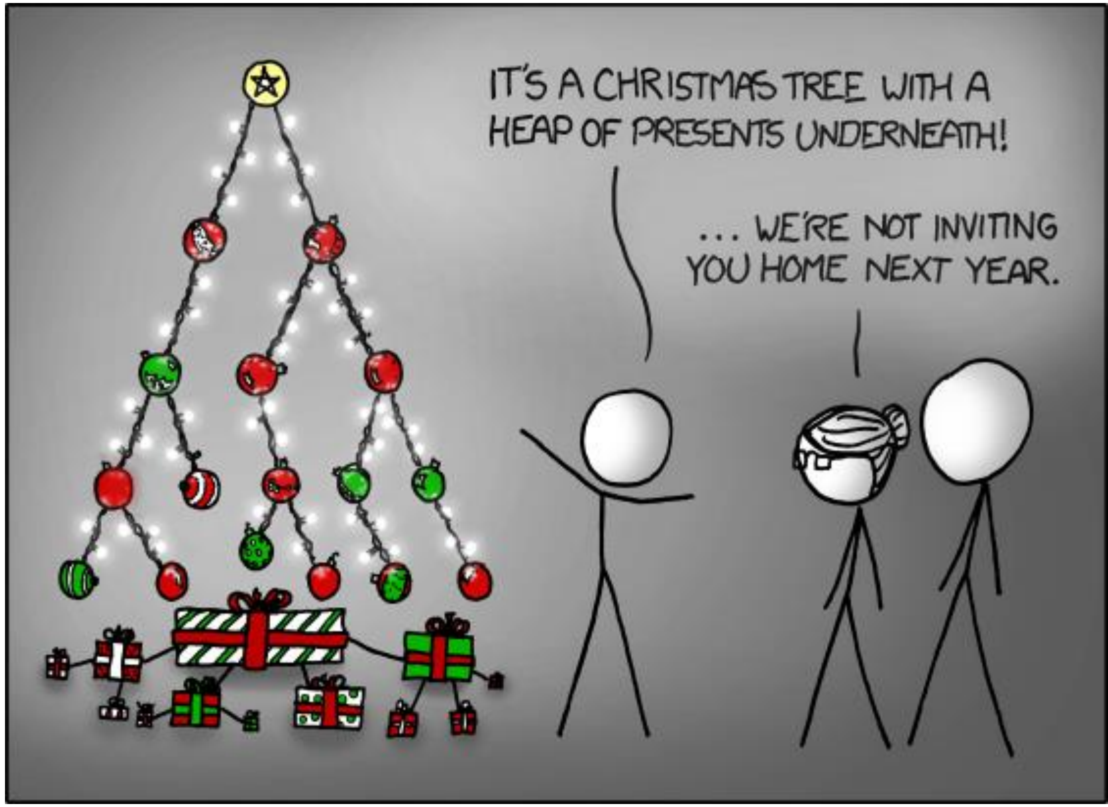
Logic and Computability

Binary Decision Diagrams (BDDs)



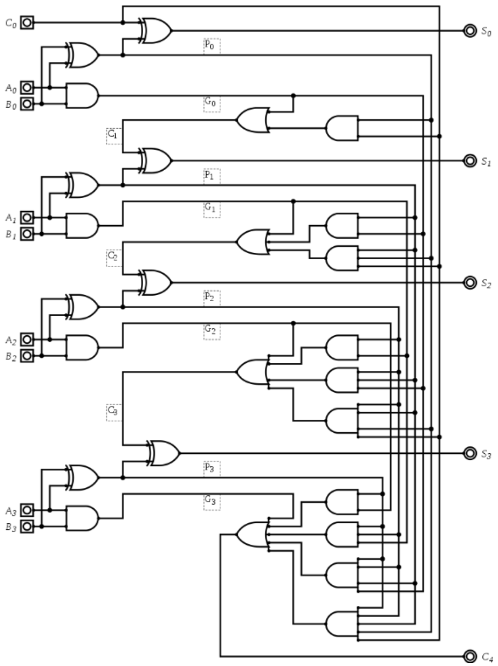
Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at

Motivation – BDDs

- Formulas are huge
  - E.g., when presenting circuit as formula
  - Hundreds of thousands of variables, millions of clauses….
- We need efficient methods
  - to store, to manipulate formula, and to decide formulas

Some Circuit



Automatically Translate

Nr variables **Nr clauses**

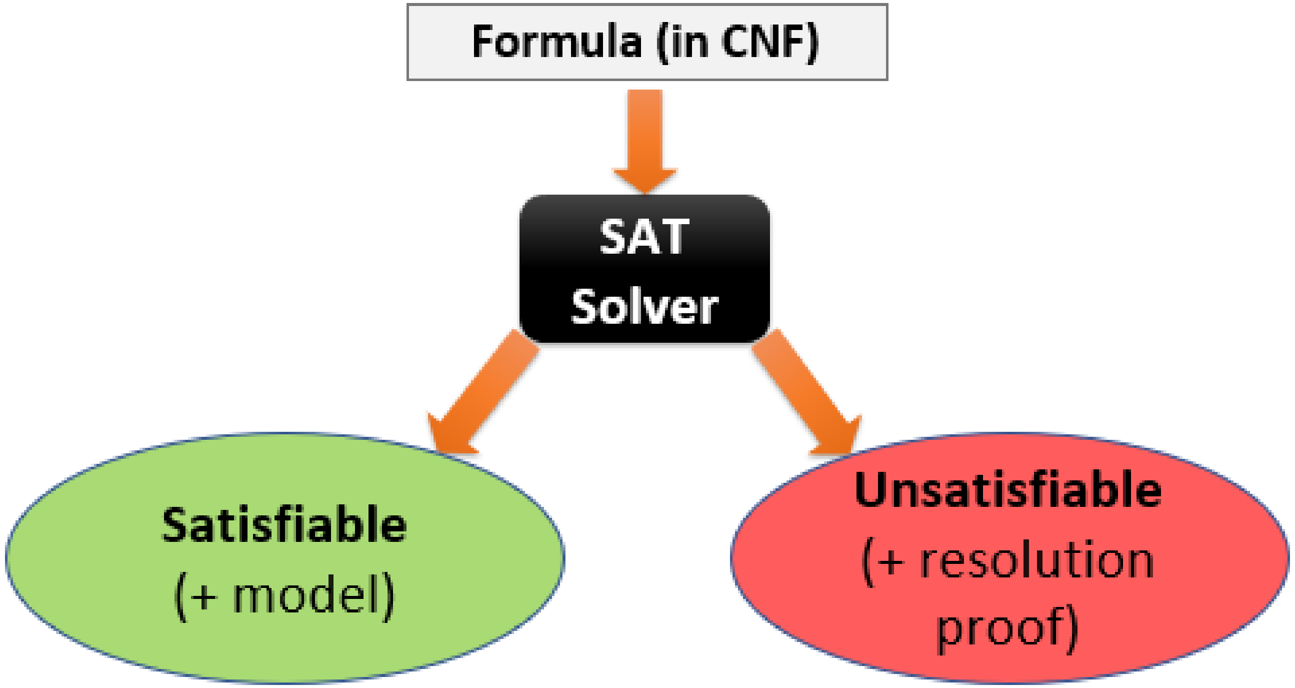
```
p cnf 51639 368352
-1 7 0
-1 6 0
-1 5 0
-1 -4 0
-1 3 0
-1 2 0
-1 -8 0
-9 15 0
-9 14 0
-9 13 0
-9 -12 0
-9 11 0
-9 10 0
-9 -16 0
-17 23 0
-17 22 0
```

$(\neg x_1 \vee x_7) \wedge$

$(\neg x_1 \vee x_6) \wedge \cdots$

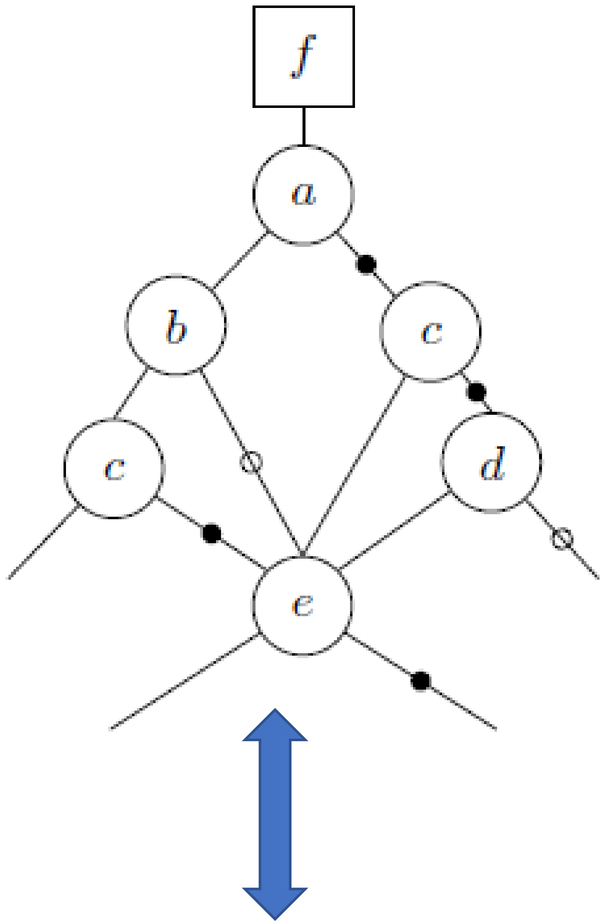
Motivation – BDDs

- Last week
  - SAT Solvers
  - DPLL - Efficient algorithm to decide huge formulas



Motivation – BDDs

- This week- BDDs
  - Graph-based data structure
  - To represent and manipulate formulas

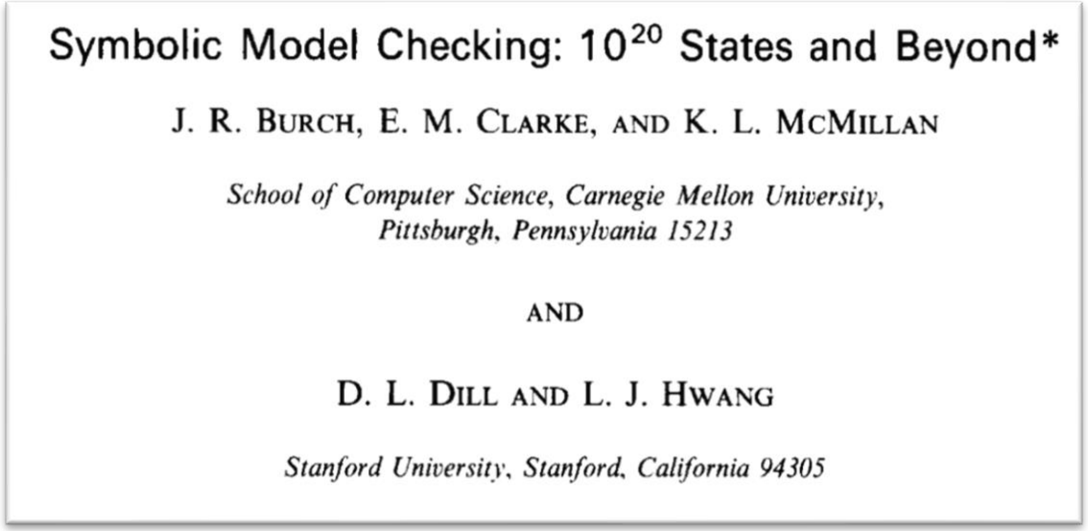


$$f = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge e) \vee (a \wedge b \wedge \neg c \wedge \neg e) \vee (\neg a \wedge \neg c \wedge d) \vee (\neg a \wedge c \wedge \neg e) \vee (\neg a \wedge \neg c \wedge d \wedge e)$$

Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, L. J. Hwang:
Symbolic Model Checking: 10^20 States and Beyond. LICS 1990: 428-439

Motivation – BDDs

- This week- BDDs
  - Graph-based data structure
  - To represent and manipulate formulas
  - E.g., Used in hardware and software verification tools



Symbolic Model Checking: 10^{20} States and Beyond*

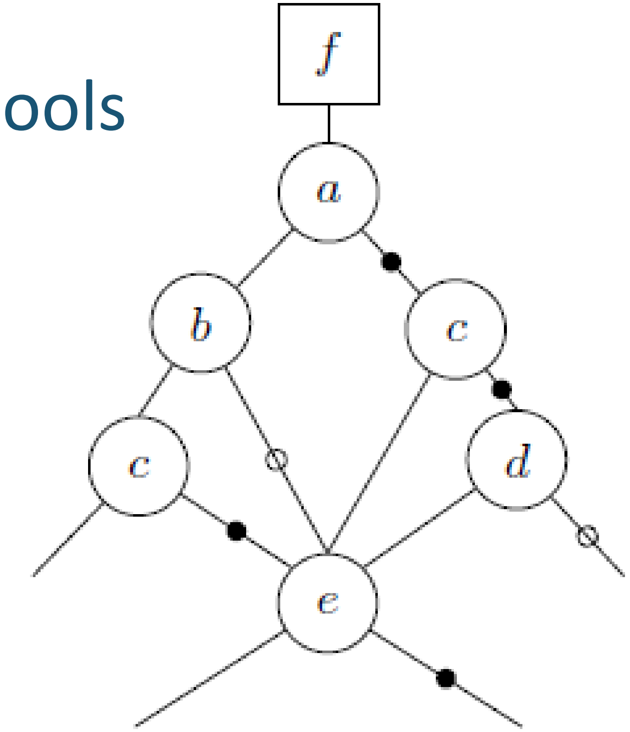
J. R. BURCH, E. M. CLARKE, AND K. L. MCMILLAN

School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

AND

D. L. DILL AND L. J. HWANG

Stanford University, Stanford, California 94305

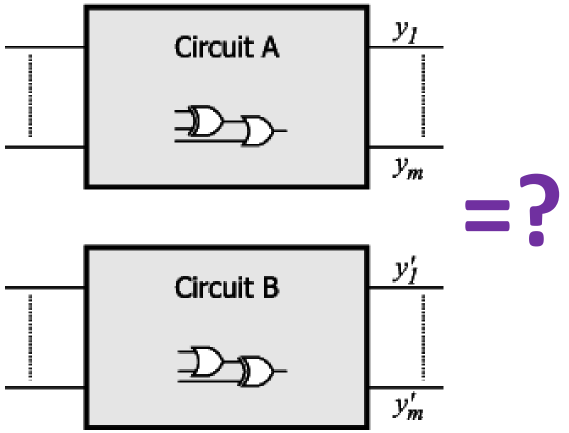


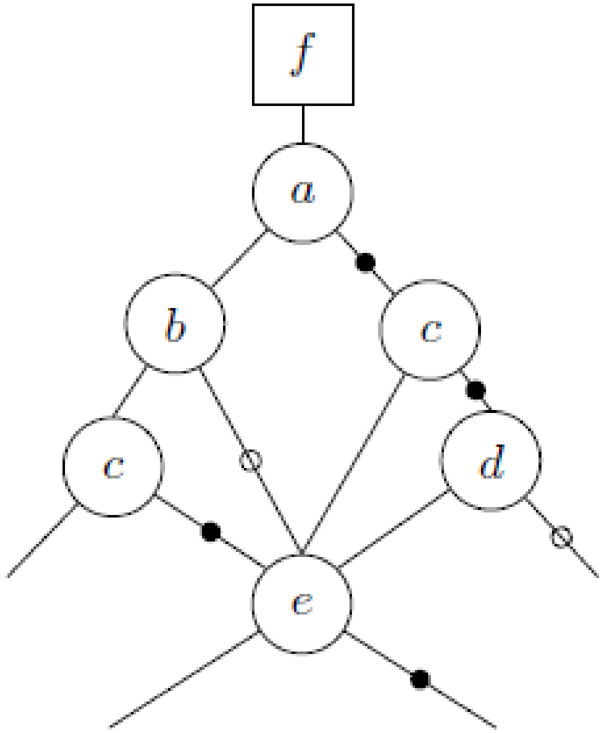
Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, L. J. Hwang: Symbolic Model Checking: 10^20 States and Beyond. LICS 1990: 428-439

Motivation – BDDs

- Advantages:
  - Efficient Manipulation
    - Boolean Operations
  - Often small representation
  - Canonical (unique) representation
    - If two formulas are equivalent, then their BDD representations are equivalent

Application:
Circuit Equivalence Checking





Outline

- What are Binary Decision Diagrams (BDDs)?
  - Intuitive Explanation
  - Formal Definition
  - Reduced-Ordered BDDs
    - for us, a BDD is always reduced and ordered

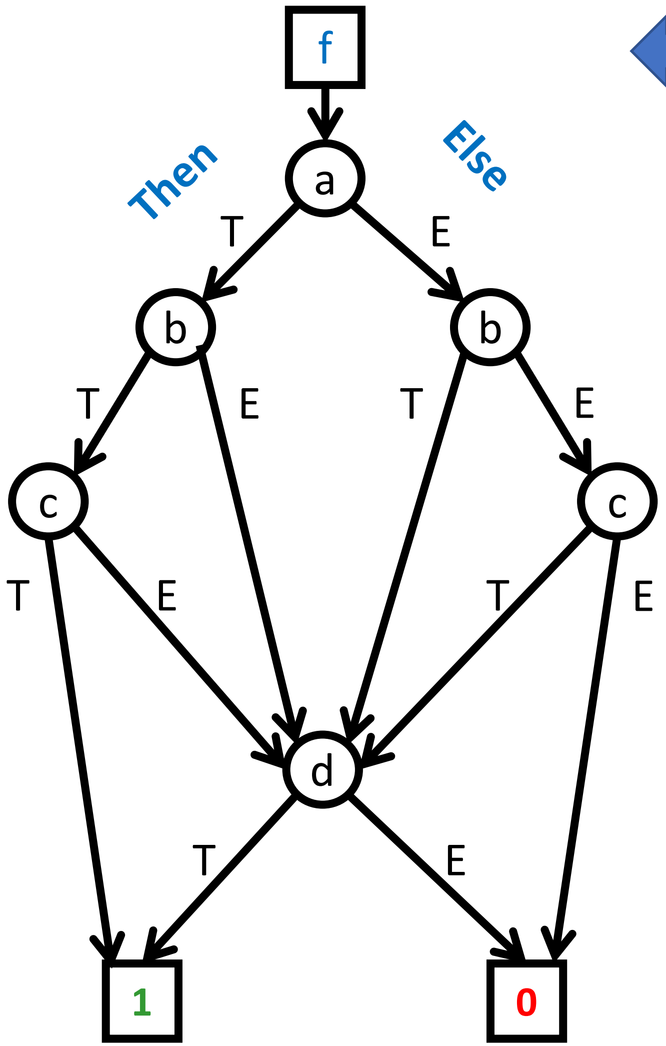
- **Represent a formula in propositional logic as BDD**
- **From the BDD, derive the formula that is represented by a BDD**

Learning Outcomes

After this lecture…

1. students can define and explain BDDs.
   - BDD = reduced and ordered binary decision diagram
   - Define and explain its elements and their meaning
2. students can represent a formula in propositional logic as BDD.
3. students can derive the formula that is represented by a BDD.
4. students can state properties of BDDs.
   - advantages, disadvantages

