Binary Decision Diagrams Part 2

15-414 Bug Catching: Automated Program Verification and Testing

Sagar Chaki
September 14, 2011

BDDs Recap

Typically mean Reduced Ordered Binary Decision Diagrams (ROBDDs)

- Can be viewed as reduced forms of Ordered Binary Decision Trees
  - Obtained by eliminating duplicate nodes and redundant nodes
- Often substantially smaller than the OBDT

Canonical representation of Boolean formulas

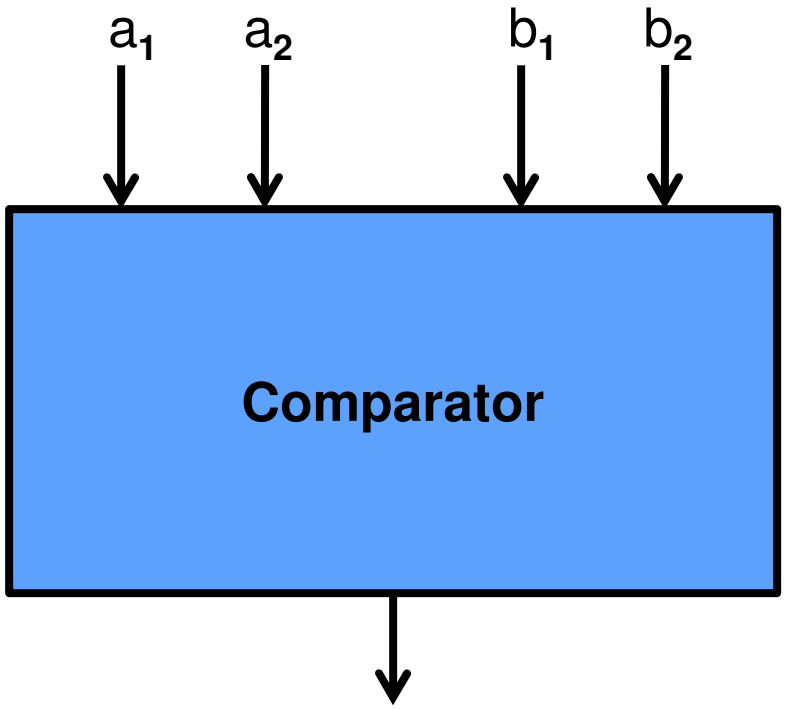
- Unlike other normal forms like CNF and DNF

Size of BDD depends critically on variable ordering

In practice, BDDs are built up from their components

- Via (efficient) Boolean operations
- Dynamic variable ordering used to manage BDD size

Running Example: Comparator



(A) $f = 1 \Leftrightarrow a_1 = b_1 \wedge a_2 = b_2$ ✓

(B) $f = 1 \Leftrightarrow a_1 = b_1 \vee a_2 = b_2$

Conjunctive Normal Form

$$(b_1 \vee a_1) \wedge (\neg a_1 \vee \neg b_1) \wedge (\neg a_2 \vee b_2) \wedge (\neg b_2 \vee a_2)$$

$$(\neg b_1 \vee a_1) \wedge (\neg a_1 \vee b_1) \wedge (\neg a_2 \vee b_2) \wedge (\neg b_2 \vee a_2)$$

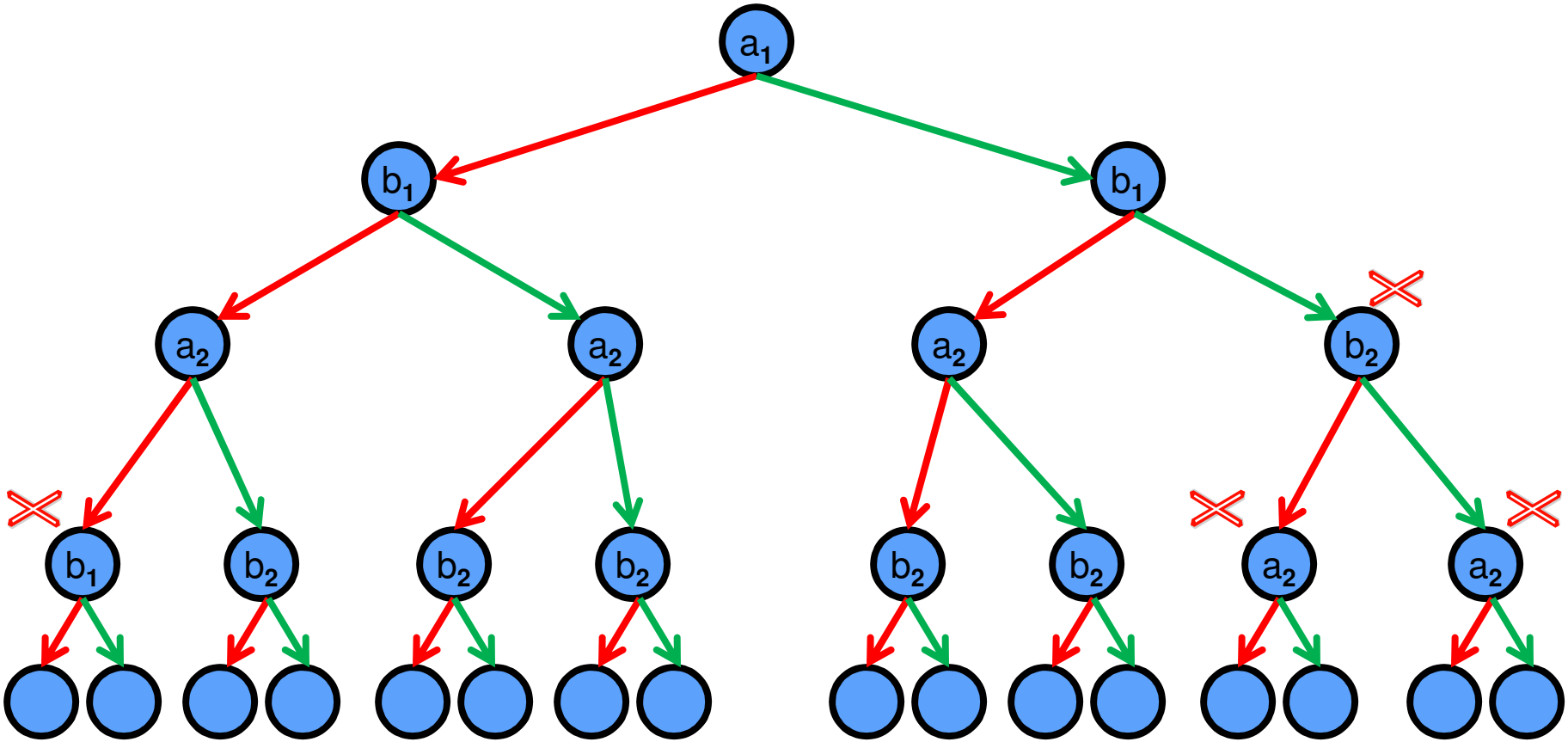
$$(\neg a_1 \vee b_1) \wedge (\neg b_1 \vee a_1) \wedge (\neg a_2 \vee b_2) \wedge (\neg b_2 \vee a_2)$$

$$(\neg a_1 \vee b_1) \wedge (\neg b_1 \vee a_1) \wedge (\neg a_2 \vee \neg b_2) \wedge (b_2 \vee a_2)$$

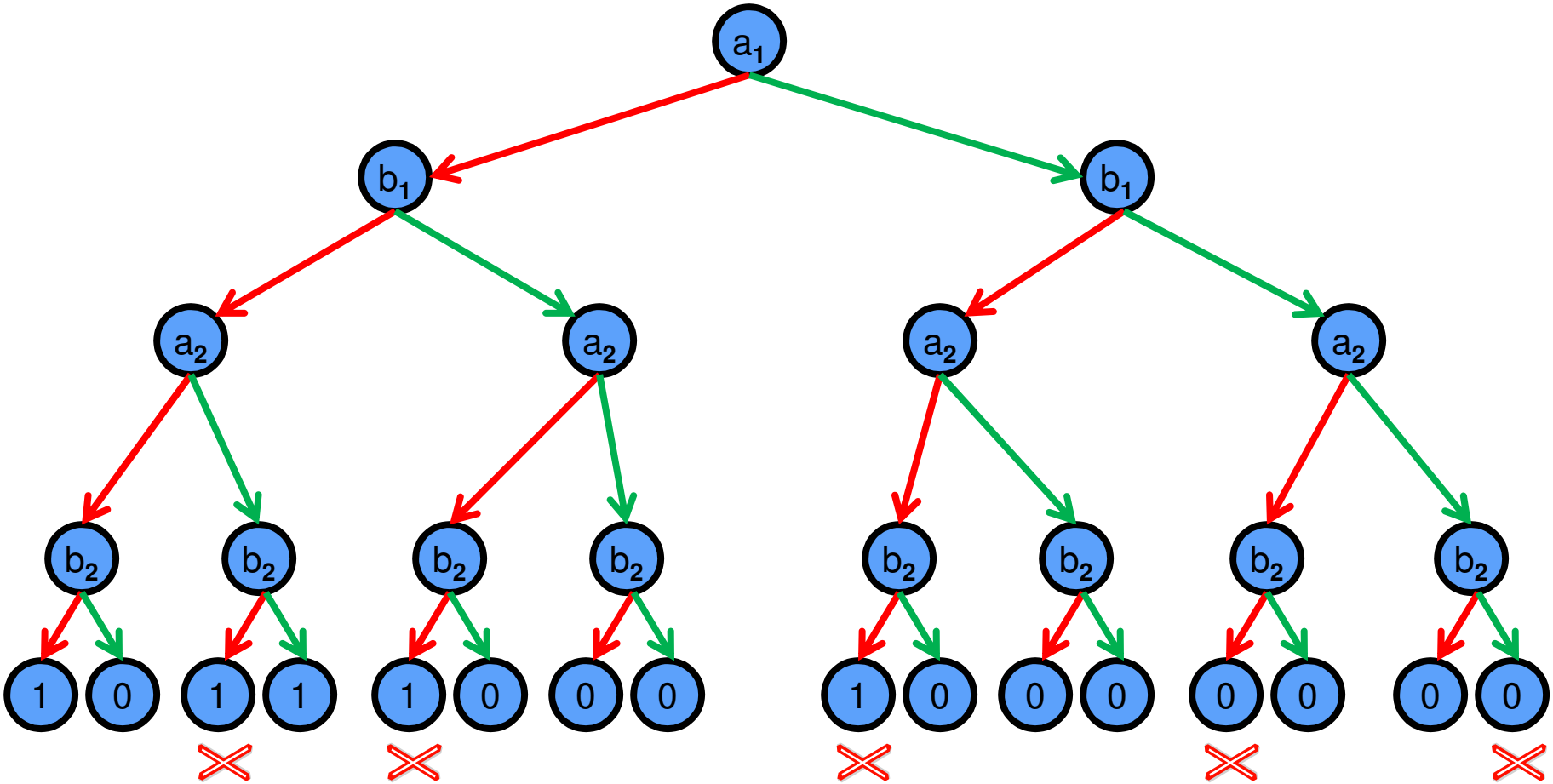
Truth Table

Row#	a_1	b_1	a_2	b_2	f
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Representing a Truth Table using a Graph



Representing a Truth Table using a Graph



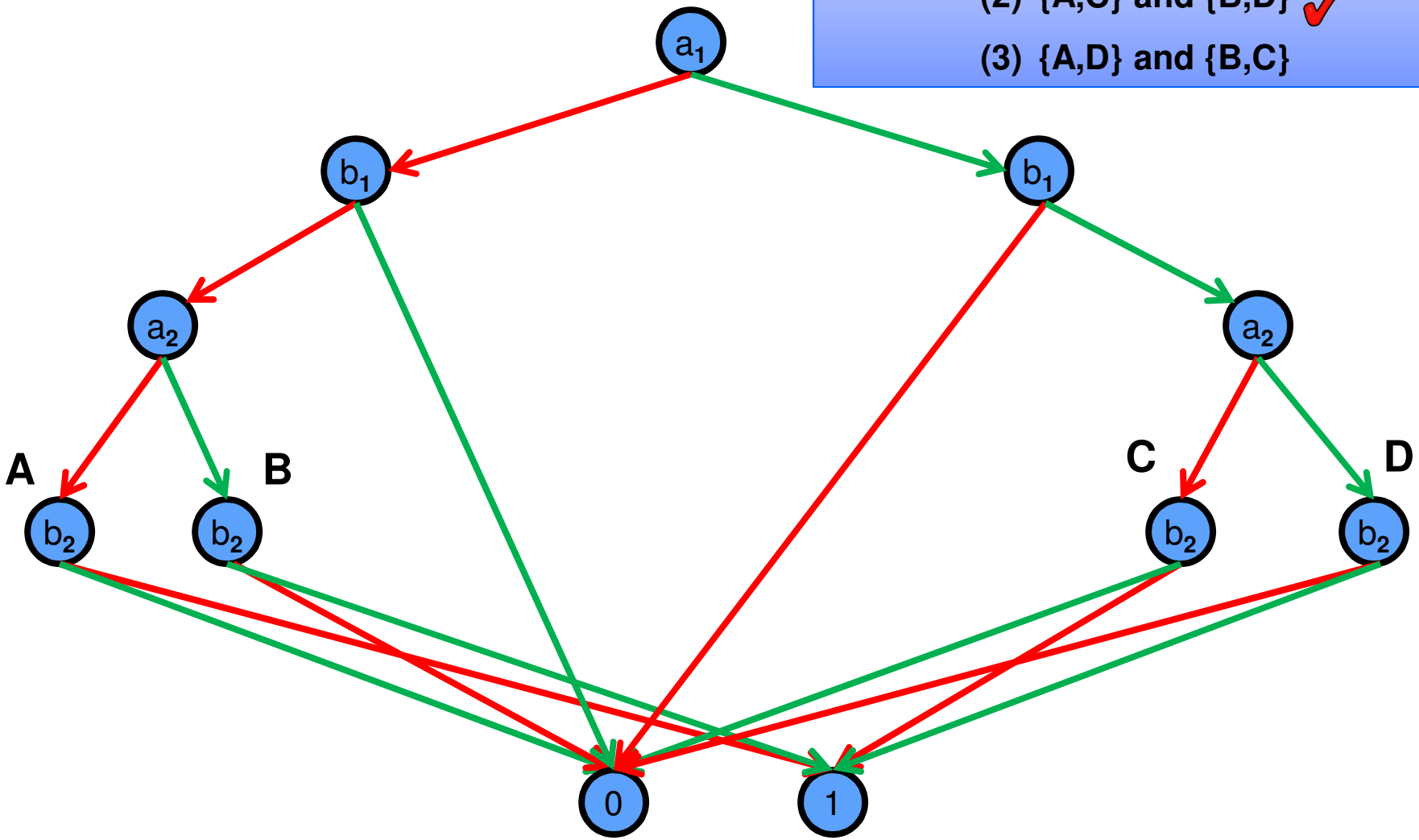
OBDT to ROBDD

Which pairs are isomorphic?

