\,\bar{b}\,c + \bar{a}\,b\,c + a\,\bar{b}\,c = m_1 + m_3 + m_5 + m_7$
- CNF $f_2 = (a+b+c)(a+\bar{b}+c)(\bar{a}+b+c)(\bar{a}+\bar{b}+c) = M_0M_2M_4M_6$

Get Full DNF & CNF from Truth Table

- DNF $f_1 = \bar{a}\,\bar{b}\,c + \bar{a}\,b\,c + a\,\bar{b}\,c = m_1 + m_3 + m_5 + m_7$
- CNF $f_2 = (a+b+c)(a+\bar{b}+c)(\bar{a}+b+c)(\bar{a}+\bar{b}+c) = M_0M_2M_4M_6$

- Do f_1 and f_2 define the same function?
- YES.
- $\neg f_1 = m_0 + m_2 + m_4 + m_6$
  - (0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0) are models of $\neg f_1$; not models of f_1.
- $f_1 = \neg\neg f_1 = \neg\,(m_0 + m_2 + m_4 + m_6) =$
  $= M_0M_2M_4M_6 = f_2$ because $\neg\, m_i \equiv M_i$

Method 2: Obtain Full CNF

- Construct a truth table for the formula.
- Use each non-model interpretation in the table to construct a clause.
  - If a variable is true, use the negative literal of this variable in the clause
  - If a variable is false, use the variable in the clause
- Connect the clauses with $\wedge$'s.

21

How to find Full CNF of $(p \vee q) \rightarrow \neg r$

p	q	r	(p ∨ q)	¬r	(p ∨ q)→¬r
1	1	1	1	0	0
1	*1*	*0*	*1*	*1*	*1*
1	0	1	1	0	0
1	*0*	*0*	*1*	*1*	*1*
0	1	1	1	0	0
0	*1*	*0*	*1*	*1*	*1*
0	*0*	*1*	*0*	*0*	*1*
0	*0*	*0*	*0*	*1*	*1*

There are 3 no-model interpretations: 3 clauses.

$(\neg p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$

22

Method 2: Obtain Full DNF

- Construct a truth table for the formula.
- Use each model in the truth table to construct a minterm
  - If the variable is true, use the variable in the minterm
  - If a variable is false, use the negative literal of the variable in the minterm
- Connect the minterms with $\vee$'s.

23

How to find Full DNF of $(p \vee q) \rightarrow \neg r$

p	q	r	(p ∨ q)	¬r	(p ∨ q)→¬r
1	1	1	1	0	0
1	*1*	*0*	*1*	*1*	*1*
1	0	1	1	0	0
1	*0*	*0*	*1*	*1*	*1*
0	1	1	1	0	0
0	*1*	*0*	*1*	*1*	*1*
0	*0*	*1*	*0*	*0*	*1*
0	*0*	*0*	*0*	*1*	*1*

There are 5 models, so there are 5 minterms

$(p \vee q) \rightarrow \neg r \equiv (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee$
$(\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$

24

Dual of NNFs

- Dual of a NNF formula is another formula where $\wedge$ is replaced by $\vee$ and $\vee$ is replaced by $\wedge$ and the sign of literals are changed.
    - dual(p) = $\neg$p if p is a positive literal
    - dual($\neg$ p) = p if p is a negative literal
    - dual(A $\vee$ B) = dual(A) $\wedge$ dual(B)
    - dual(A $\wedge$ B) = dual(A) $\vee$ dual(B)
- **Example:** dual($\neg p \vee \neg q \vee r$) = $p \wedge q \wedge \neg r$
- **Theorem**: If A is in NNF, then dual(A) is in NNF and dual(A) $\equiv \neg$ A.
- Proof: Induction on the structure of A and use DeMorgan's laws.

25

Obtain Full DNF from Full CNF

- If A is in CNF, then dual(A) is in DNF and vice versa.
- If we have a method to obtain full CNF, we may obtain the full DNF of A from the full CNF of $\neg$ A.
- If B is a full CNF equivalent to $\neg$ A, then dual(B) is a full DNF of A.
- If B is a full DNF equivalent to $\neg$ A, then dual(B) is a full CNF of A.