Binary Decision Diagram (BDD) → Initial Representation

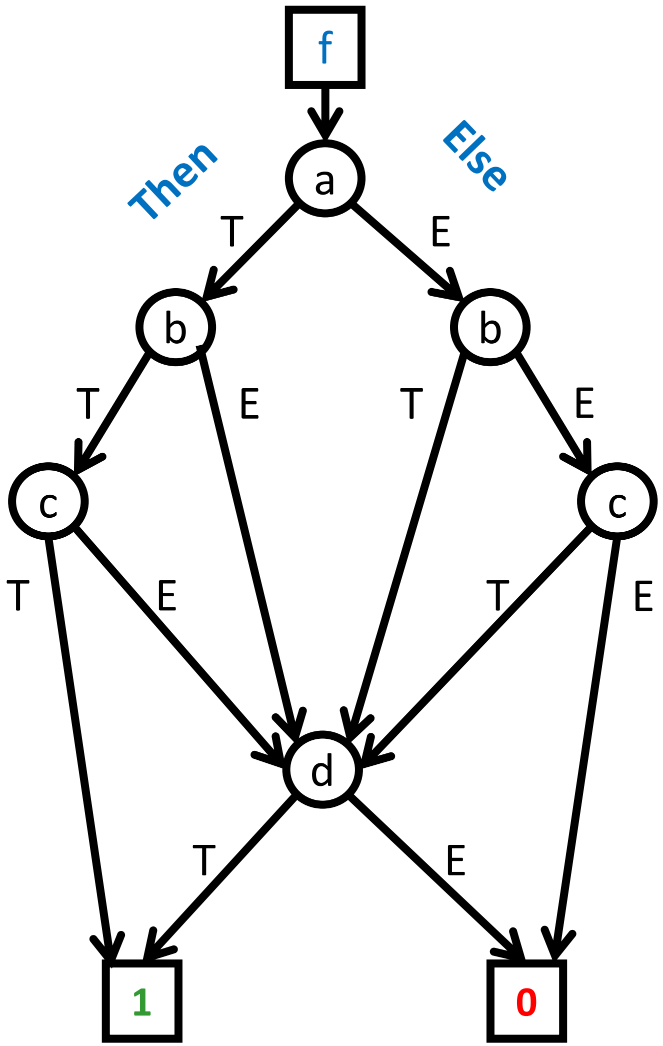


Function node:
Represents formula f

Internal nodes:
Variables of f

Terminal nodes:
Represent truth values

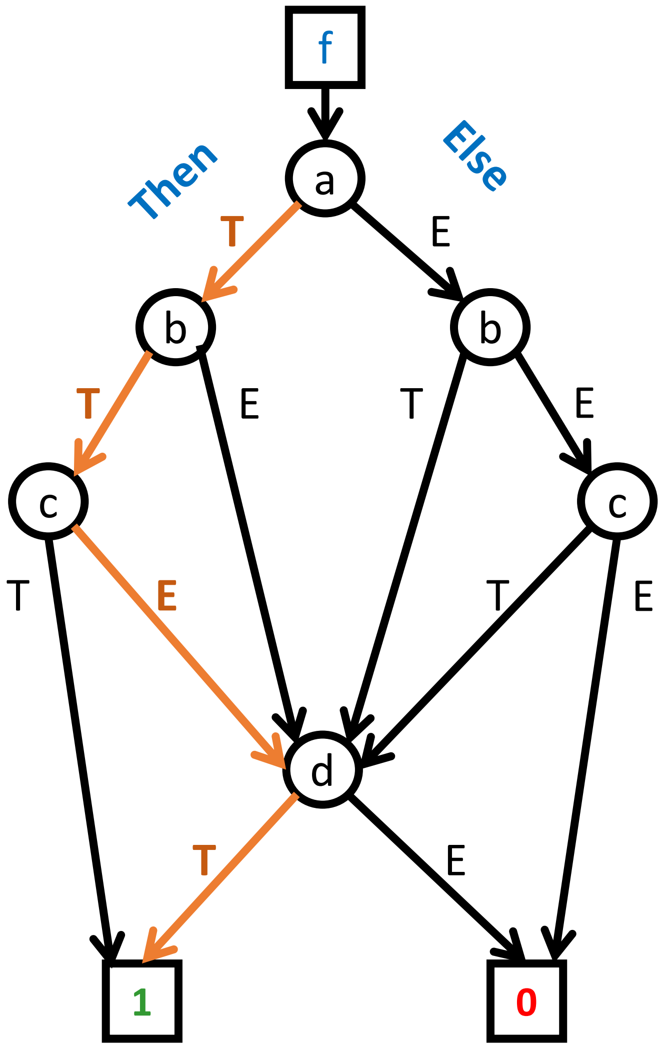
Binary Decision Diagram (BDD)



Given $M := \{a = T, b = T, c = T, d = T\}$

Does it hold that $M \vDash f$?

Binary Decision Diagram (BDD)

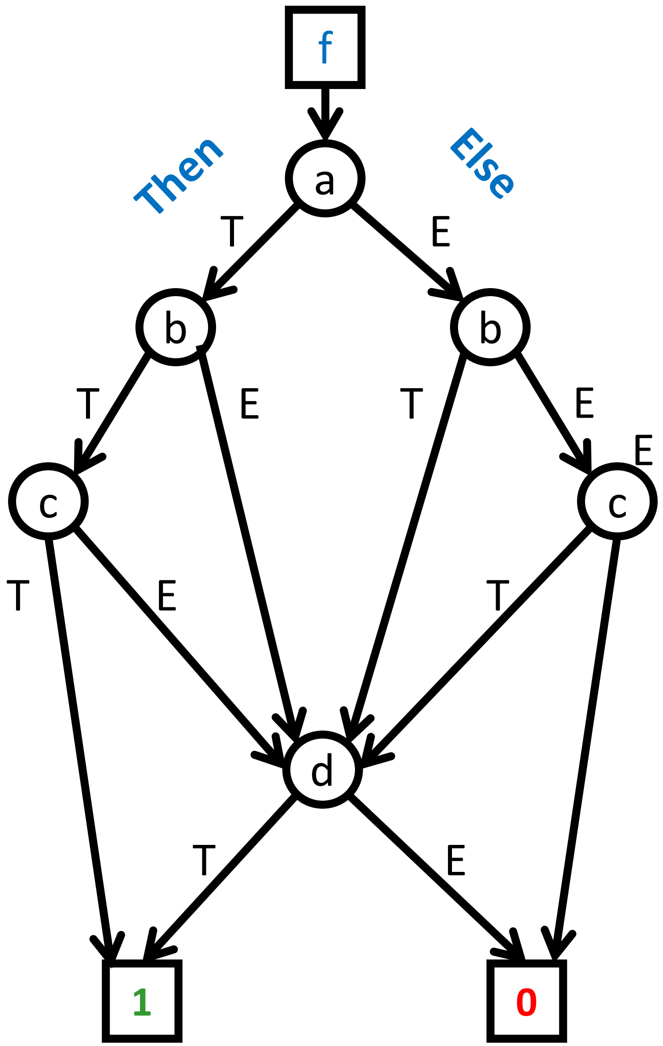


Given $M := \{a = T, b = T, c = T, d = T\}$

Does it hold that M $\vDash f$?

***M* is a satisfying assignment**

Binary Decision Diagram (BDD)

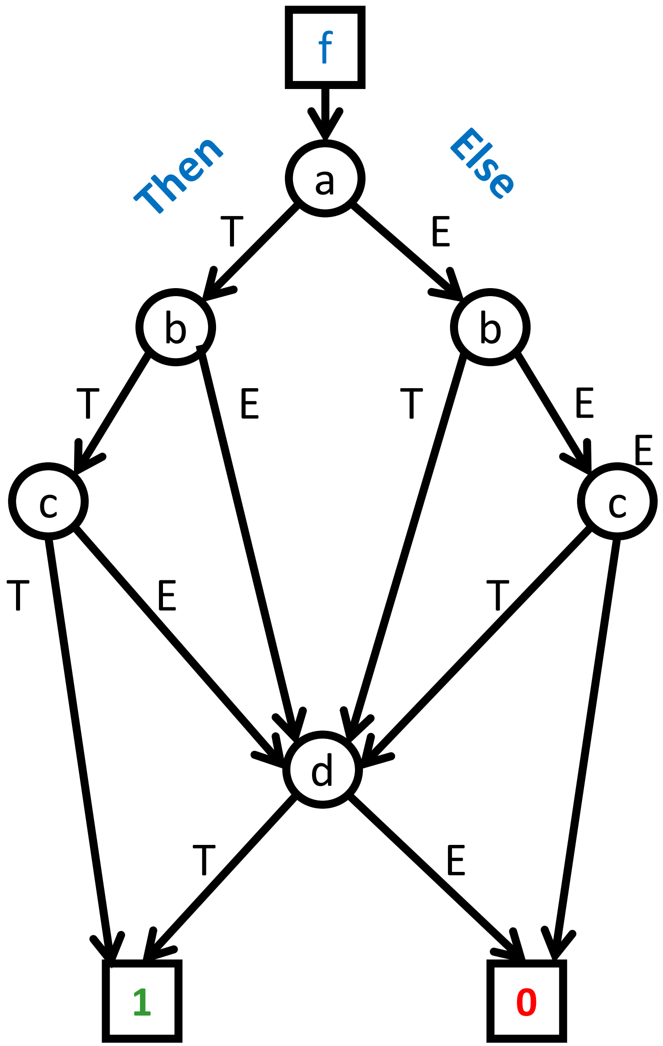


$M \vDash f$ iff its path in BDDs ends in terminal node **1**

$M \nvDash f$ iff its path in BDDs ends in terminal node **0**

How can we find the formula f that is represented by this BDD?

Binary Decision Diagram (BDD)



$M \models f$ iff its path in BDDs ends in terminal node 1

$M \not\models f$ iff its path in BDDs ends in terminal node 0

How can we find the formula f that is represented by this BDD?

Represent f in DNF by enumerating all paths (models) that end in 1

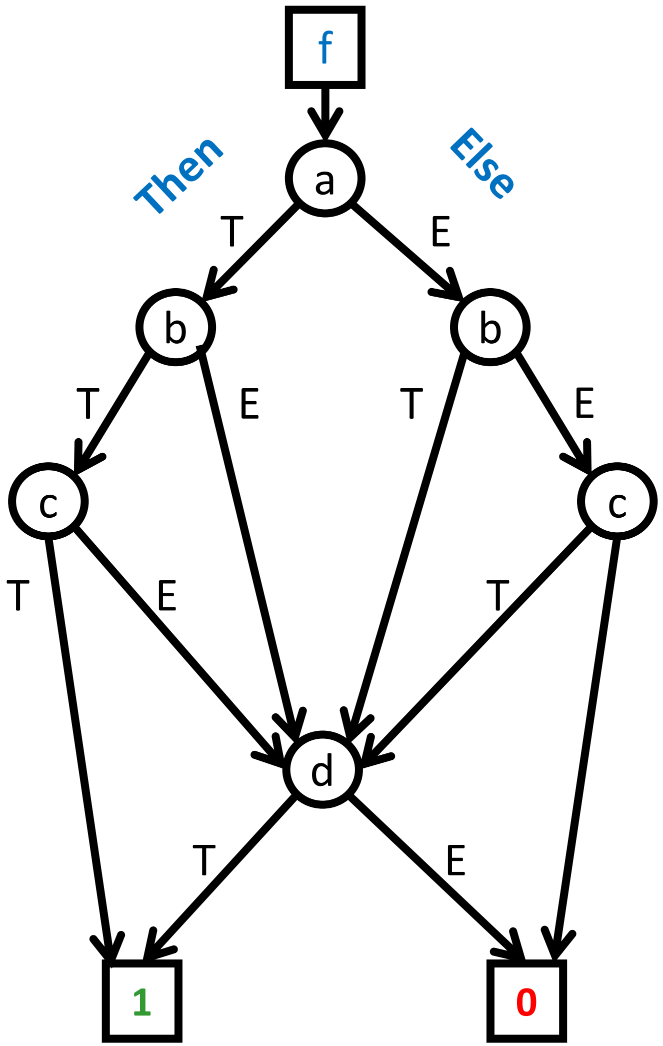
Or

Exclude all paths that end in 0

Binary Decision Diagram (BDD)

f is in disjunctive normal form (DNF) if it is a **disjunction** of **conjunctions**.

$$f = (a \wedge b) \vee (\neg a \vee c)$$

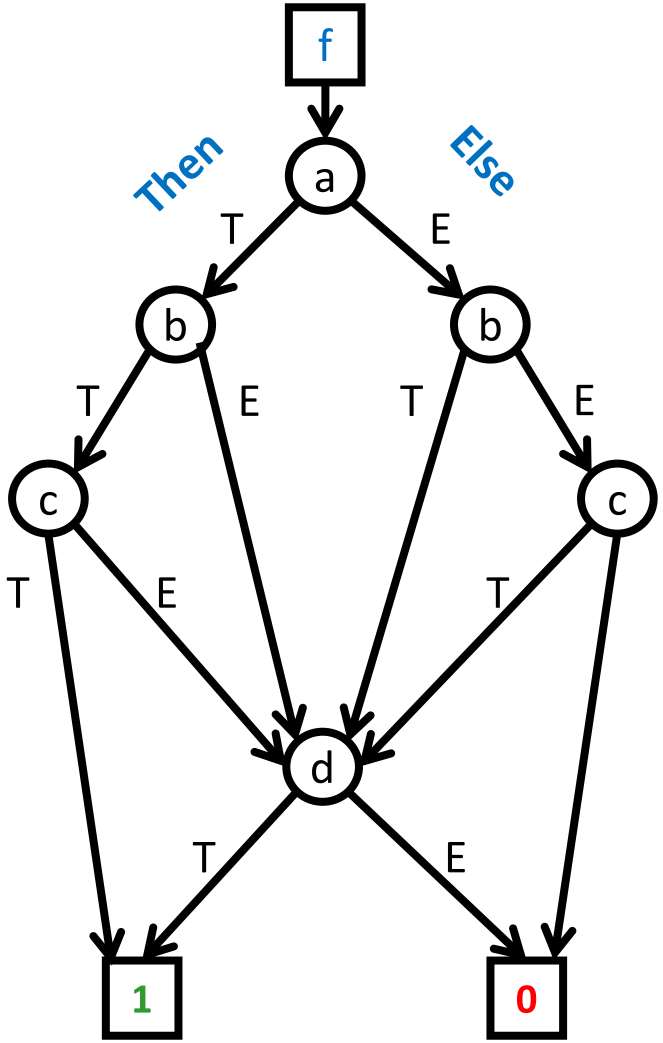


Represent f in DNF by enumerating all paths (models) that end in 1

Binary Decision Diagram (BDD)

f is in disjunctive normal form (DNF) if it is a **disjunction** of **conjunctions**.