(1) {A,B} and {C,D}

(2) {A,C} and {B,D} ✓

(3) {A,D} and {B,C}

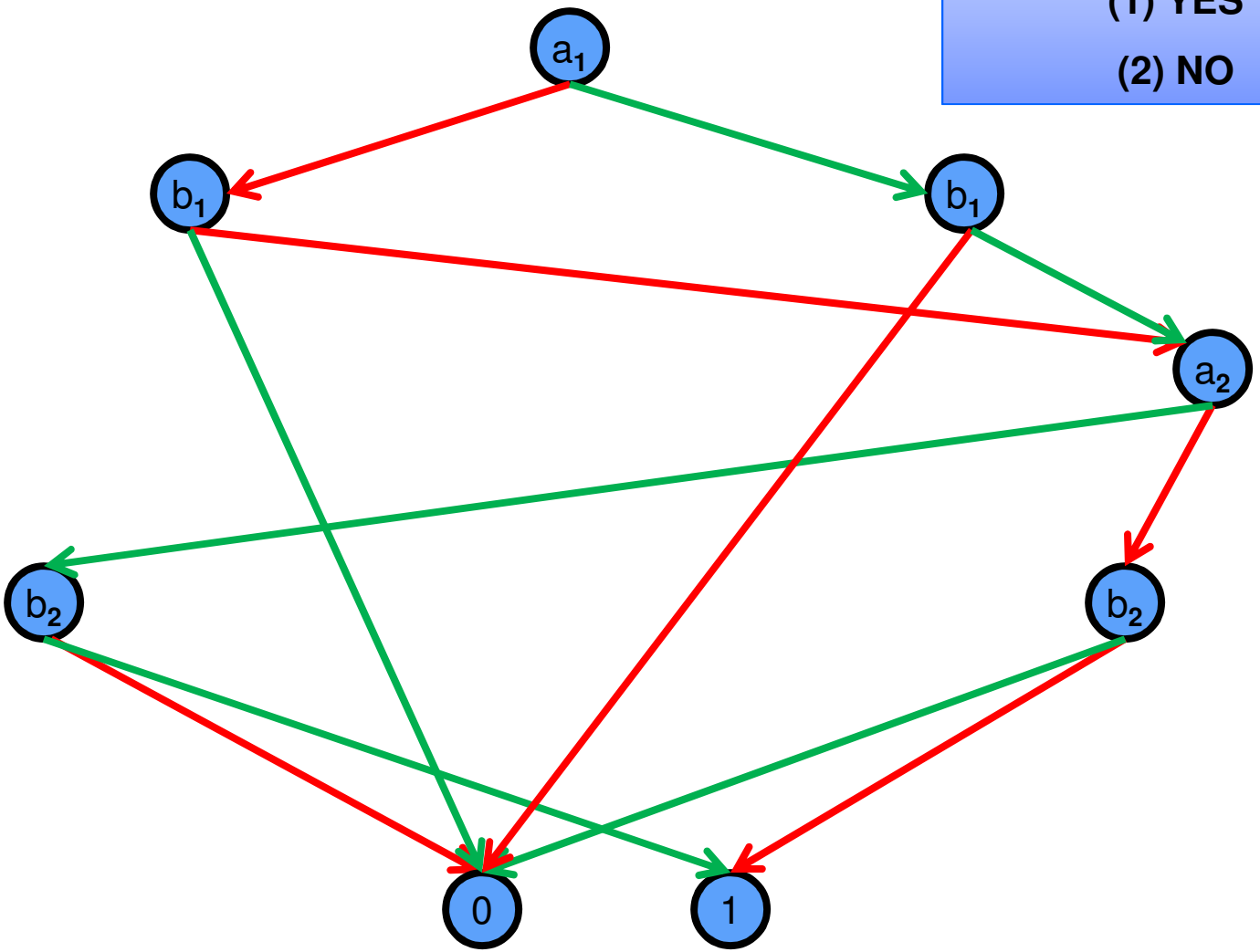


OBDT to ROBDD

Is this a ROBDD?

(1) YES ✓

(2) NO



OBDT to ROBDD

What function does X represent?

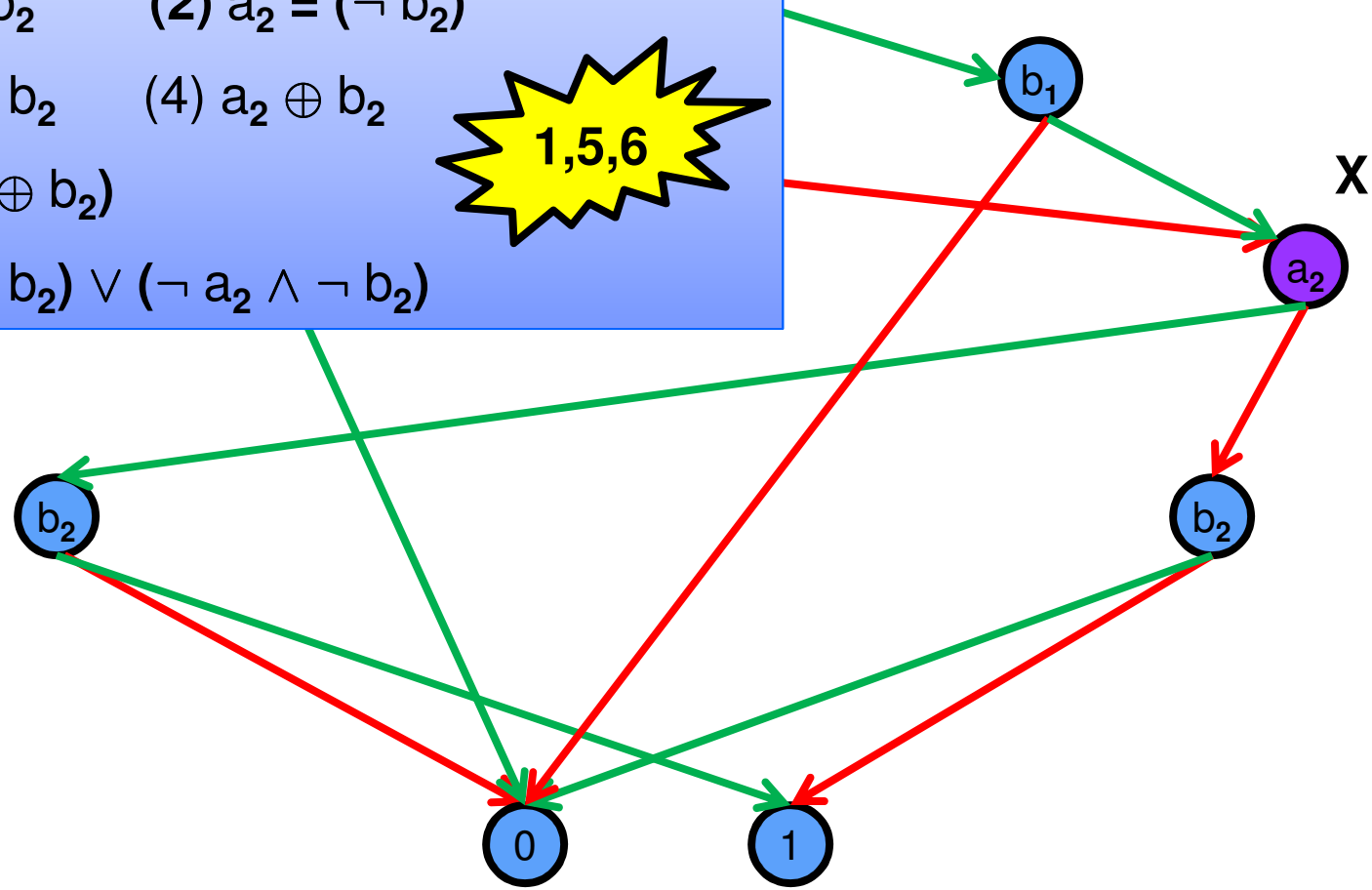
(1) $a_2 = b_2$ **(2)** $a_2 = (\neg b_2)$

(3) $a_2 \Rightarrow b_2$ (4) $a_2 \oplus b_2$

(5) $\neg(a_2 \oplus b_2)$

(6) $(a_2 \wedge b_2) \vee (\neg a_2 \wedge \neg b_2)$

1,5,6



ROBDD (a.k.a. BDD) Summary

If BDD(f_1) and BDD(f_2) are isomorphic then:

✓ 1. $f_1 = f_2$
✓ 2. f_1 and f_2 have the same variables
✓ 3. BDD(f_1) and BDD(f_2) have the same variable ordering

If BDD(f) is the leaf node "1" then f is:

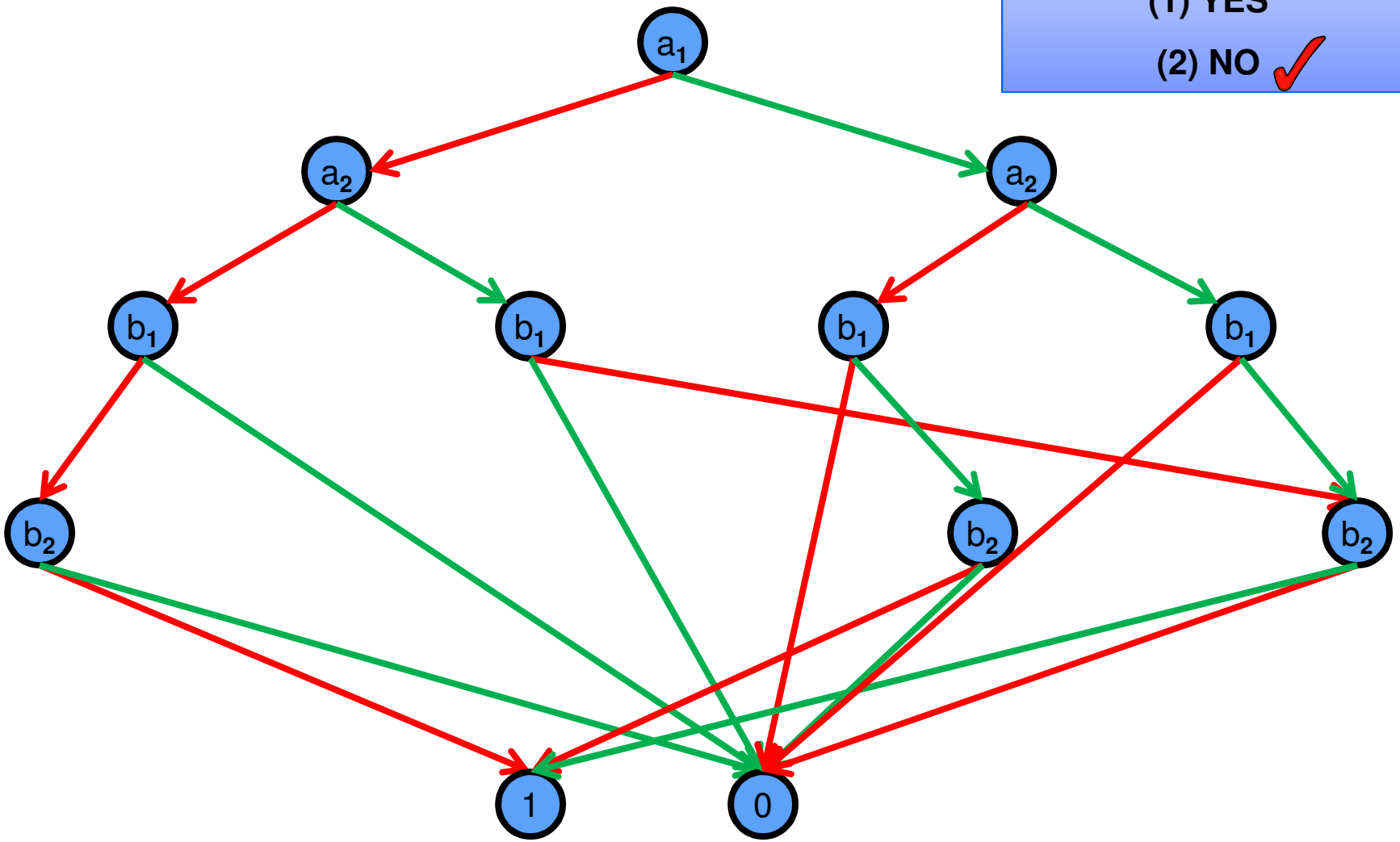
1. Satisfiable
2. Unsatisfiable
✓ 3. Valid

ROBDD and variable ordering

Is this a ROBDD?