26

Binary Decision Diagrams (BDD)

- Compact data structure for propositional logic, using only ite, 0, and 1.
  - can represents sets of objects (states) encoded as Boolean functions
- Canonical Form
  - reduced ordered BDDs (ROBDD) are canonical
  - Important tool for circuit verification
  - Can display as a directed graph.

27

27

Binary Decision Diagrams (BDD)

- Let + denote $\vee$, x' denote $\neg$x, product denote $\wedge$ (often omitted)
- Let f_x and $f_{x'}$ denote f[x $\leftarrow$1] and f[x $\leftarrow$0]
- Based on recursive Shannon expansion

$$f = x \wedge f_x \vee \neg x \wedge f_{\neg x} = x\, f_x + x'\, f_{x'}$$

- Equivalently, $f = ite(x,\ f_x,\ f_{x'})$
- ite(x, y, z) stands for "if x then y else z"

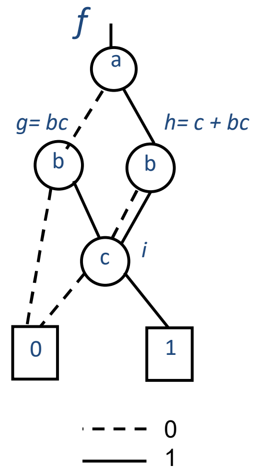
28

28

Shannon Expansion → BDD

- $g = f_{a'} = f(a=0) = bc$
- $h = f_a = f(a=1) = c + bc$
- $g_{b'} = (bc)_{|b=0} = 0$
- $g_b = (bc)_{|b=1} = c$
- $h_{b'} = (c+bc)_{|b=0} = c$
- $h_b = (c+bc)_{|b=1} = c$

$f = ac + bc$



$f = ite(a, h, g)$, $g = ite(b, i, 0)$,
$h = ite(b, i, i)$, $i = ite(c, 1, 0)$

29

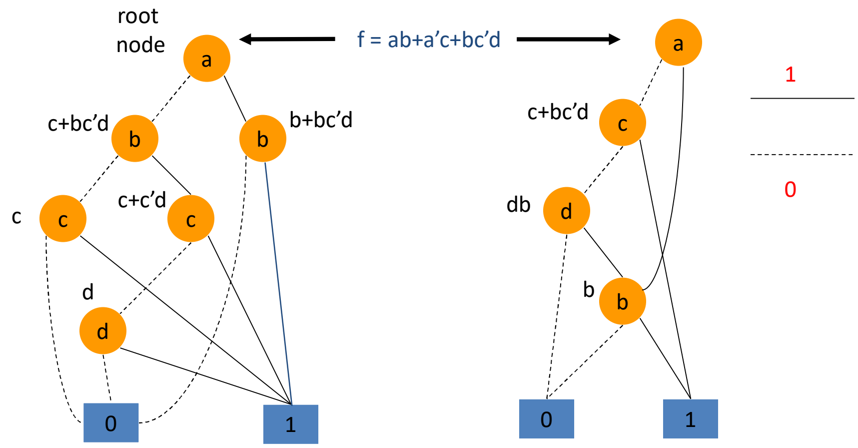
BDDs

- Directed acyclic graph (DAG): one root node, two terminals 0, 1; each node has two children, and a variable.
- Reduced:
  - any node with two identical children is removed
  - two nodes with isomorphic BDD's are merged
- Ordered:
  - Splitting variables always follow the same order along all paths

$$x_{i_1} > x_{i_2} > x_{i_3} > \dots > x_{i_n}$$

30

Example: Variable Orders



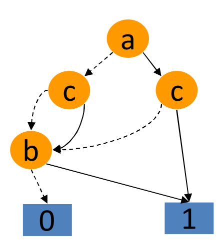
Two different orderings, same function.