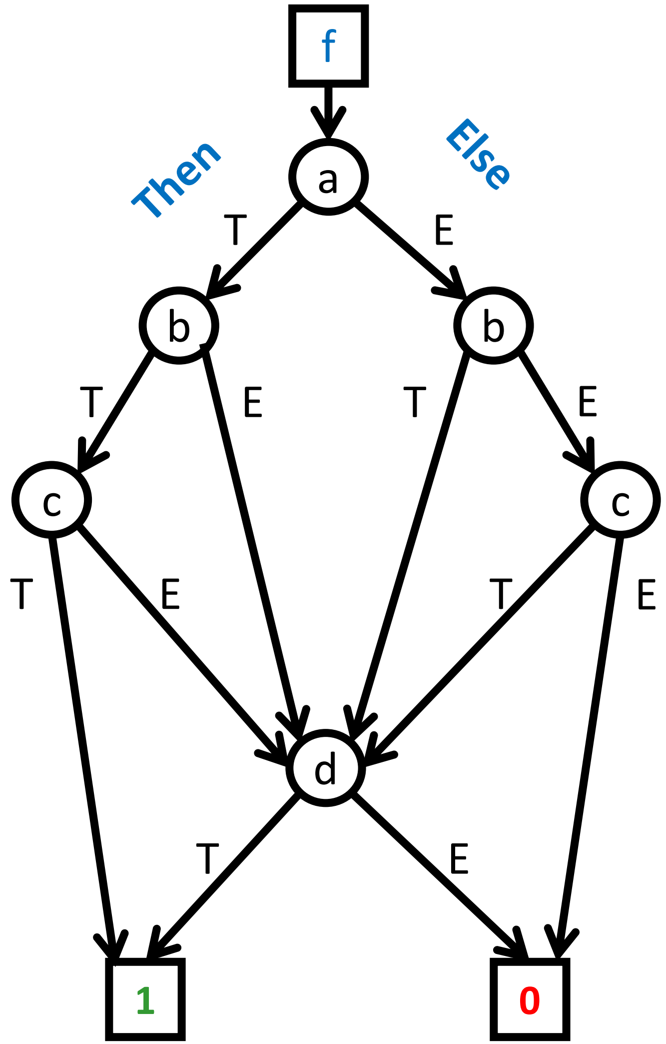
$$f = (a \land b) \lor (\neg a \lor c)$$

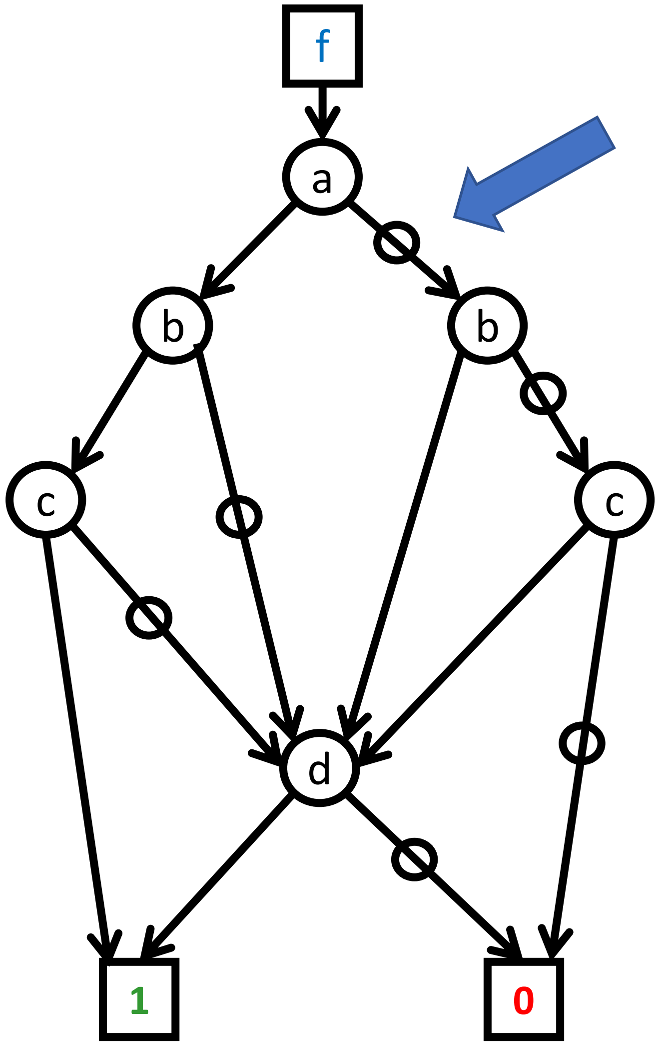


Represent f in DNF by enumerating all paths (models) that end in 1

$$f := (a \land b \land c) \lor (a \land b \land \neg c \land d) \lor (a \land \neg b \land d) \lor (\neg a \land b \land d) \lor (\neg a \land \neg b \land \mathrm{c} \land d)$$

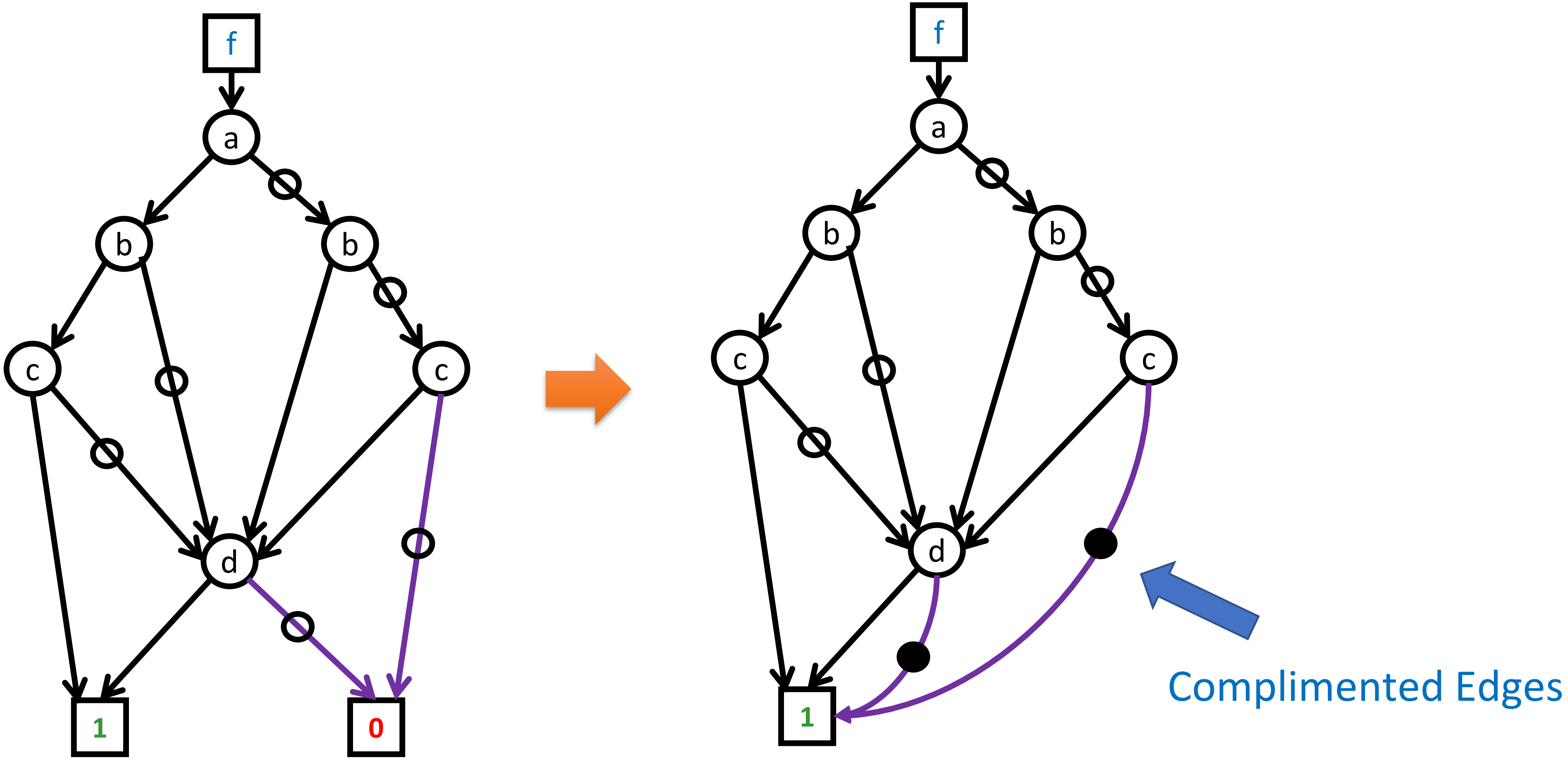
BDD - Representation





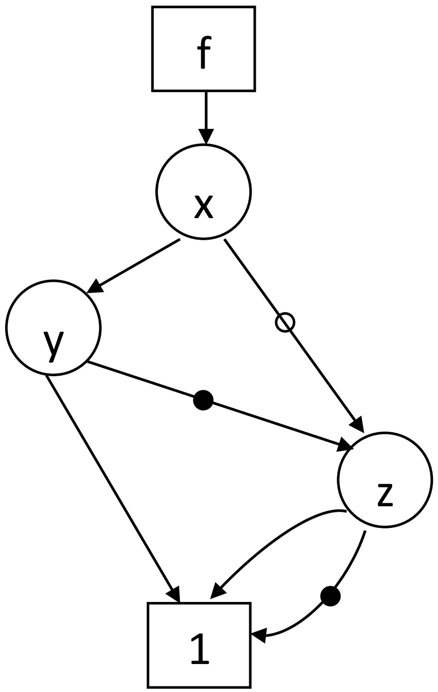
Else-edges are marked by circles

BDD - Representation



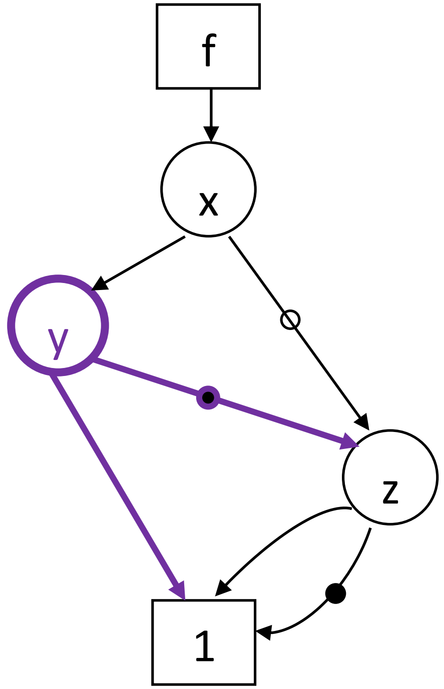
Definition of BDDs

- Directed Acyclic Graph
- $(V \cup f \cup \{\mathbf{1}\},\ E)$
  - Internal Nodes $v \in V$
  - Function Node f
    - Represents propositional formula f
    - May have additional nodes for subformulas
  - Terminal Node $\mathbf{1}$
    - Represents the truth value T
  - Edges E
    - “Complement” attribute