(1) YES

(2) NO ✓

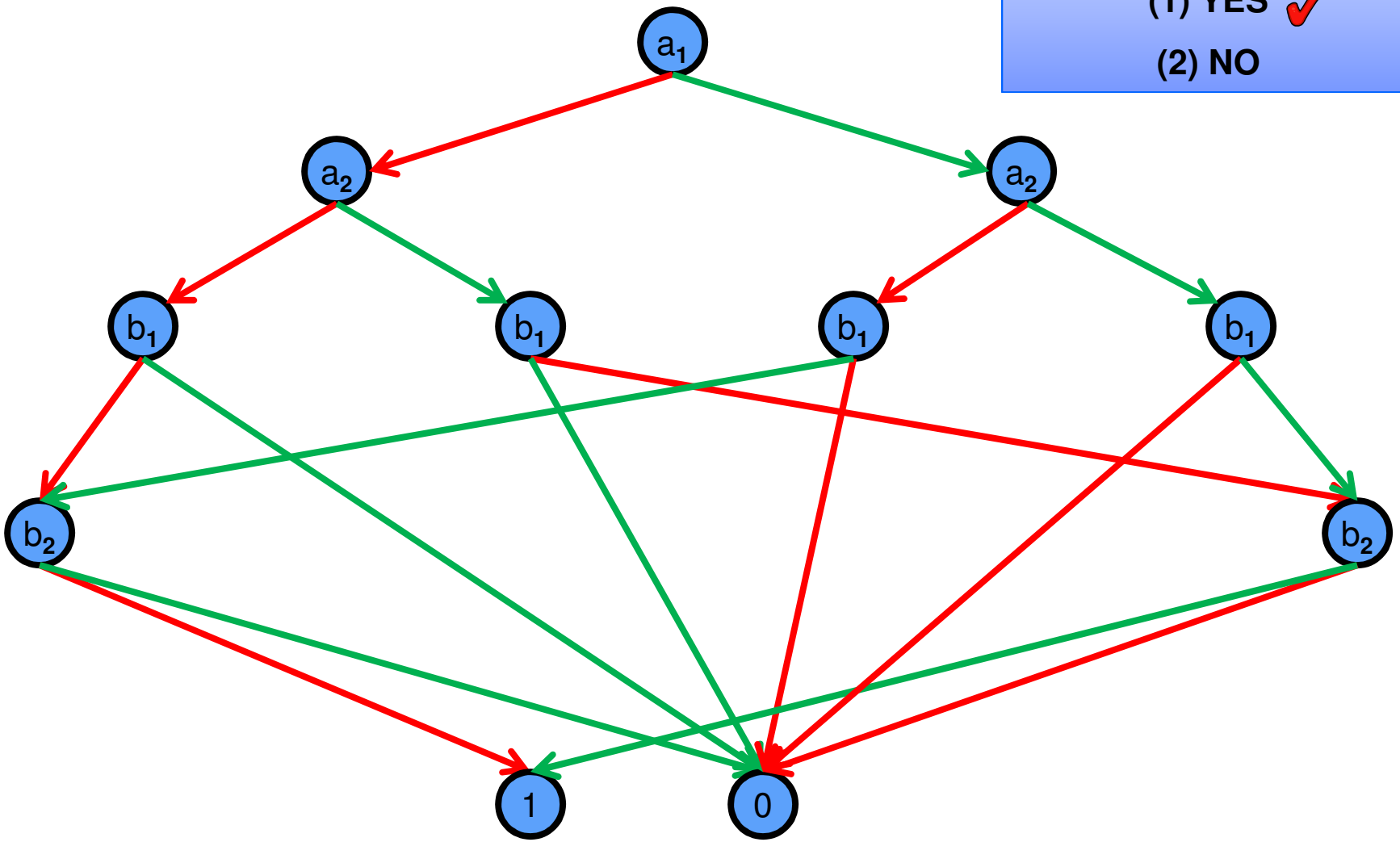


ROBDD and variable ordering

Is this a ROBDD?

(1) YES ✓

(2) NO



ROBDD and variable ordering

There exists a function whose BDD grows polynomially in the number of variables for some ordering and exponentially for others?

- TRUE

There exists a function whose BDD grows exponentially for all variable orderings?

- TRUE

There exists a function whose BDD grows linearly for all variable orderings?

- TRUE

BDD Operations

True : BDD(TRUE)

False: BDD(FALSE)

Var : $v \mapsto$ BDD(v)

Not : BDD(f) $\mapsto$ BDD($\neg$f)

And : BDD(f_1) $\times$ BDD(f_2) $\mapsto$ BDD($f_1 \wedge f_2$)

Or : BDD(f_1) $\times$ BDD(f_2) $\mapsto$ BDD($f_1 \vee f_2$)

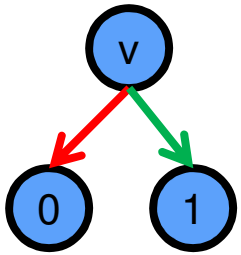
Exist : BDD(f) $\times$ v $\mapsto$ BDD($\exists$ v. f)

Basic BDD Operations

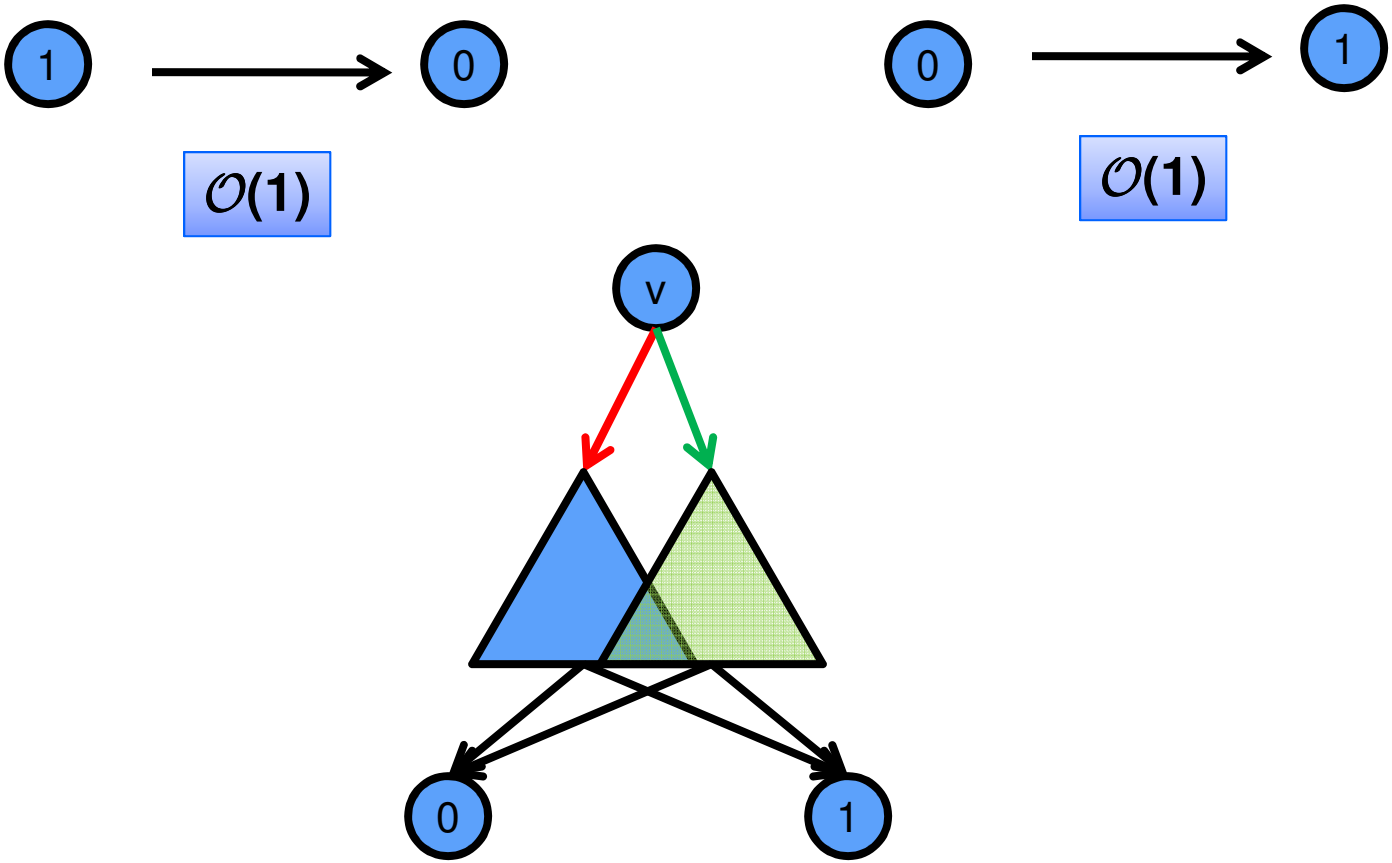
True

False

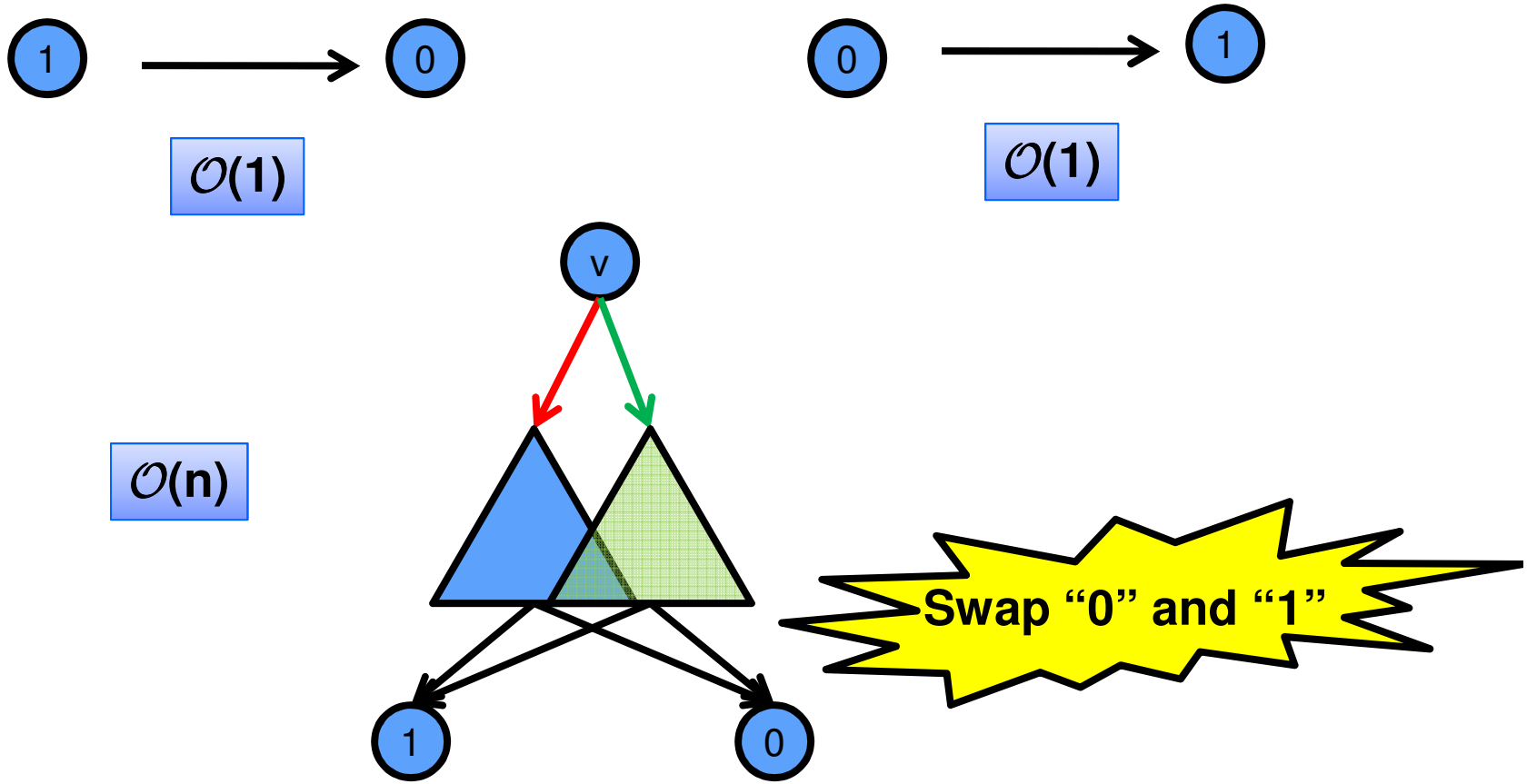
Var(v)