31

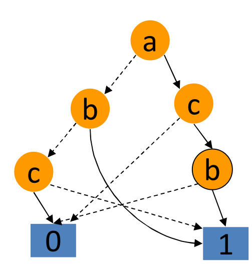
OBDD

Ordered BDD (OBDD): Input variables are ordered - each path from root to sink visits nodes with labels (variables) in descending order.

ordered
order = a,c,b



Not ordered



32

ROBDD

Reduced Ordered BDD (ROBDD)

Reduction rules:

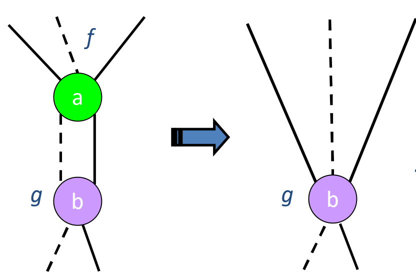
1. if the two children of a node are the same, the node is eliminated: replace ite(v, f, f) by f.
2. two nodes have isomorphic graphs => replace by one by the other

These two rules make ROBDD as a canonical form, so that each node represents a distinct logic function.

33

BDD Reduction Rules -1

1. Eliminate *redundant* nodes (with both edges pointing to same node)



$f = a'\, g(b) + a\, g(b) = g(b)$

$(f_a = f_{a'})$

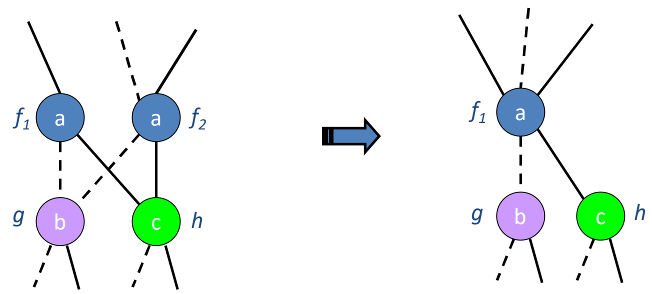
$$f = ite(a, g, g) \Rightarrow f = g$$

34

34

BDD Reduction Rules -2

2. Merge duplicate nodes (*isomorphic* subgraphs)
   - Nodes must be unique



$f_1 = a'\, g(b) + a\, h(c) = f_2$

$f_1 = f_2$

$f_1 = ite(a, h, g) = f_2 \Rightarrow f_1 = f_2$

35

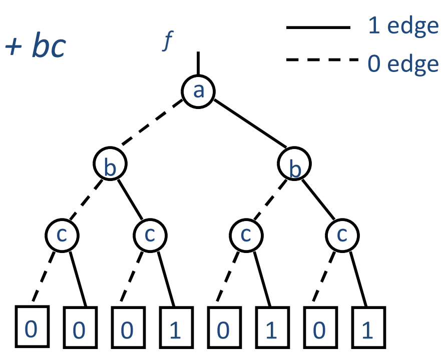
BDD Construction from Truth Table

- A slow method for getting a Reduced Ordered BDD

$f = ac + bc$

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

Truth table

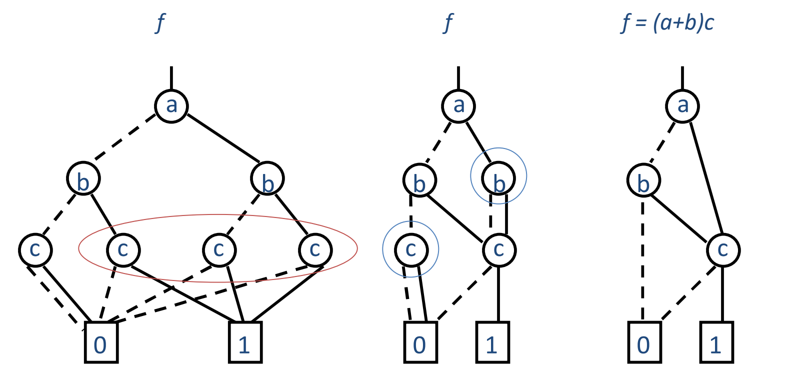


Decision tree

36

36

BDD Construction – cont'd



1. Merge terminal nodes

2. Merge duplicate nodes

3. Remove redundant nodes

37

37

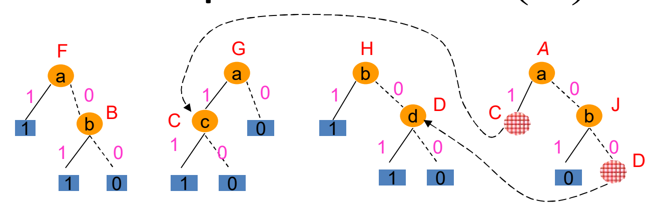
Recursive Construction of ITE

```
Algorithm ROBDD(A)
  // input: formula A
  // output: the ROBDD node of A
  // a hash table for triples (v, x, y).