Definition of BDDs: Internal Node

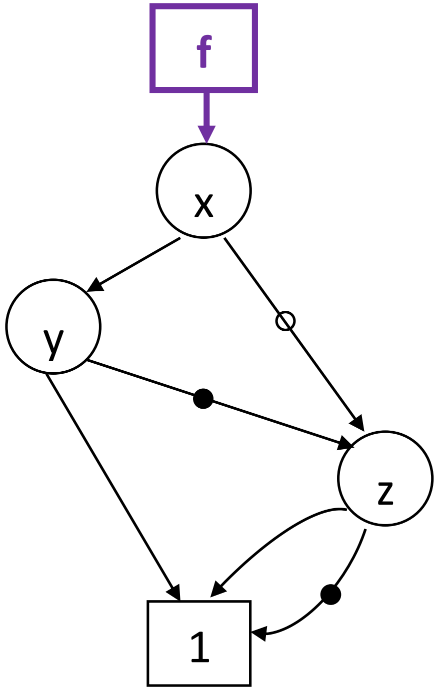
- Label $l(v) \in \{x_1, \dots, x_n\}$
  - Variables of f
- Out–degree: 2
  - Then-Edge T
  - Else-Edge E
    - Marked with (empty) circle
    - Can have complement attribute (full cycle)



Definition of BDDs: Function Node

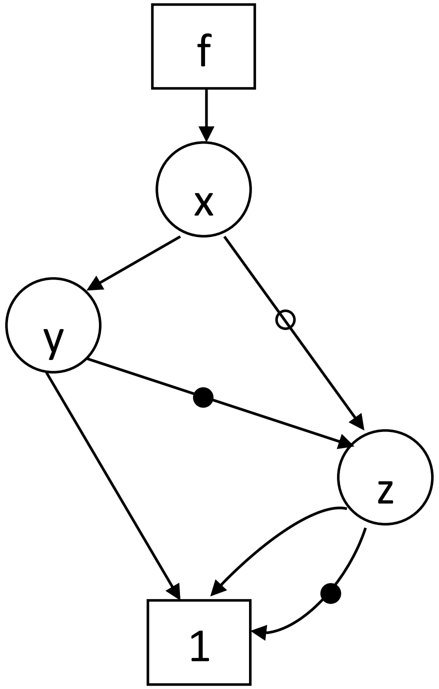
- Represents Boolean Formula f

- In-degree: 0
- Out-degree: 1
  - Edge can have complement attribute



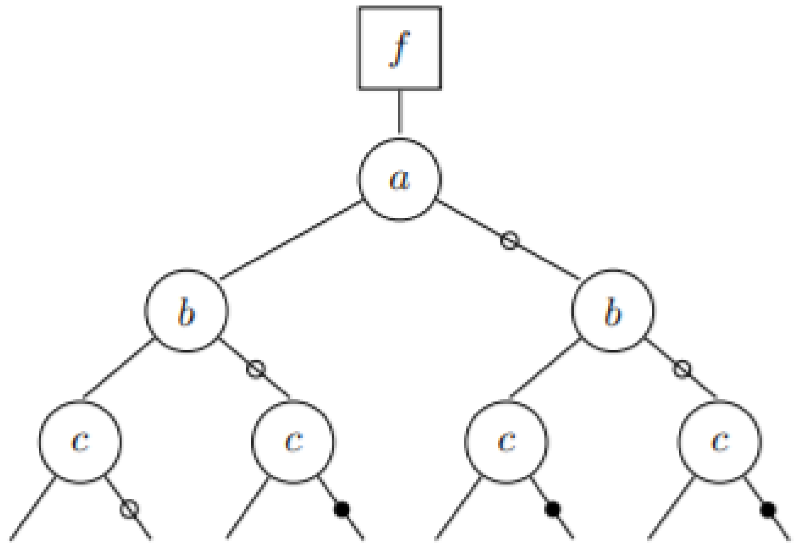
Definition of BDDs: Terminal Node

- Constant Function **True**
- Out-degree: 0

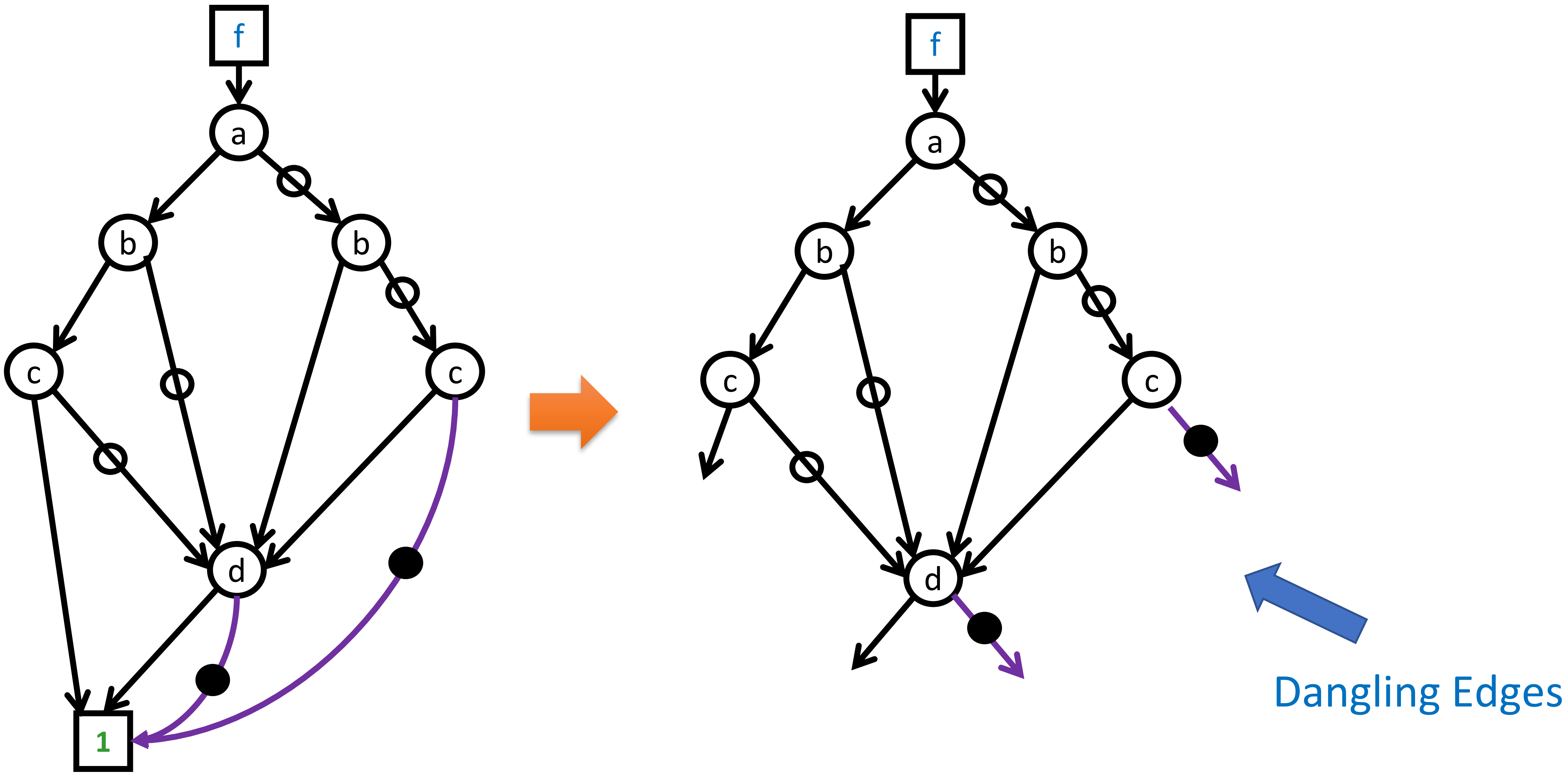


Size of a BDD

- **Worst case: exponential**
  - If each internal node has 2 Sub-BDDs, then BDD has $2^n - 1$ internal nodes
- Often: BDDs contain much redundancy
  - Obtain compact BDD by removing redundancies