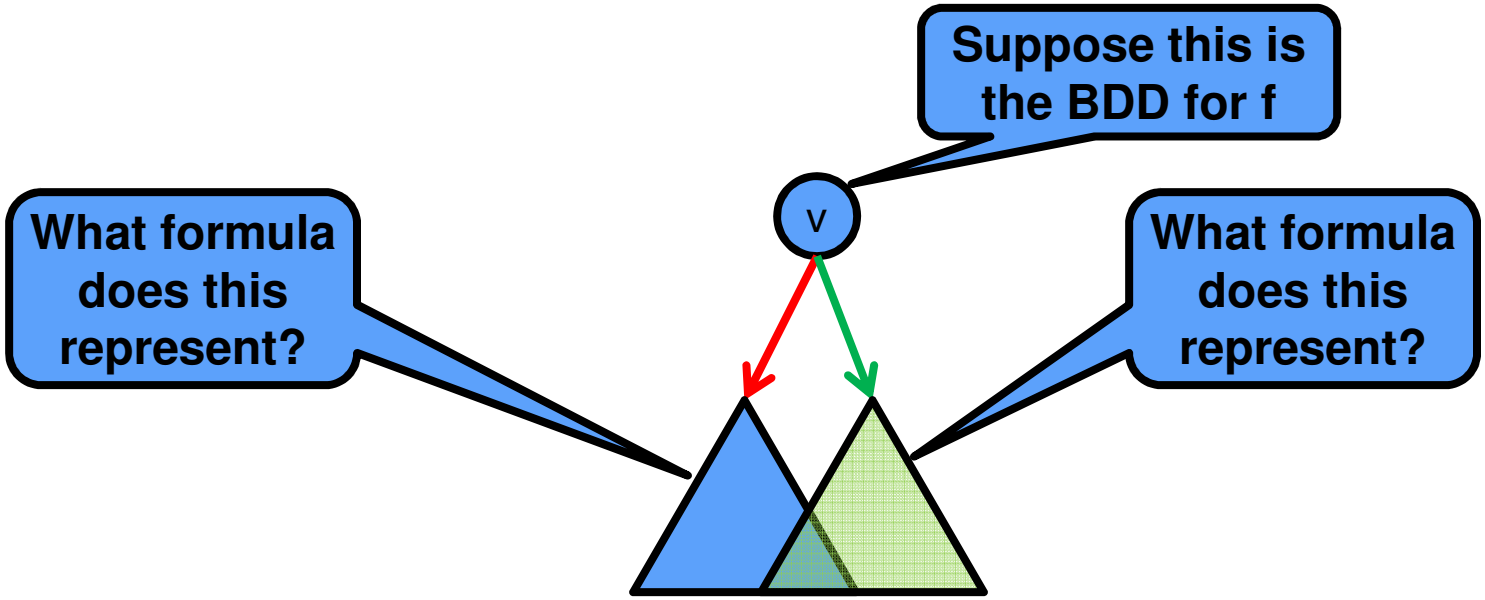
BDD Operations: Not



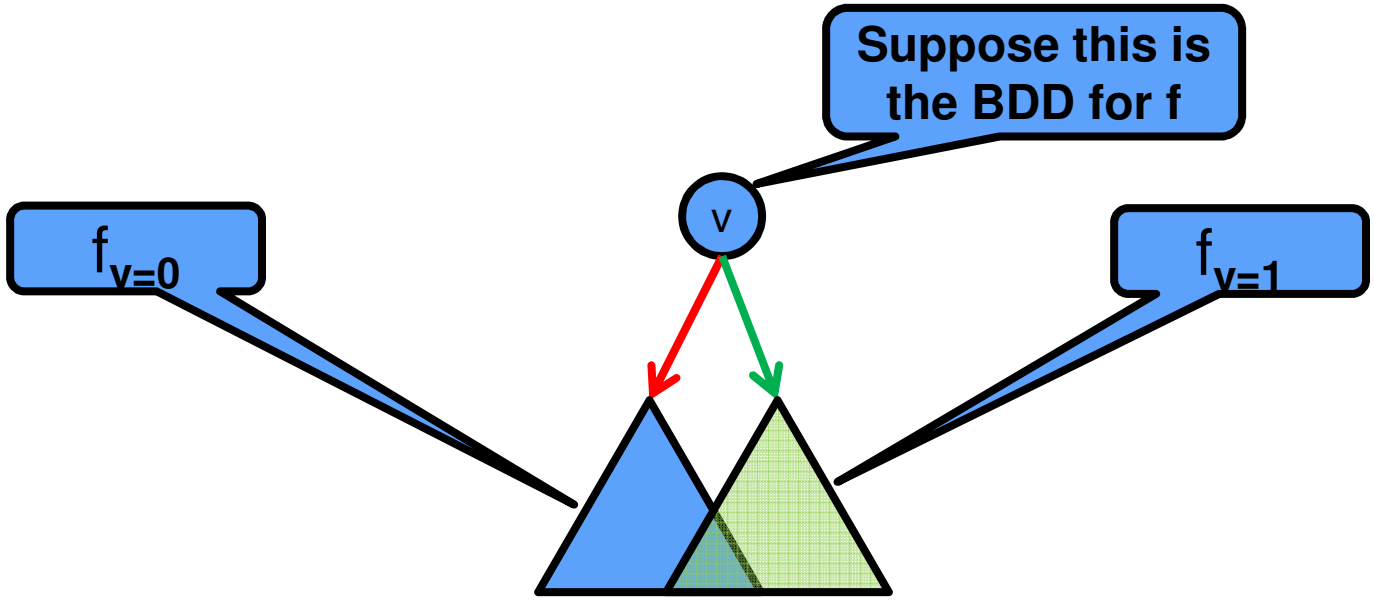
BDD Operations: Not



BDD Operations: And



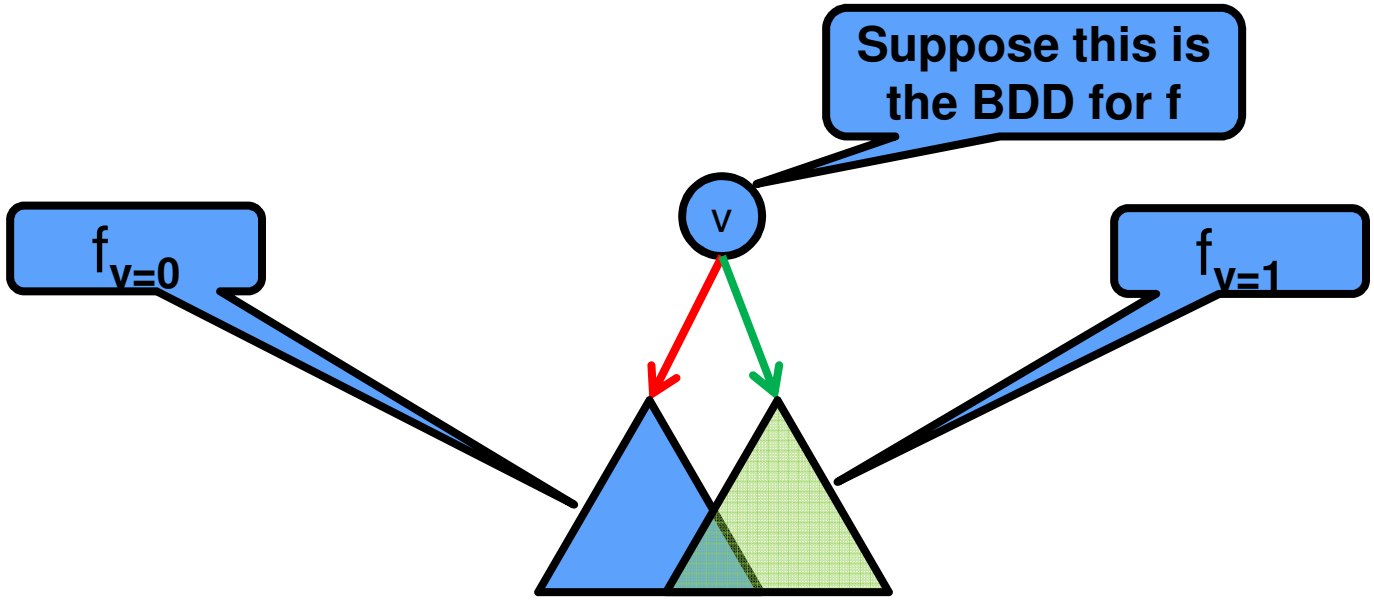
BDD Operations: And



$f_{v=0}$ and $f_{v=1}$ are known as the co-factors of f w.r.t. v

$$f = (X \wedge f_{v=0}) \vee (Y \wedge f_{v=1})$$

BDD Operations: And



$f_{v=0}$ and $f_{v=1}$ are known as the co-factors of f w.r.t. v

$$f = (\neg v \wedge f_{v=0}) \vee (v \wedge f_{v=1})$$

BDD Operations: And (Simple Cases)

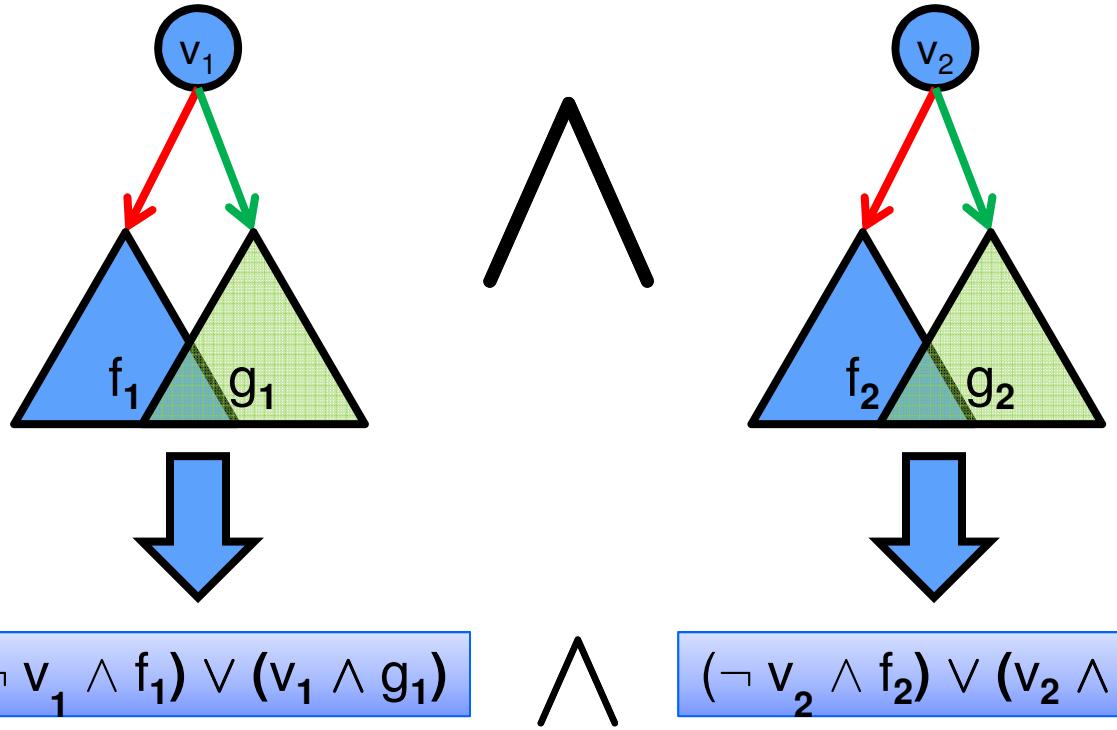
And (f, 0) = 0

And (f, 1) = f

And (1 ,f) = f

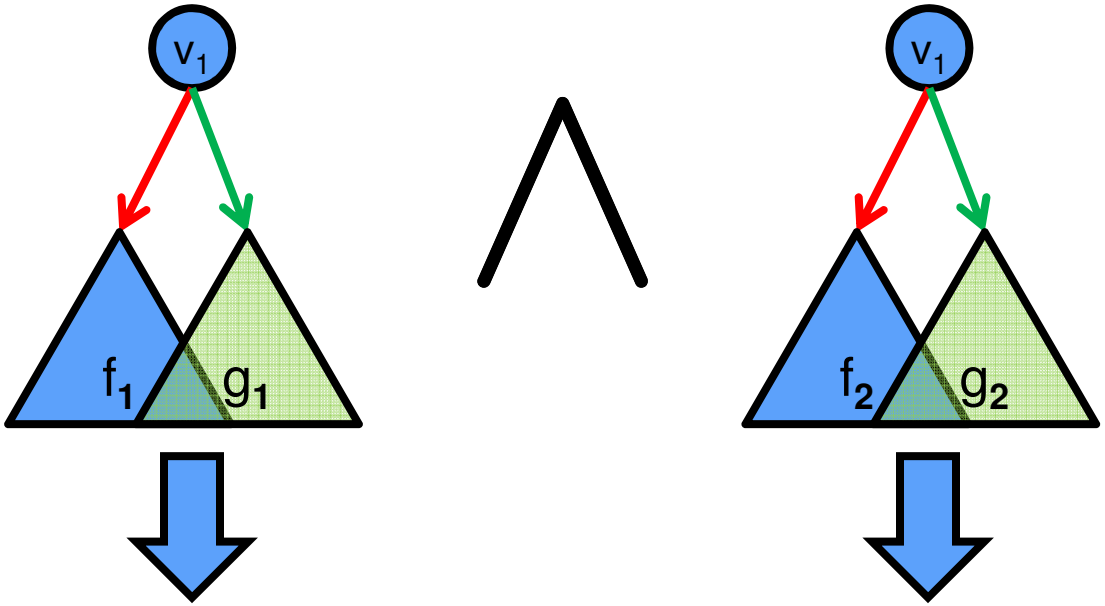
And (0 ,f) = 0

BDD Operations: And (Complex Case)



BDD Operations: And (Complex Case 1)

$v_1 = v_2$



$(\neg v_1 \wedge f_1) \vee (v_1 \wedge g_1)$ $\wedge$ $(\neg v_1 \wedge f_2) \vee (v_1 \wedge g_2)$

BDD Operations: And (Complex Case 1)

$v_1 = v_2$

$(\neg v_1 \wedge X) \vee (v_1 \wedge Y)$

$(\neg v_1 \wedge f_1) \vee (v_1 \wedge g_1)$ $\wedge$ $(\neg v_1 \wedge f_2) \vee (v_1 \wedge g_2)$

BDD Operations: And (Complex Case 1)

$v_1 = v_2$

Compute recursively

$(\neg v_1 \wedge (f_1 \wedge f_2)) \vee (v_1 \wedge (g_1 \wedge g_2))$

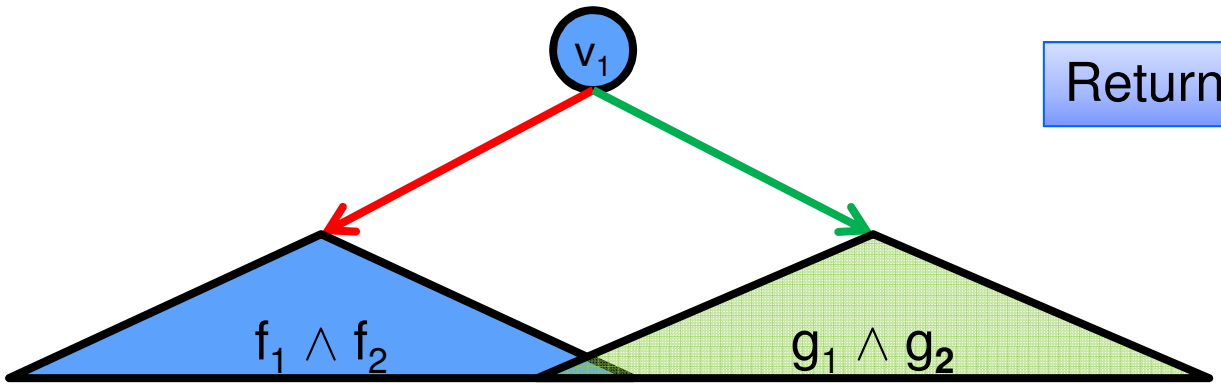
$(\neg v_1 \wedge f_1) \vee (v_1 \wedge g_1)$ $\wedge$ $(\neg v_1 \wedge f_2) \vee (v_1 \wedge g_2)$

BDD Operations: And (Complex Case 1)

$v_1 = v_2$

What if $f_1 \wedge f_2 = g_1 \wedge g_2$?