  if (vars(A) = {}) return simplify(A)
  v := TOP_VARIABLE(vars(A)) // top variable
  x := ROBDD(f[v ← 1])       // recursive calls
  y := ROBDD(f[v ← 0])
  if (x = y) return x         // reduction
  p := LOOKUP_HASH_TABLE(v, x, y)
  if (p ≠ null) return p      // sharing
  return SAVE_CREATE_NODE(v, x, y)
```

38

38

Example: ROBDD(A)



F,G,H,I,J,B,C,D are pointers

A $= ite(F, G, H)$

$= (a,\ ite(F_a, G_a, H_a),\ ite(F_{\bar{a}}, G_{\bar{a}}, H_{\bar{a}}))$

$= (a,\ ite(1, C, H),\quad ite(B, 0, H))$

$= (a,\ C,\quad (b,\ ite(B_b, 0, H_b),\ ite(B_{\bar{b}}, 0, H_{\bar{b}}))$

$= (a,\ C,\quad (b,\ ite(1, 0, 1),\ ite(0, 0, D)))$

$= (a,\ C,\quad (b, 0, D))$

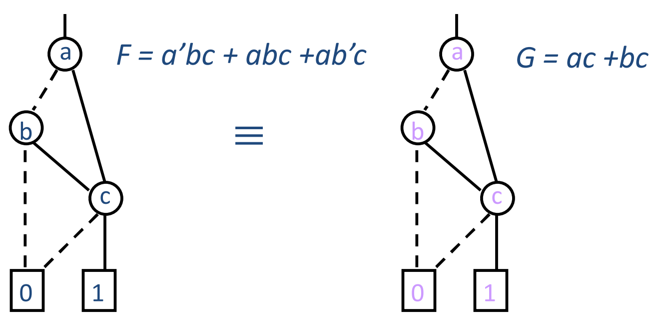
$= (a,\ C,\quad J)$ // both *J and A* are new nodes

39

39

Application to Verification

- Equivalence Checking of *combinational* circuits
- *Canonicity* property of OBDDs:
  - if F and G are equivalent, their OBDDs are identical (for the same ordering of variables)

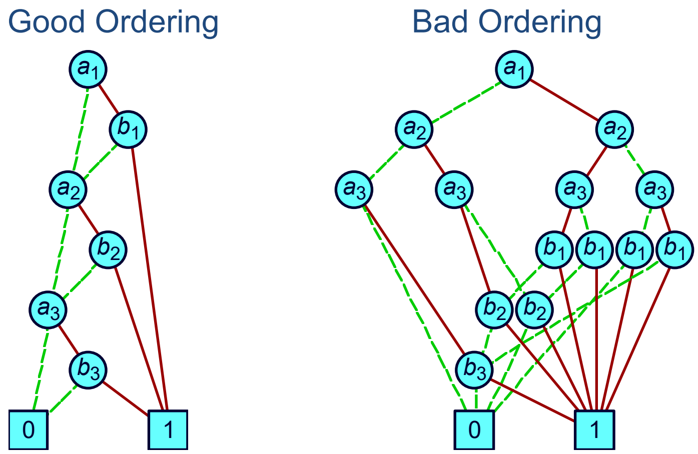


40

40

Effect of Variable Ordering

$$(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee (a_3 \wedge b_3)$$



Linear Growth

Exponential Growth

41

Static Variable Ordering

- Variable ordering is computed up-front based on the problem structure
- Works very well for many combinational functions that come from circuits
  - general scheme: control variables first
- Work bad for unstructured problems
  - e.g., using BDDs to represent arbitrary sets
- Lots of research in ordering algorithms
  - simulated annealing, genetic algorithms
  - give better results but extremely costly

42

42