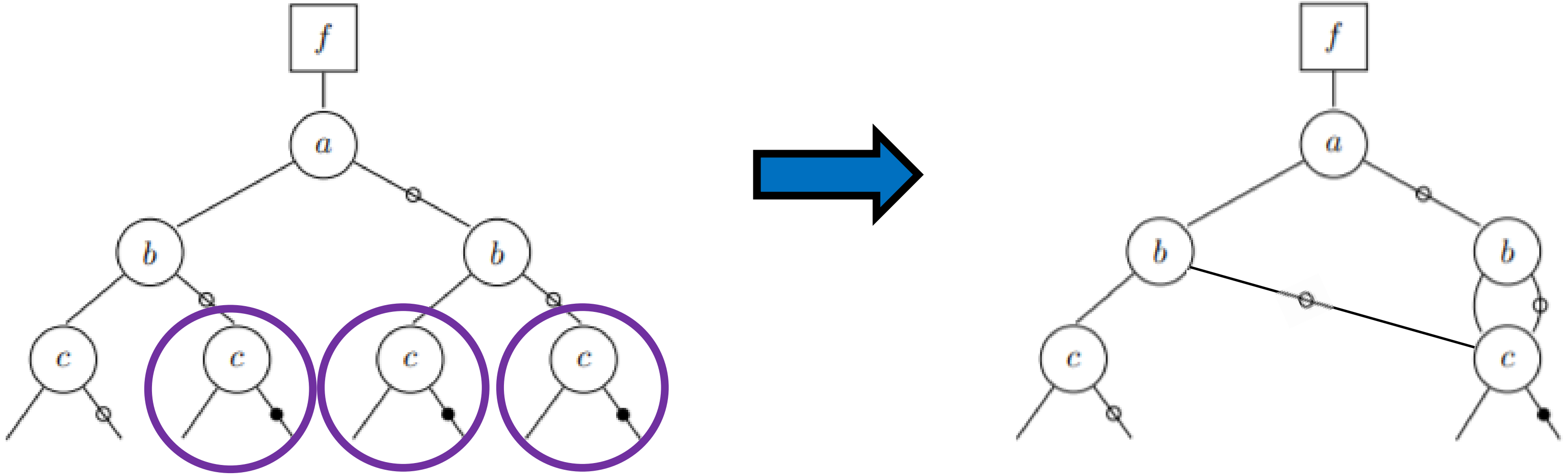


BDD – Representation



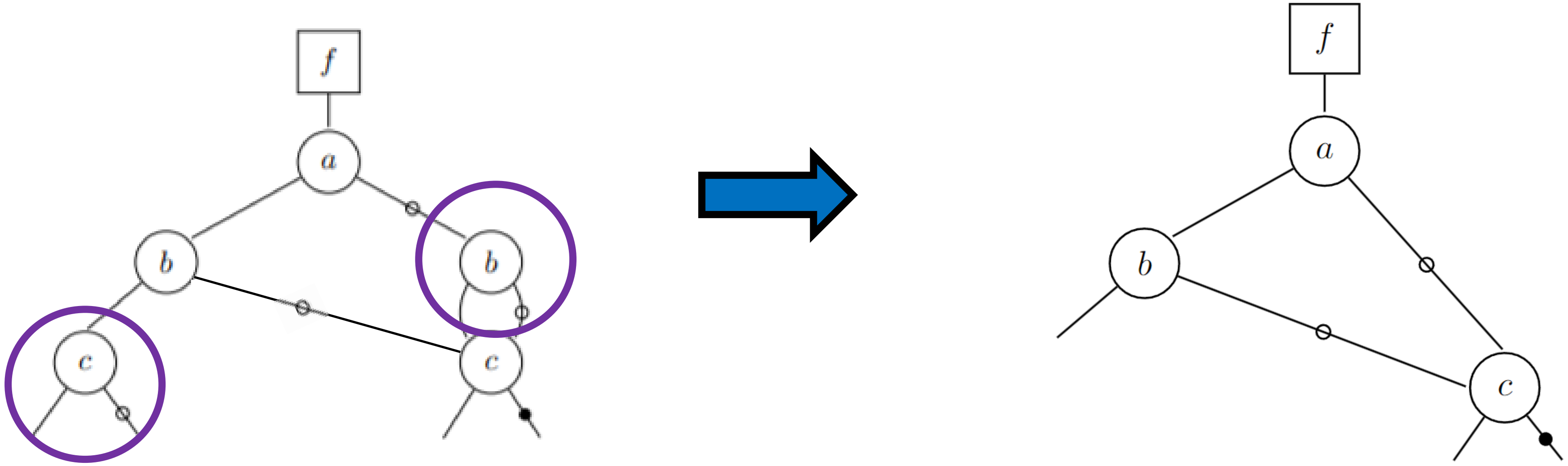
Reduced Ordered BDD

1. No duplicate sub-BDDs



Reduced Ordered BDD

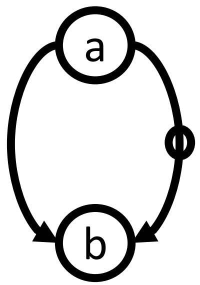
1. No duplicate sub-BDDs
2. No redundant nodes



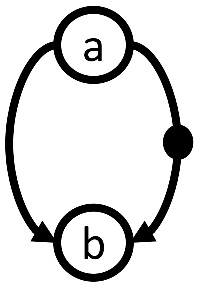
Reduced Ordered BDD

1. No duplicate sub-BDDs
2. No redundant nodes

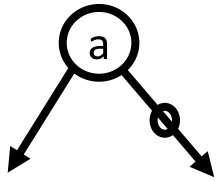
Redundant



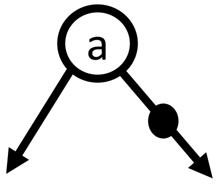
NOT Redundant



Redundant
(special case)



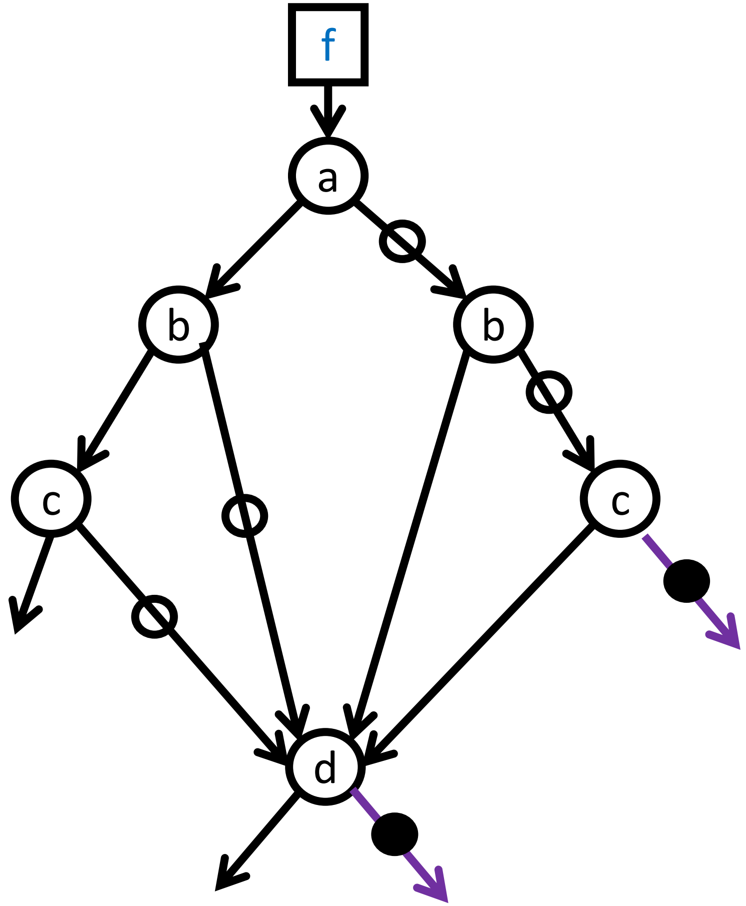
NOT Redundant
(special case)



Reduced Ordered BDD

1. No duplicate sub-BDDs
2. No redundant nodes

3. Ordering on the variables along any path
   - E.g., $a < b < c < \mathrm{d}$

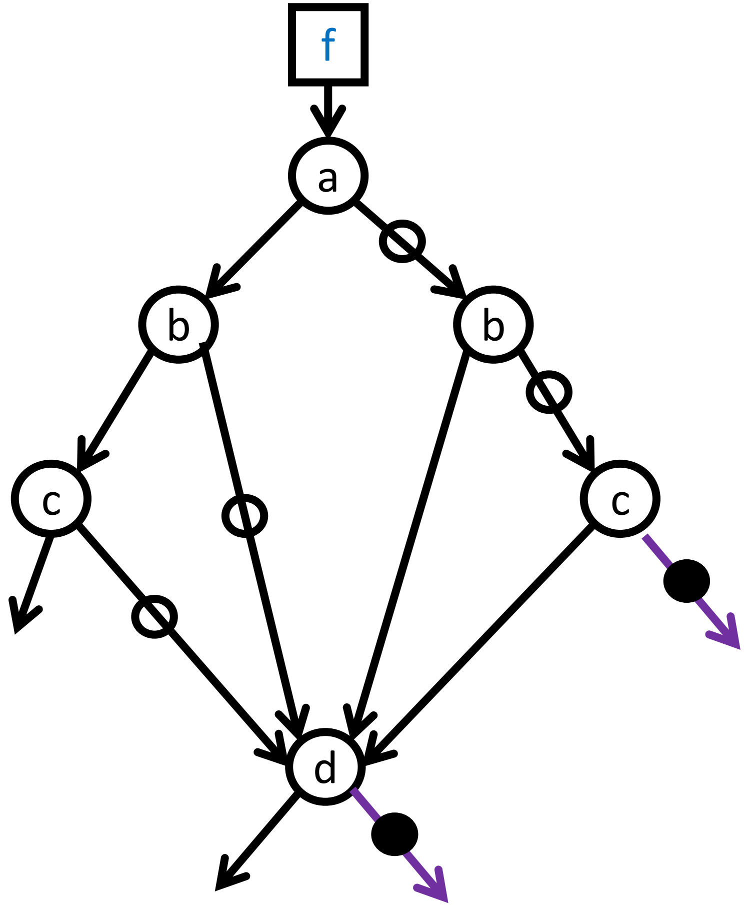


Reduced Ordered BDD

1. No duplicate sub-BDDs
2. No redundant nodes

3. Ordering on the variables along any path
   - E.g., $a < b < c < d$

A reduced and ordered BDD gives a **canonical** representation of a formula

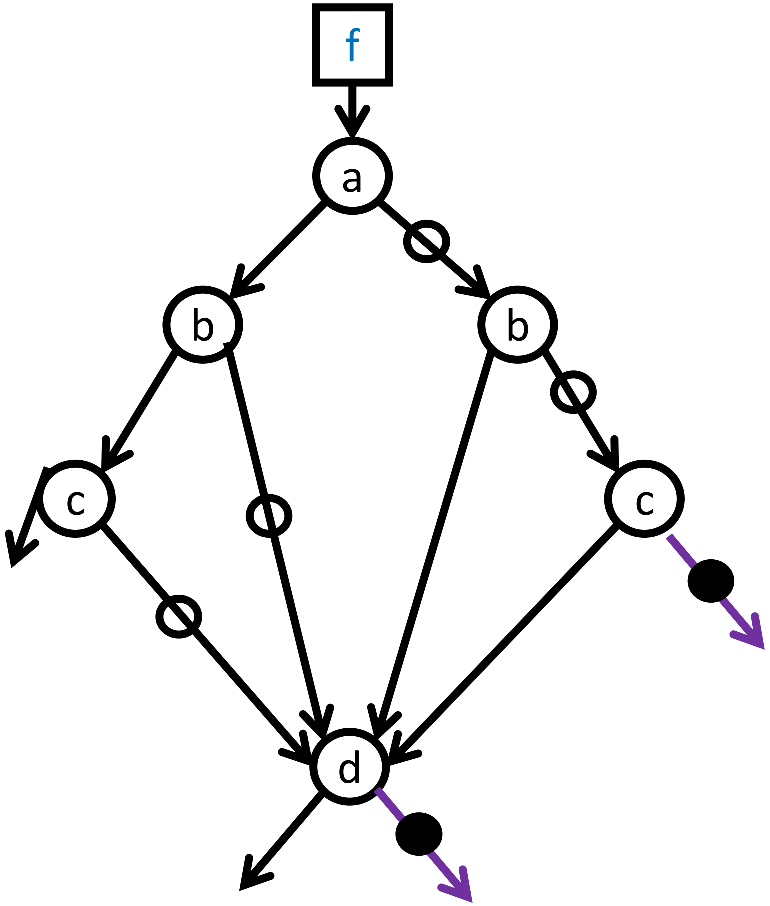


Reduced Ordered BDD

A reduced and ordered BDD gives a **canonical** representation of a formula

How can we use canonicity to decide validity and satisfiability in constant time?

Assume: BDD is given

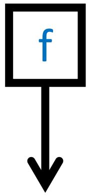


Reduced Ordered BDD

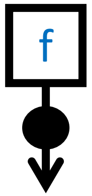
How can we use canonicity to decide validity and satisfiability in constant time?