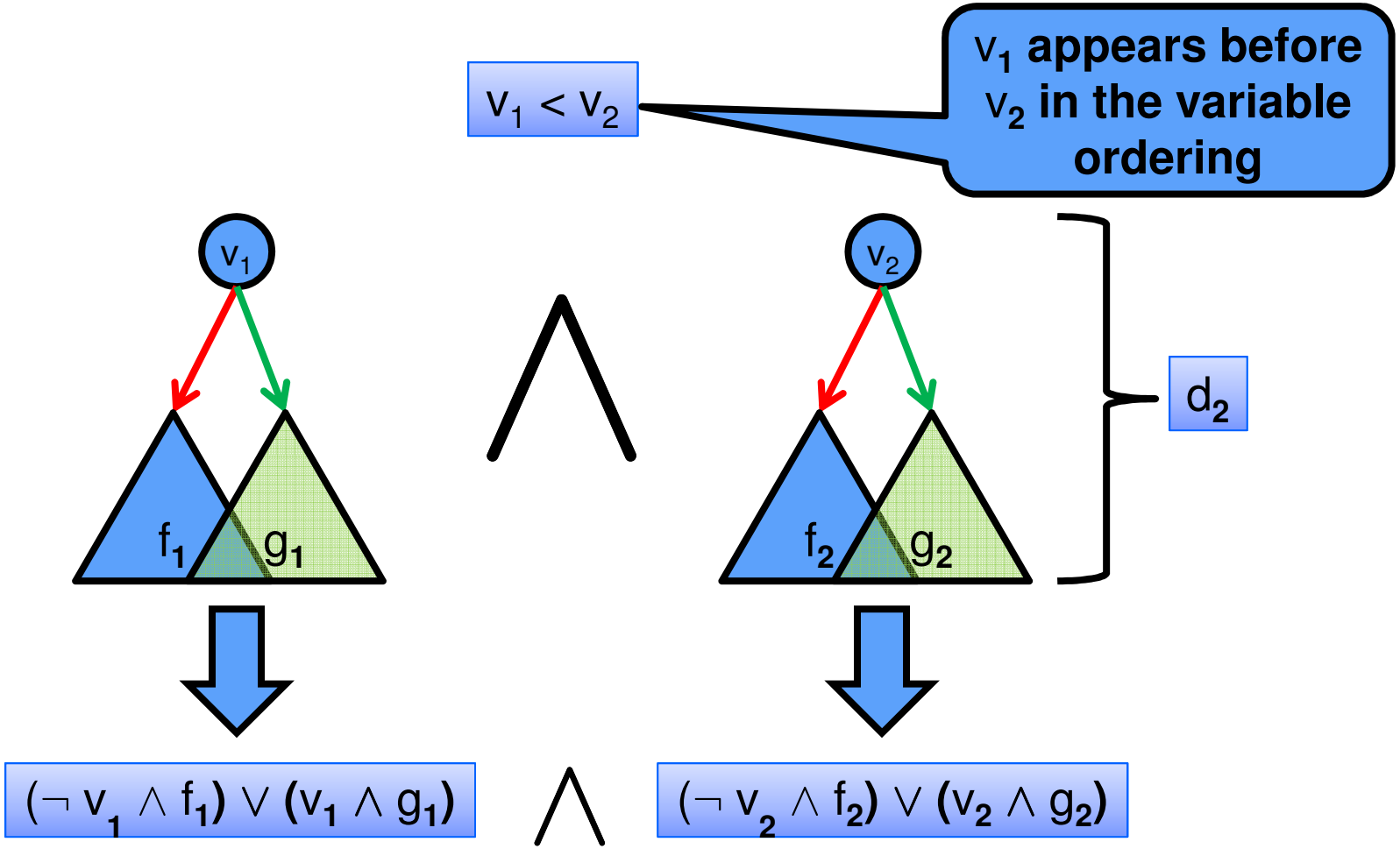
Return $f_1 \wedge f_2$



$(\neg v_1 \wedge (f_1 \wedge f_2)) \vee (v_1 \wedge (g_1 \wedge g_2))$

$(\neg v_1 \wedge f_1) \vee (v_1 \wedge g_1)$ $\wedge$ $(\neg v_1 \wedge f_2) \vee (v_1 \wedge g_2)$

BDD Operations: And (Complex Case 2)

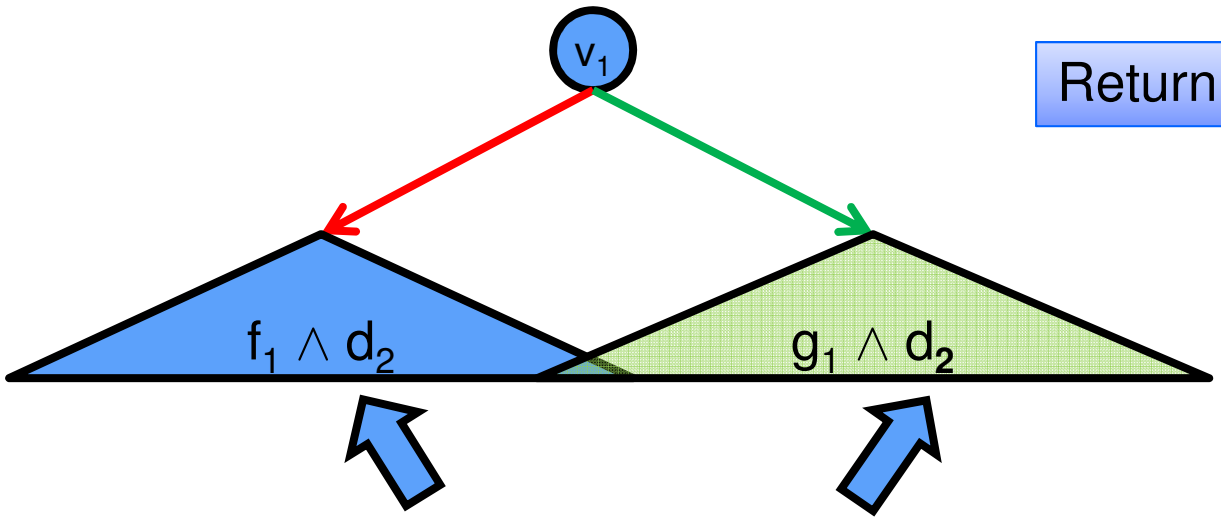


BDD Operations: And (Complex Case 2)

$v_1 < v_2$

What if $f_1 \wedge d_2 = g_1 \wedge d_2$?

Return $f_1 \wedge d_2$



$(\neg v_1 \wedge (f_1 \wedge d_2)) \vee (v_1 \wedge (g_1 \wedge d_2))$

$(\neg v_1 \wedge f_1) \vee (v_1 \wedge g_1)$ $\wedge$ d_2

$\mathcal{O}(n_1 \times n_2)$

BDD Operations: Or

$$\mathrm{Or}(d1, d_2)$$

$$=$$

$$\textbf{Not (And (} \mathrm{Not}(d_1), \mathrm{Not}(d_2) \textbf{))}$$

$$\mathcal{O}(n_1 \times n_2)$$

BDD Operations: Exist

Exist(“0”,v) = ?

BDD Operations: Exist

Exist(“0”,v) = “0”

Exist(“1”,v) = ?

BDD Operations: Exist

$$\text{Exist}(\text{“0”}, v) = \text{“0”}$$

$$\text{Exist}(\text{“1”}, v) = \text{“1”}$$

$$\text{Exist}((\neg v \wedge f) \vee (v \wedge g), v) = ?$$

BDD Operations: Exist

$$\text{Exist}(\text{“0”}, v) = \text{“0”}$$

$$\text{Exist}(\text{“1”}, v) = \text{“1”}$$

$$\text{Exist}((\neg v \wedge f) \vee (v \wedge g), v) = \text{Or}(f,g)$$

$$\text{Exist}((\neg v' \wedge f) \vee (v' \wedge g), v) = ?$$

BDD Operations: Exist

$\mathcal{O}(n^2)$

$$\text{Exist}(\text{“0”},v) = \text{“0”}$$

$$\text{Exist}(\text{“1”},v) = \text{“1”}$$

$$\text{Exist}((\neg v \wedge f) \vee (v \wedge g), v) = \text{Or}(f,g)$$

$$\text{Exist}((\neg v' \wedge f) \vee (v' \wedge g), v) =$$

$$(\neg v' \wedge \text{Exist}(f,v)) \vee (v' \wedge \text{Exist}(g,v))$$

But f is SAT iff $\exists$ V. f is not “0”. So why doesn’t this imply P = NP?

Because the BDD size changes!

BDD Applications

SAT is great if you are interested to know if a solution exists

BDDs are great if you are interested in the set of all solutions

- How many solutions are there?
- How do you do this on a BDD?

Or if your problem involves computing a fixed point

- Set of nodes reachable from a given node in a graph

BDD Application: Counting Sudoku Solutions

	1	2	3	4
1			1	
2				
3	3		2	
4				

How many ways can you solve this puzzle?

BDD Application: Counting Sudoku Solutions

	1	2	3	4
1	0	3	1	2
2	2	1	0	3
3	3	0	2	1
4	1	2	3	0

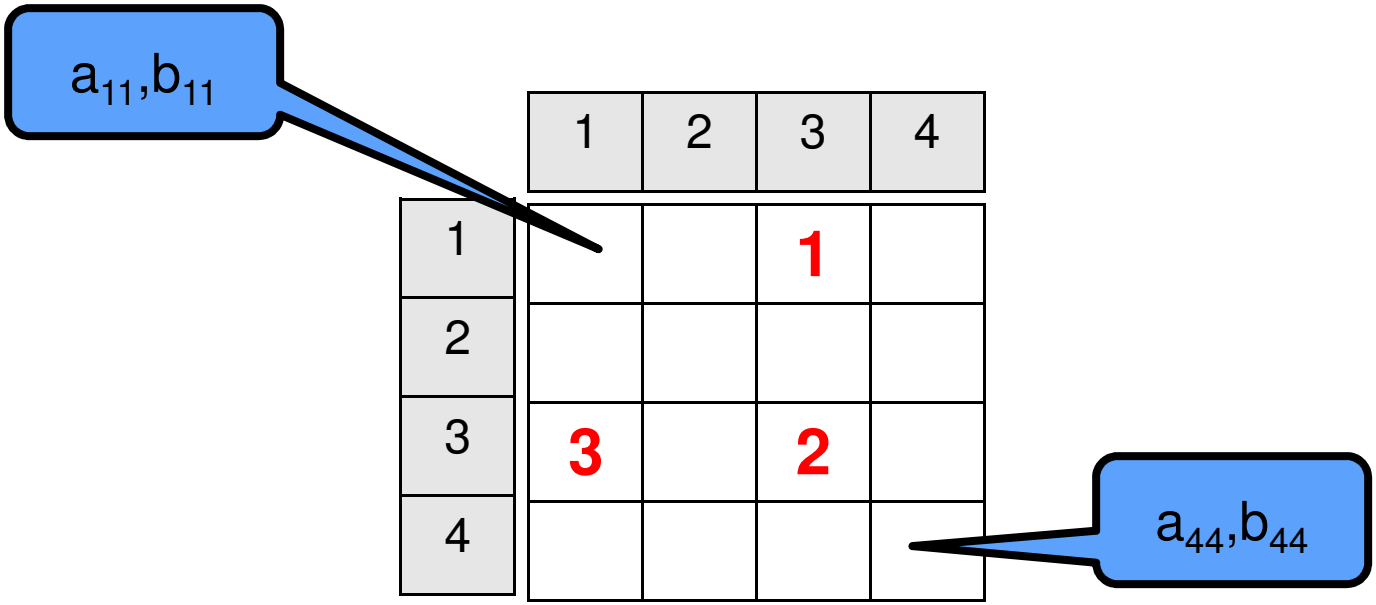
How many ways can you solve this puzzle? At least 2.

BDD Application: Counting Sudoku Solutions

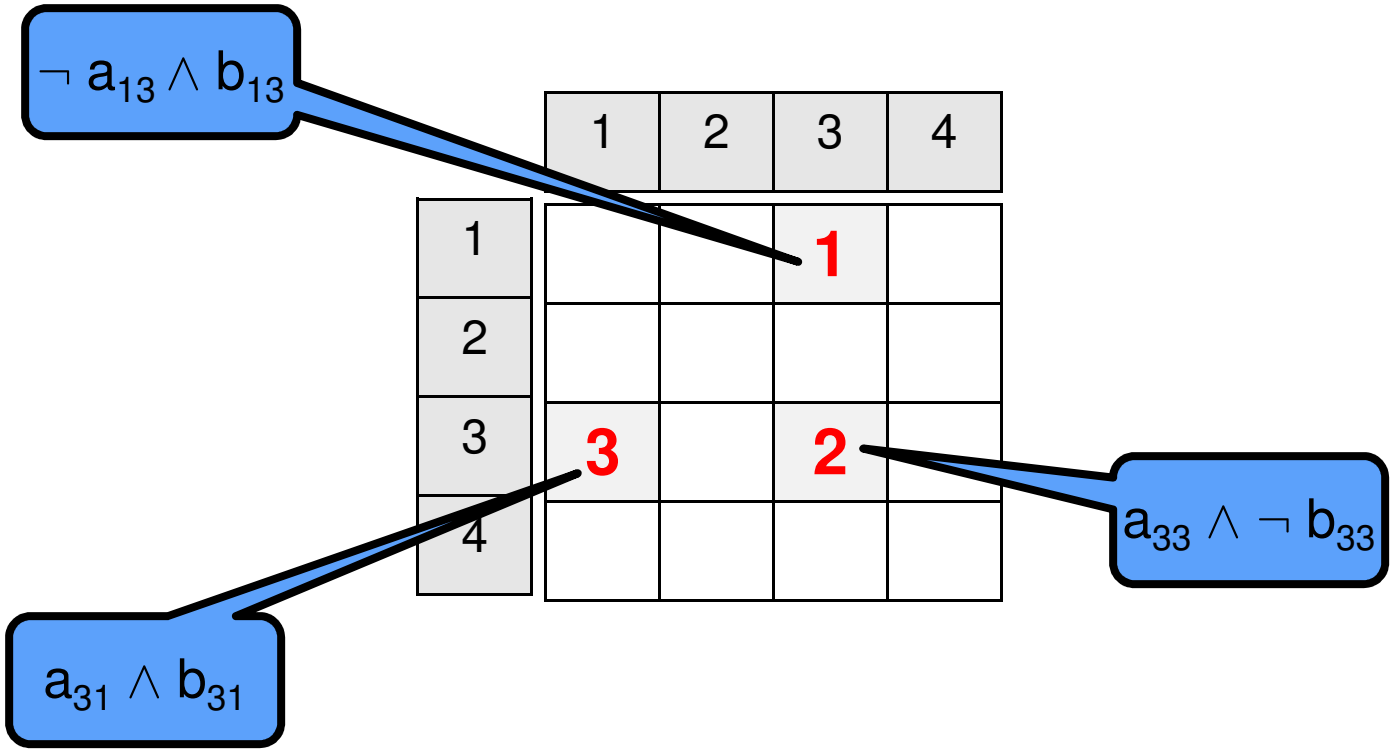
	1	2	3	4
1	0	3	1	2
2	1	2	0	3
3	3	0	2	1
4	2	1	3	0

How many ways can you solve this puzzle? At least 2.

BDD Application: Counting Sudoku Solutions



BDD Application: Counting Sudoku Solutions



BDD Application: Counting Sudoku Solutions

$$a_{11} \oplus a_{12} \vee b_{11} \oplus b_{12}$$
$$\wedge$$
$$a_{11} \oplus a_{13} \vee b_{11} \oplus b_{13}$$
$$\wedge$$
$$a_{11} \oplus a_{14} \vee b_{11} \oplus b_{14}$$
$$\wedge$$
$$a_{12} \oplus a_{13} \vee b_{12} \oplus b_{13}$$
$$\wedge$$
$$a_{12} \oplus a_{14} \vee b_{12} \oplus b_{14}$$
$$\wedge$$
$$a_{13} \oplus a_{14} \vee b_{13} \oplus b_{14}$$

	1	2	3	4
1			1	
2				
3	3		2	
4				

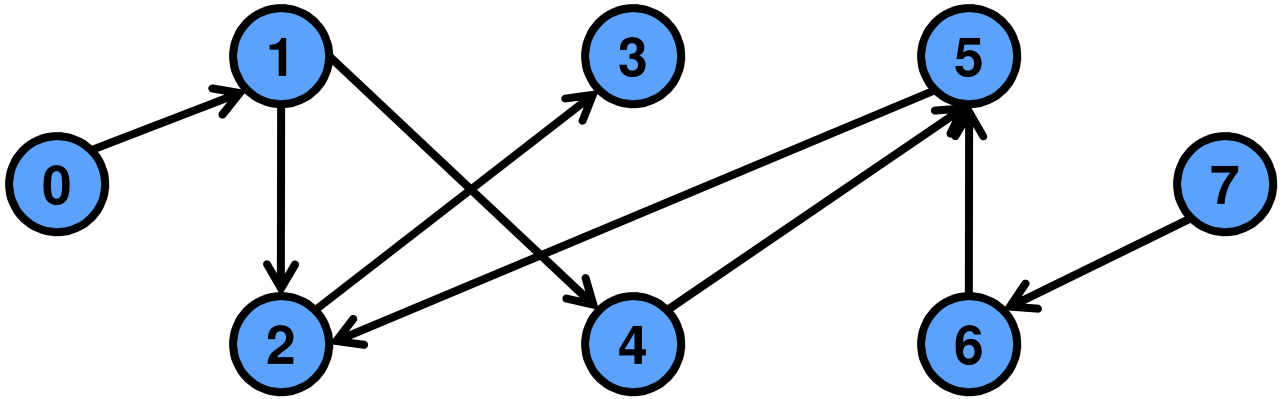
Distinct Elements

Repeat for each row, column and sub-square

Construct BDD

Count number of solutions

Graph Reachability

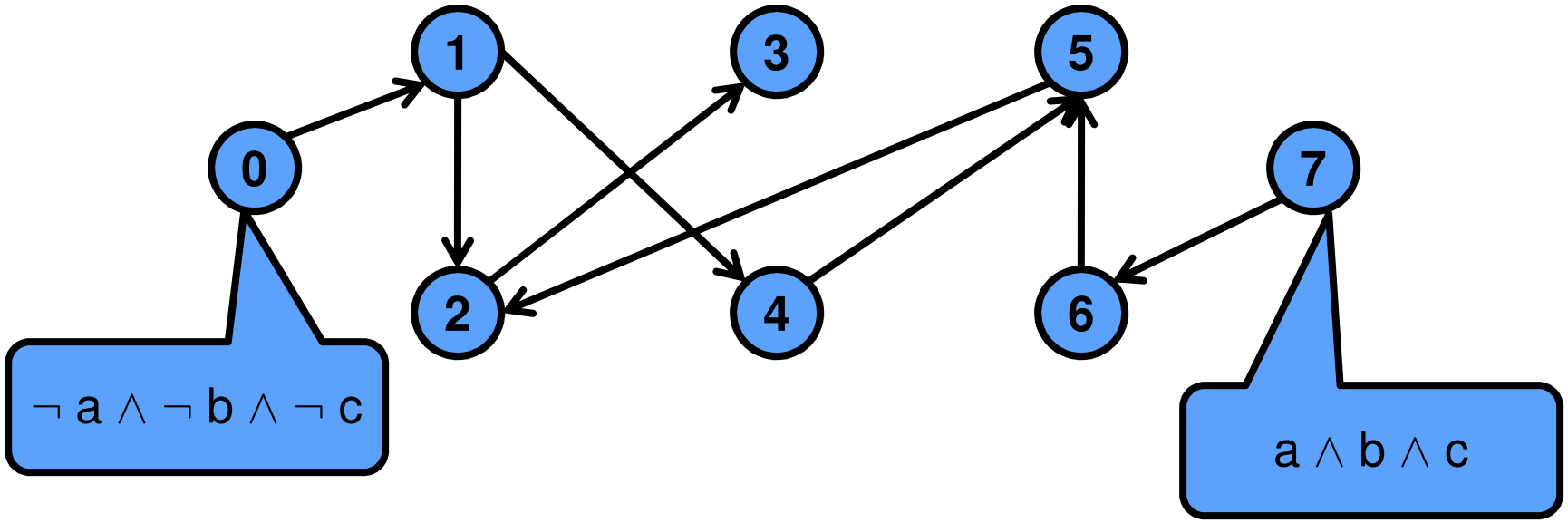


Which nodes are reachable from “7”?

{2,3,5,6,7}

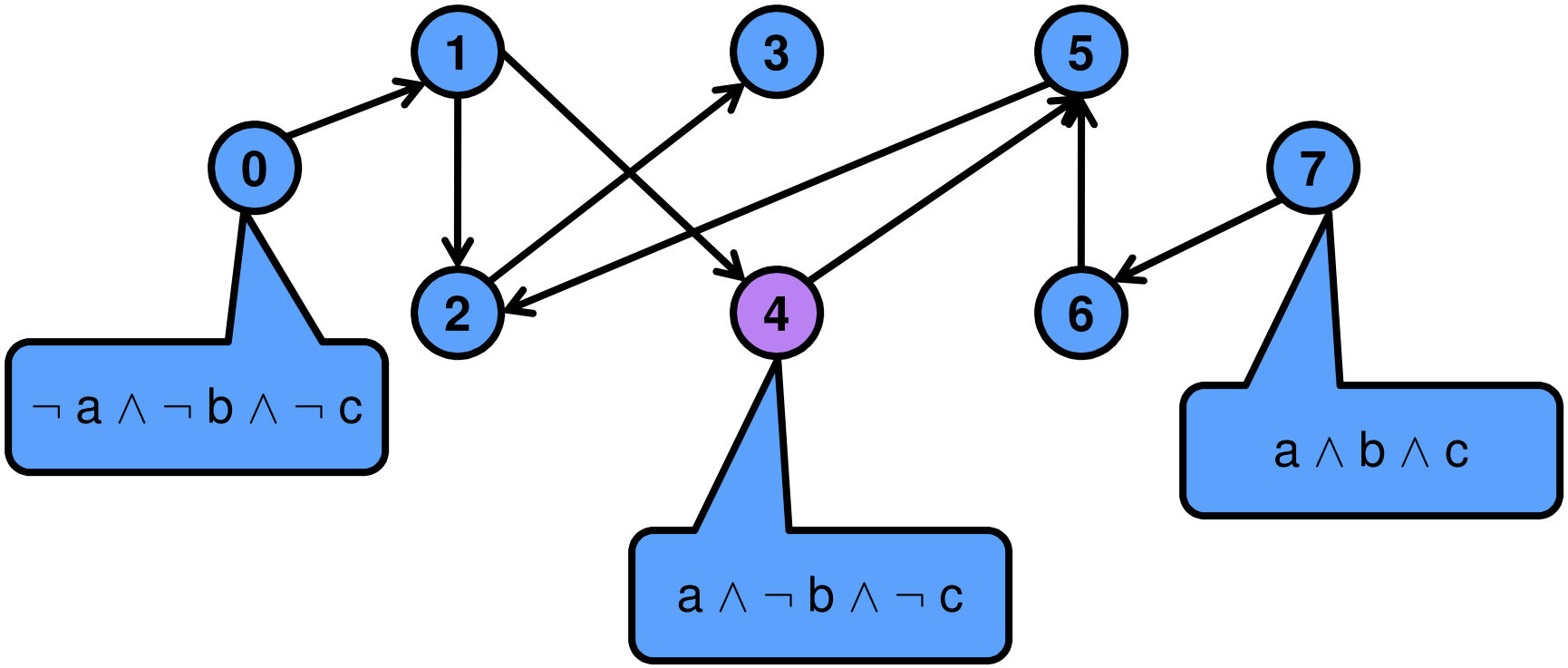
But what if the graph has trillions of nodes?

Graph Reachability



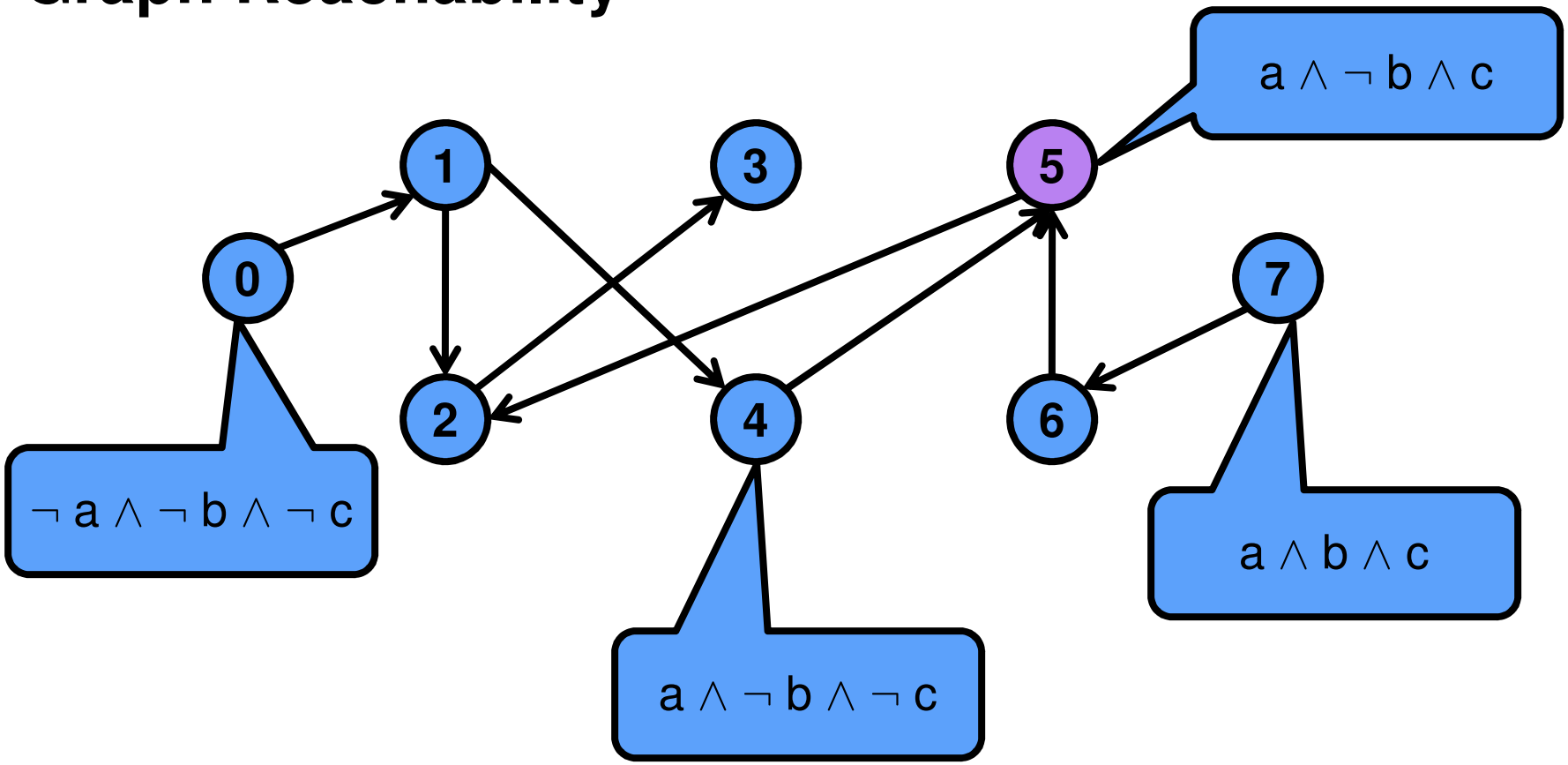
Use three Boolean variables (a,b,c) to encode each node?