Assume: BDD is given

BDD of a valid formula



BDD of UNSAT formula



Formula f is SAT if BDD looks different than this

From now on

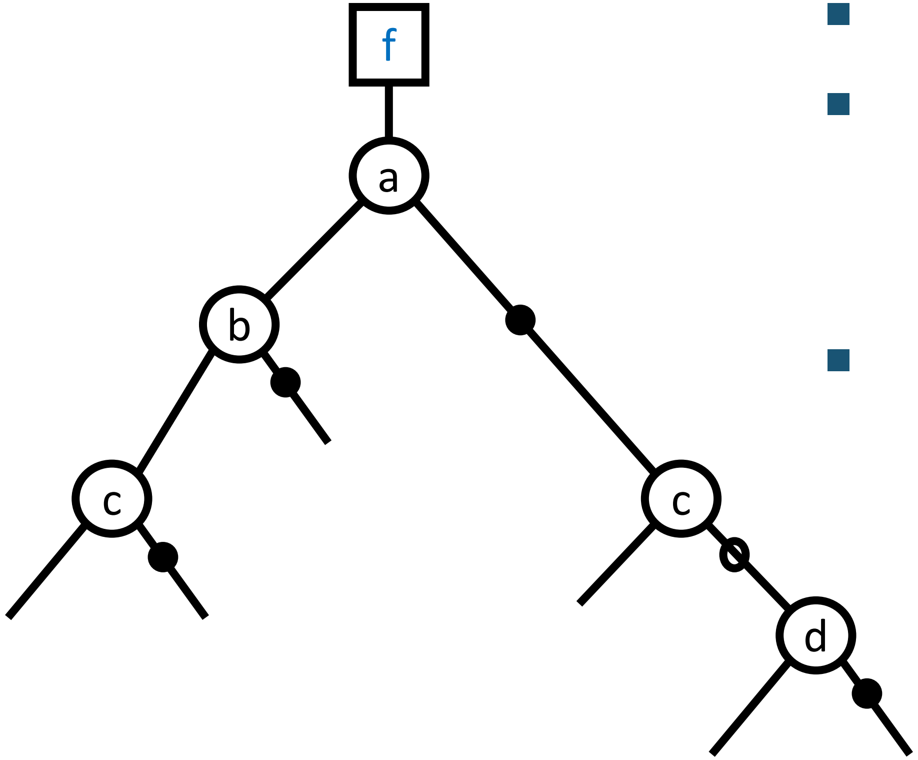
BDDs are always reduced and ordered!

Outline

- What are Binary Decision Diagrams (BDDs)?
  - Intuitive Explanation
  - Formal Definition
  - Reduced-Ordered BDDs
    - for us, a BDD is always reduced and ordered

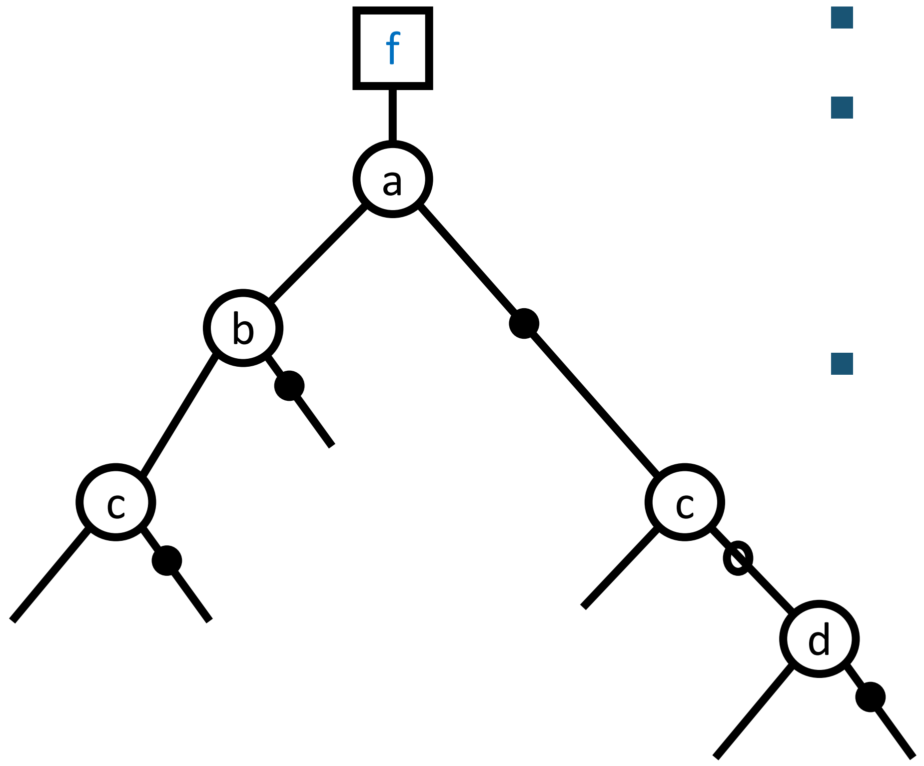
- **Represent a formula in propositional logic as BDD**
- **From the BDD, derive the formula that is represented by a BDD**

From BDD to Formula



- Complement flips truth value
- Satisfying model
  - Represented by path with **even number of negations**
- Build DNF from satisfying models

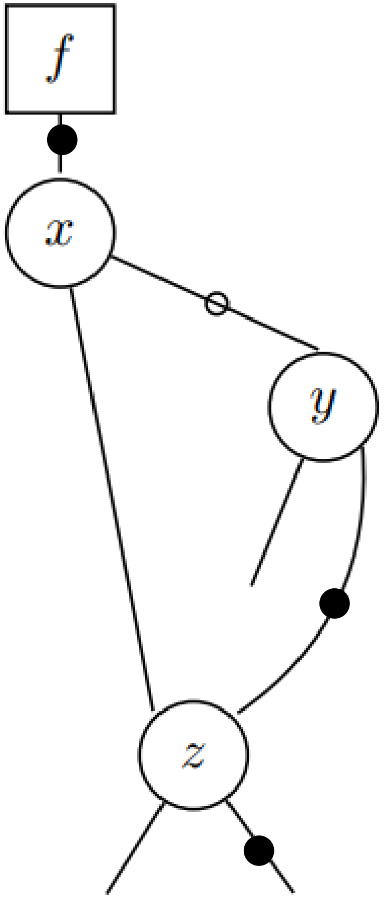
From BDD to Formula



- Complement flips truth value
- Satisfying model
  - Represented by path with **even number of negations**
- Build DNF from satisfying models

$$f = (a \land b \land c) \lor (\neg a \land \neg c \land \neg d)$$

From BDD to Formula



- Complement flips truth value
- Satisfying model
  - Represented by path with **even number of negations**
- Build DNF from satisfying models

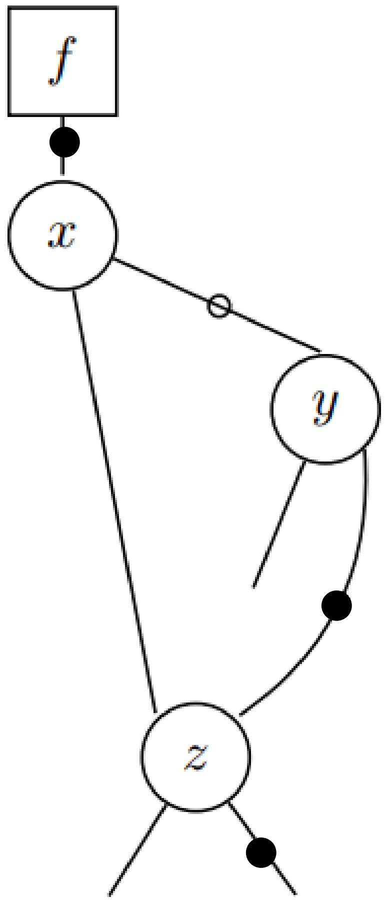
What is the formula f represented by this BDD?

(a) $f = (x \wedge z) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y \wedge \neg z)$

(b) $f = (x \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge z)$

(c) $f = (x \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z)$

From BDD to Formula



- Complement flips truth value
- Satisfying model
  - Represented by path with **even number of negations**
- Build DNF from satisfying models

(a) $f = (x \wedge z) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y \wedge \neg z)$

(b) $f = (x \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge z)$

(c) $\boldsymbol{f = (x \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z)}$

Outline

- What are Binary Decision Diagrams (BDDs)?
  - Intuitive Explanation
  - Formal Definition
  - Reduced-Ordered BDDs
    - for us, a BDD is always reduced and ordered

- **Represent a formula in propositional logic as BDD**
- **From the BDD, derive the formula that is represented by a BDD**

From Formula to BDD

1. Compute all Cofactors
2. Draw ROBDD from Cofactors
3. Shift Negations Upwards

From Formula to BDD – Step 1: Cofactors

- Boolean formula f w.r.t. a variable x
  - Positive Cofactor f_x: f with x set to $\top$
  - Negative Cofactor $f_{\neg x}$: f with x set to $\bot$

- Example:
  - $f = (x \wedge y) \vee (\neg x \wedge z)$

$$f_x =$$
$$f_{\neg x} =$$

From Formula to BDD – Step 1: Cofactors

- Boolean formula f w.r.t. a variable x
  - Positive Cofactor f_x: f with x set to $\top$
  - Negative Cofactor $f_{\neg x}$: f with x set to $\bot$

- Example:
  - $f = (x \wedge y) \vee (\neg x \wedge z)$
    - $f_x = y$
    - $f_{\neg x} = z$

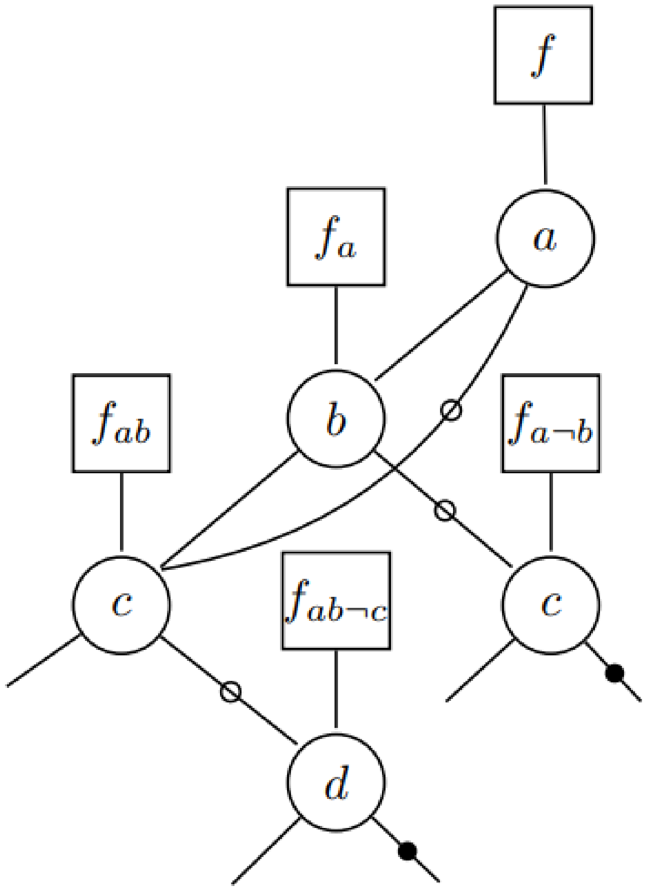
From Formula to BDD

Construct the BDD for the formula $\boldsymbol{f} = ((\boldsymbol{a} \wedge \boldsymbol{b} \vee \neg \boldsymbol{a}) \wedge \neg \boldsymbol{c} \wedge \boldsymbol{d}) \vee \boldsymbol{c}$.
Use the variable order $a < b < c < d$