Graph Reachability



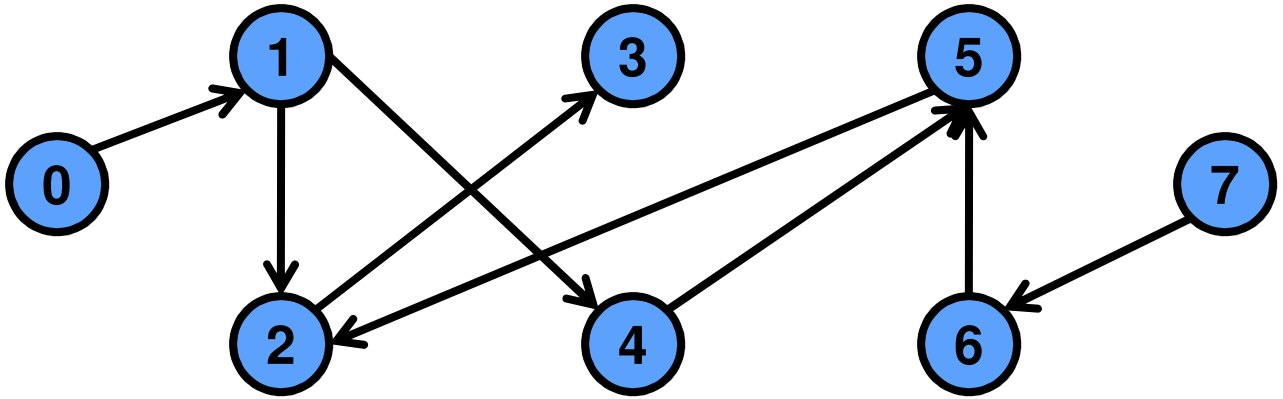
Use three Boolean variables (a,b,c) to encode each node?

Graph Reachability



Use three Boolean variables (a,b,c) to encode each node?

Graph Reachability

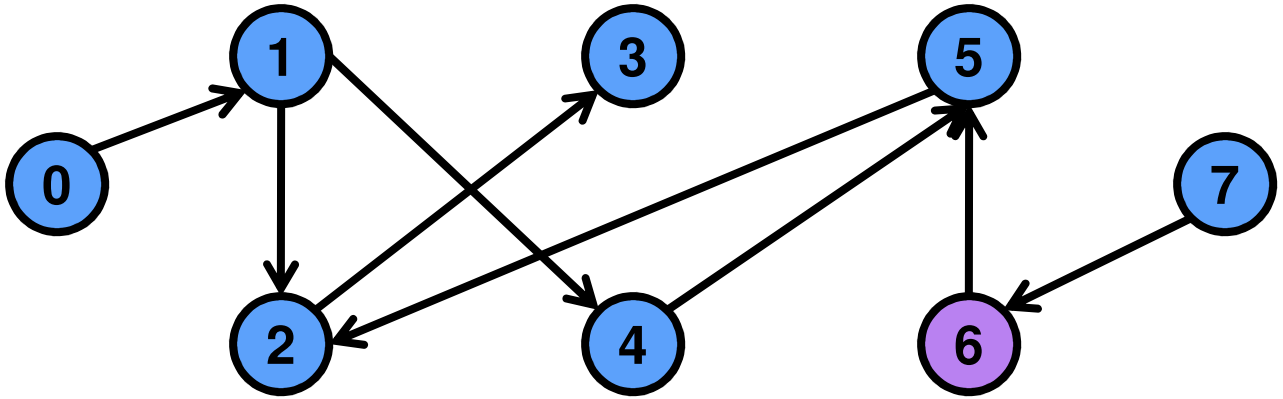


$a \wedge b \wedge \neg c = ?$

Key Idea 1: Every Boolean formula represents a set of nodes!

The nodes whose encodings satisfy the formula.

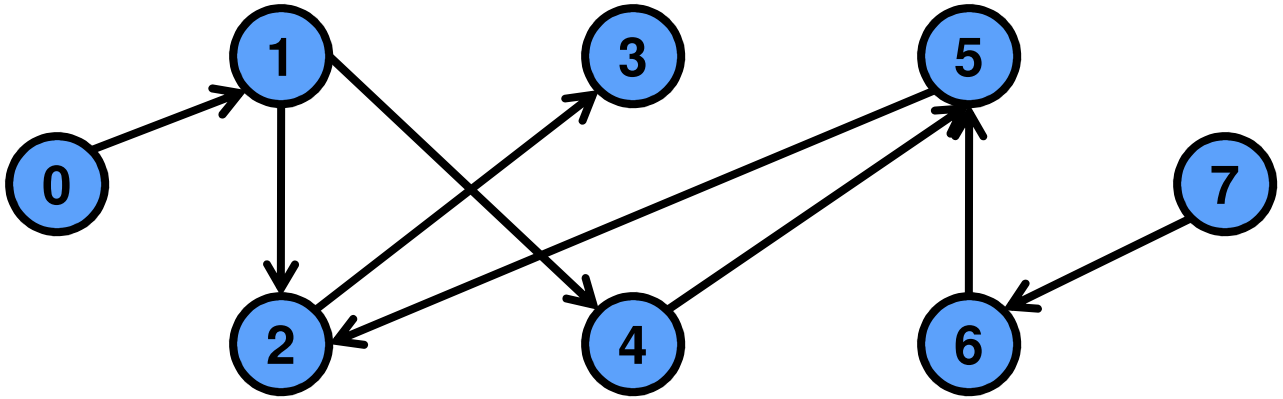
Graph Reachability



$a \wedge b \wedge \neg c = \{6\}$

Key Idea 1: Every Boolean formula represents a set of nodes!

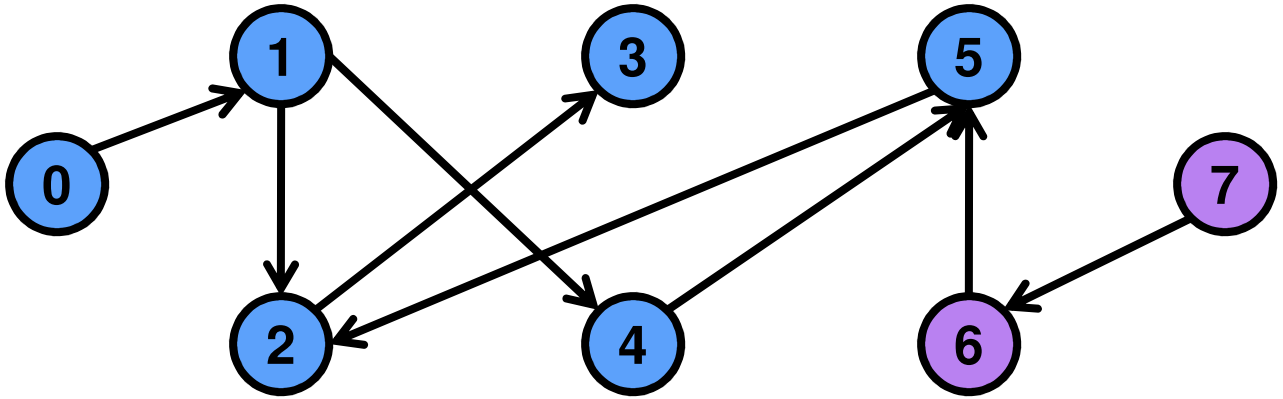
Graph Reachability



$a \wedge b = ?$

Key Idea 1: Every Boolean formula represents a set of nodes!

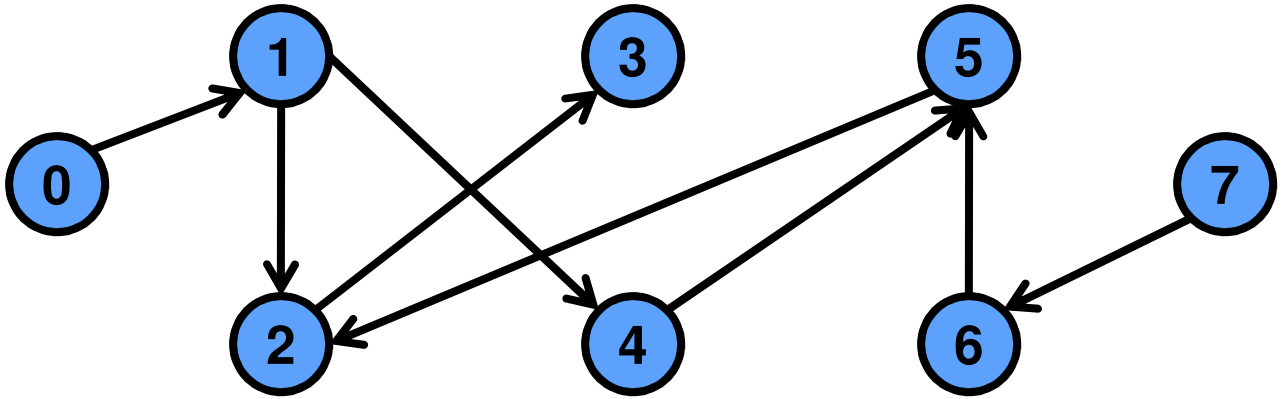
Graph Reachability



$a \wedge b = \{6,7\}$

Key Idea 1: Every Boolean formula represents a set of nodes!

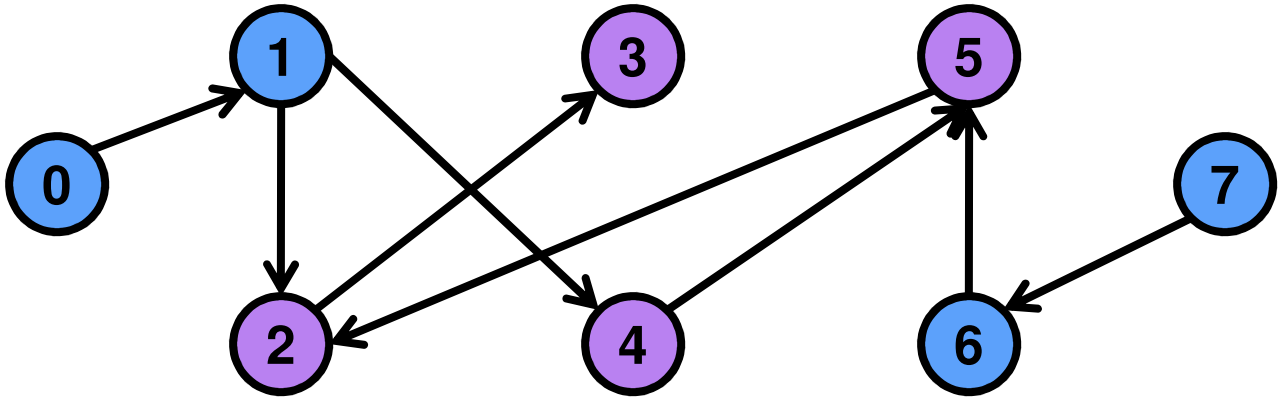
Graph Reachability



$a \oplus b = ?$

Key Idea 1: Every Boolean formula represents a set of nodes!

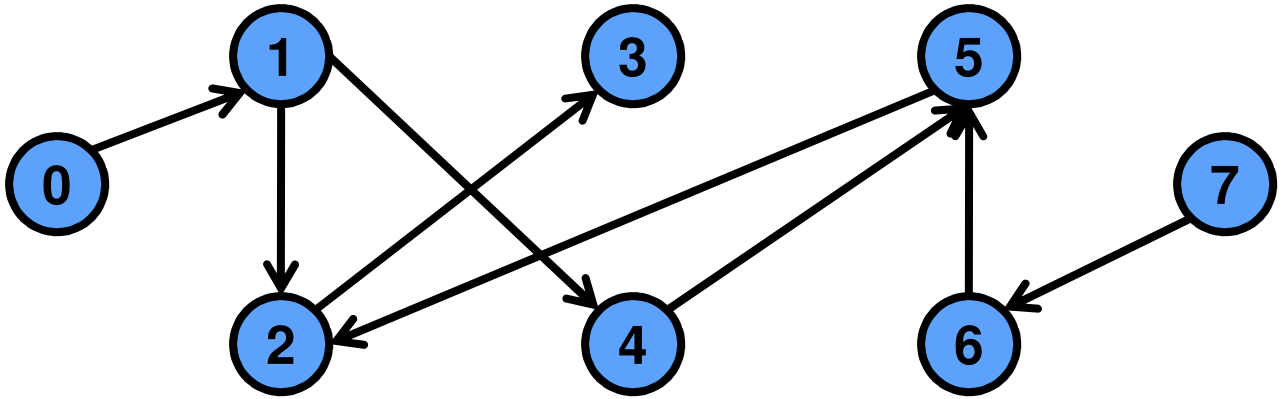
Graph Reachability



$a \oplus b = \{2,3,4,5\}$

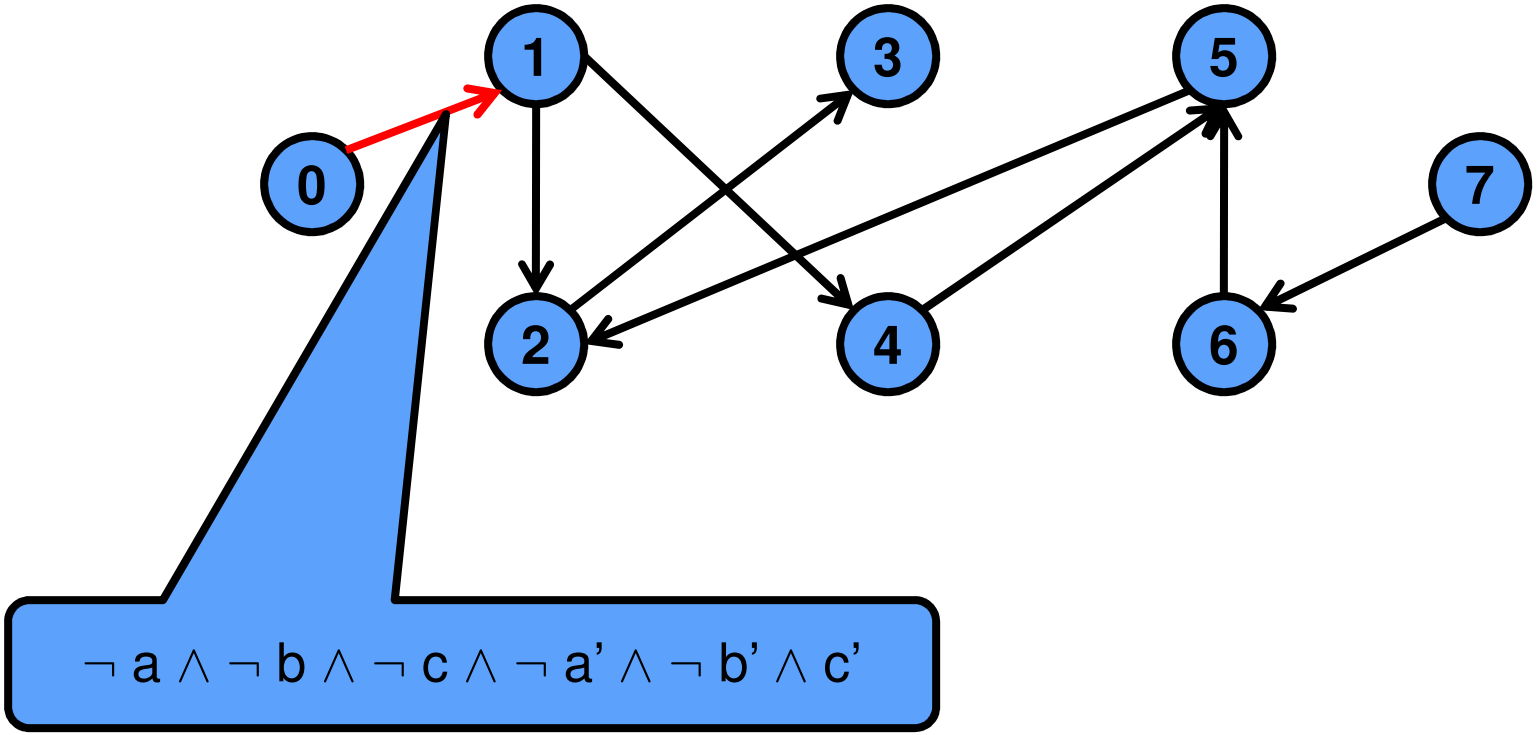
Key Idea 1: Every Boolean formula represents a set of nodes!

Graph Reachability



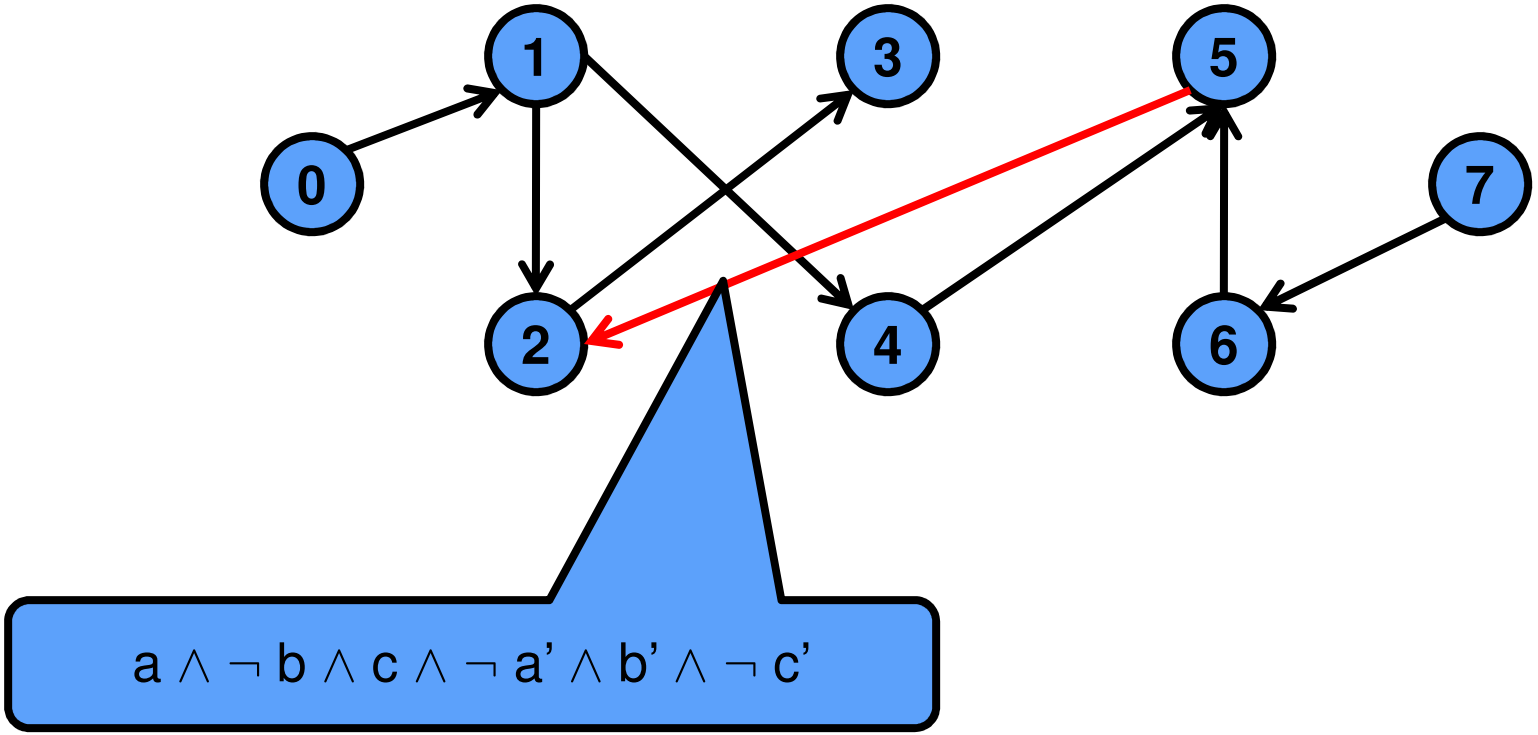
- Key Idea 2: Edges can also be represented by Boolean formulas
- An edge is just a pair of nodes
- Introduce three new variables: a', b', c'
- Formula Φ represents all pairs of nodes (n,n') that satisfy Φ when n is encoded using (a,b,c) and n' is encoded using (a',b',c')

Graph Reachability



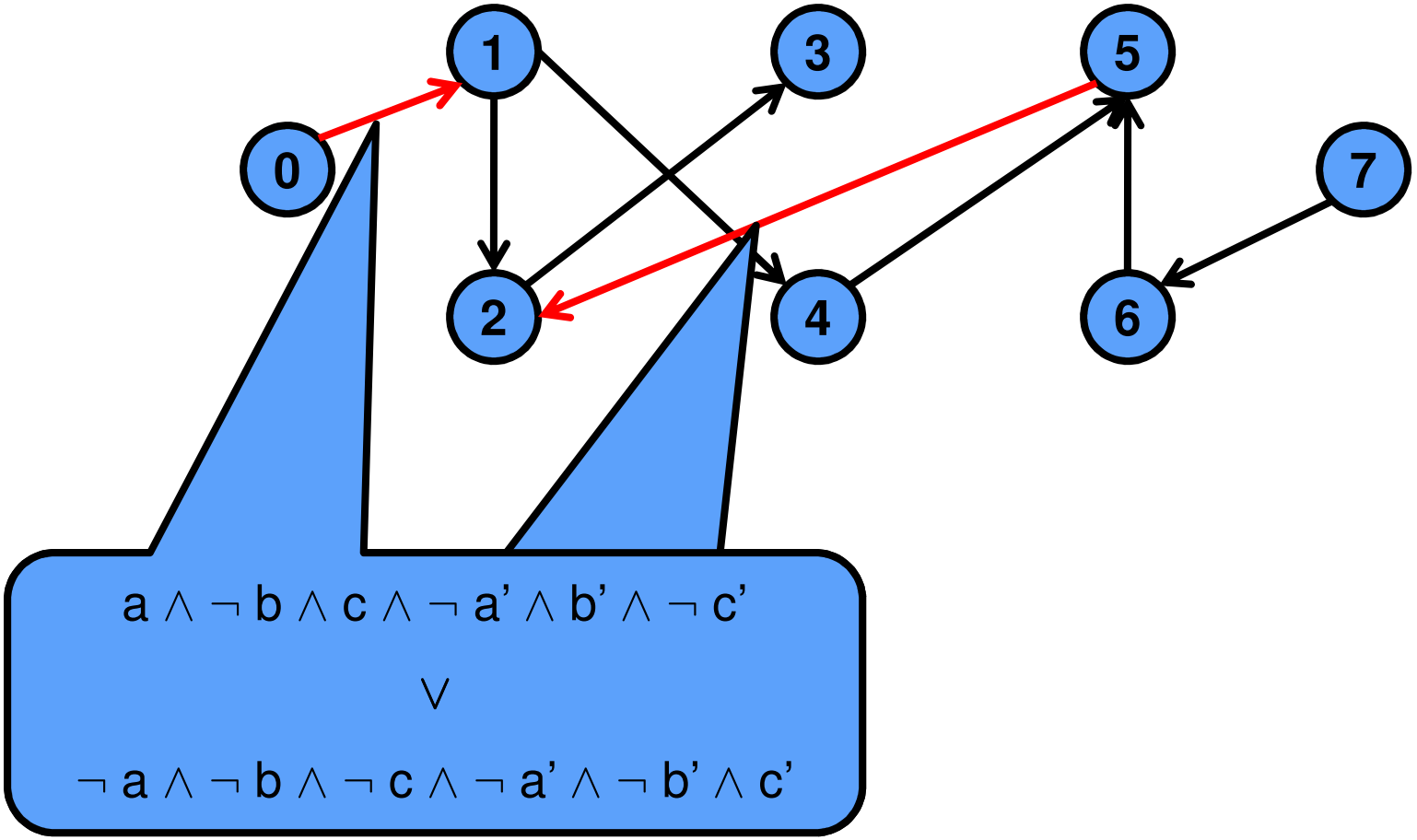
Key Idea 2: Edges can also be represented by Boolean formulas

Graph Reachability



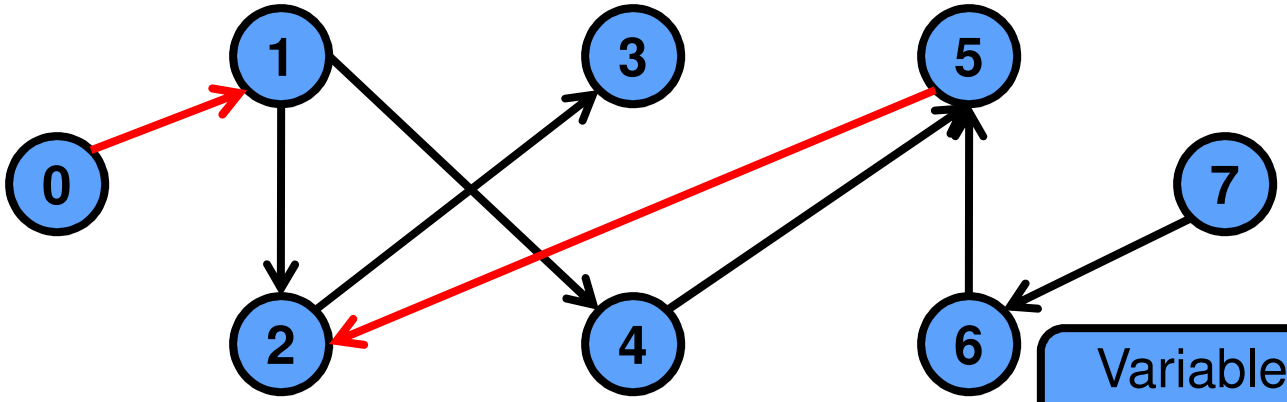
Key Idea 2: Edges can also be represented by Boolean formulas

Graph Reachability



Key Idea 2: Edges can also be represented by Boolean formulas

Graph Reachability



Variable renaming : replace a' with a

$$\text{Image}(S,R) =$$

$$(\exists\, a,b,c \,.\, (S \wedge R))\, [\, a \setminus a',\ b \setminus b',\ c \setminus c' \,]$$

Key Idea 3: Given the BDD for a set of nodes S, and the BDD for the set of all edges R, the BDD for all the nodes that are adjacent to S can be computed using the BDD operations

Graph Reachability Algorithm

```
S = BDD for initial set of nodes;
R = BDD for all the edges of the graph;

while (true) {
   I = Image(S,R); //compute adjacent nodes to S
   if (And(Not(S),I) == False) //no new nodes found
      break;
   S = Or(S,I); //add newly discovered nodes to result
}

return S;
```

Symbolic Model Checking. Has been done for graphs with 10^{20} nodes.

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