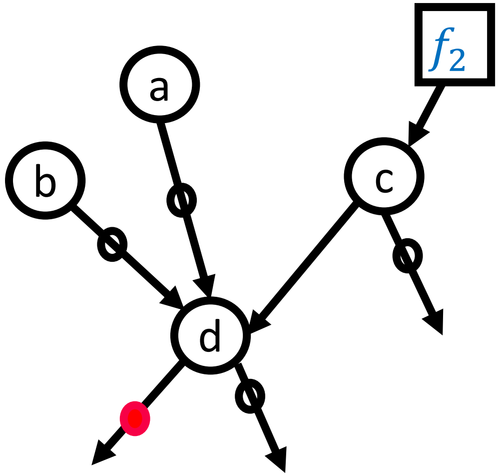
From Formula to BDD

Construct the BDD for the formula $\boldsymbol{f} = ((\boldsymbol{a} \wedge \boldsymbol{b} \vee \neg \boldsymbol{a}) \wedge \neg \boldsymbol{c} \wedge \boldsymbol{d}) \vee \boldsymbol{c}$.
Use the variable order $a < b < c < d$

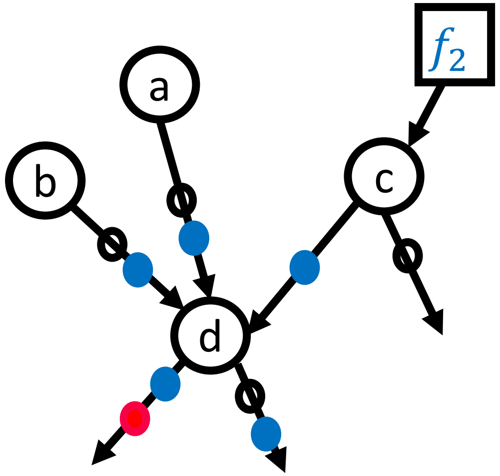
$$
\begin{aligned}
f_a &= b \wedge \neg c \wedge d \vee c \\
f_{ab} &= \neg c \wedge d \vee c \\
f_{abc} &= \top \\
f_{ab\neg c} &= d \\
f_{ab\neg cd} &= \top \\
f_{ab\neg c\neg d} &= \bot \\
f_{a\neg b} &= c \\
f_{a\neg bc} &= \top \\
f_{a\neg b\neg c} &= \bot \\
f_{\neg a} &= \neg c \wedge d \vee c = f_{ab}
\end{aligned}
$$



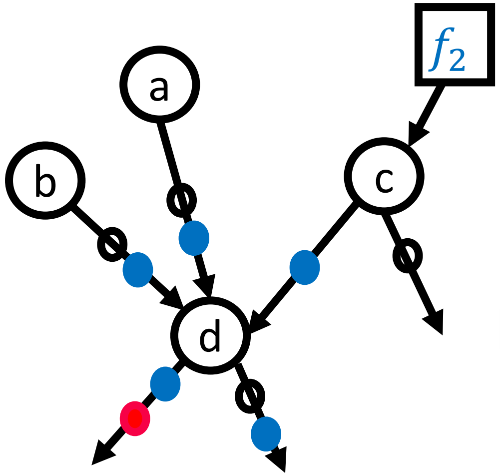
From Formula to BDD – Step 3: Shift Negations Upwards

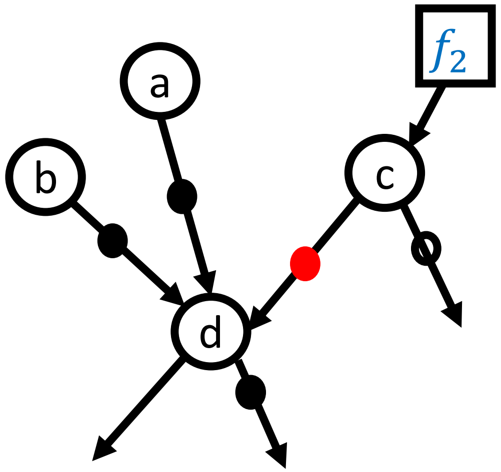


From Formula to BDD – Step 3: Shift Negations Upwards

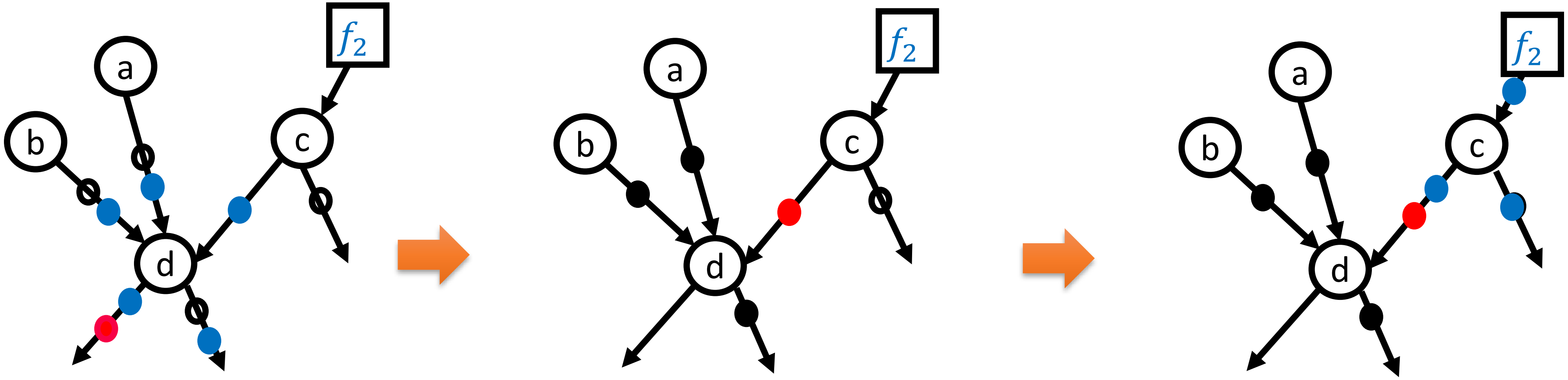


From Formula to BDD – Step 3: Shift Negations Upwards

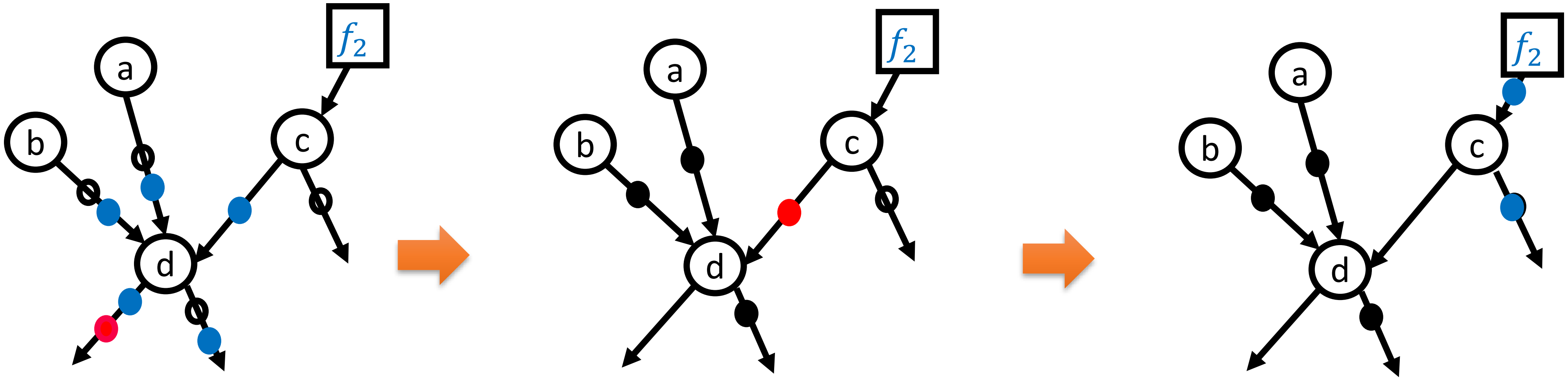




From Formula to BDD – Step 3: Shift Negations Upwards



From Formula to BDD – Step 3: Shift Negations Upwards



From Formula to BDD

Construct the BDD for the formula $\boldsymbol{f} = (\boldsymbol{a} \land \neg \boldsymbol{c}) \lor (\neg \boldsymbol{a} \land (\boldsymbol{b} \lor (\neg \boldsymbol{b} \land \boldsymbol{c})))$.
Use the variable order $a < b < c < d$

From Formula to BDD

Construct the BDD for the formula $\boldsymbol{f} = (\boldsymbol{a} \land \neg \boldsymbol{c}) \lor (\neg \boldsymbol{a} \land (\boldsymbol{b} \lor (\neg \boldsymbol{b} \land \boldsymbol{c})))$.
Use the variable order $a < b < c < d$

$f_a = \neg c$

$f_{ac} = \bot$

$f_{a\neg c} = \top$

$f_{\neg a} = b \lor (\neg b \land c)$

$f_{\neg ab} = \top$

$f_{\neg a \neg b} = c = \neg f_a$

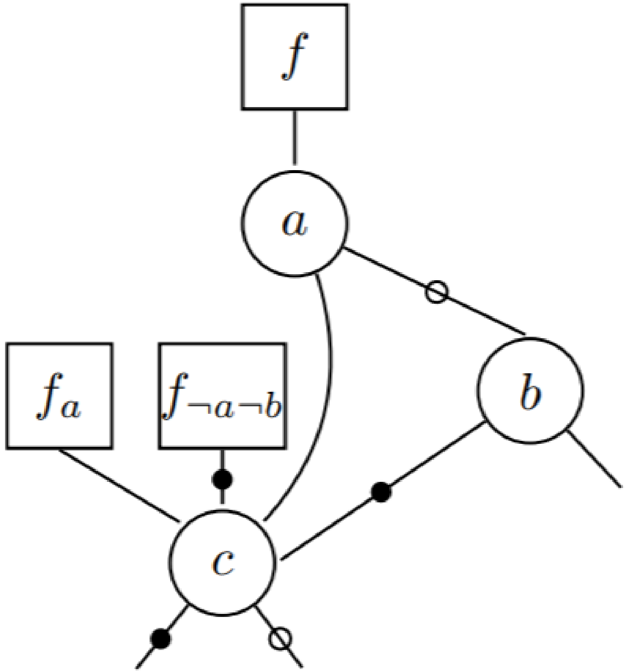
From Formula to BDD

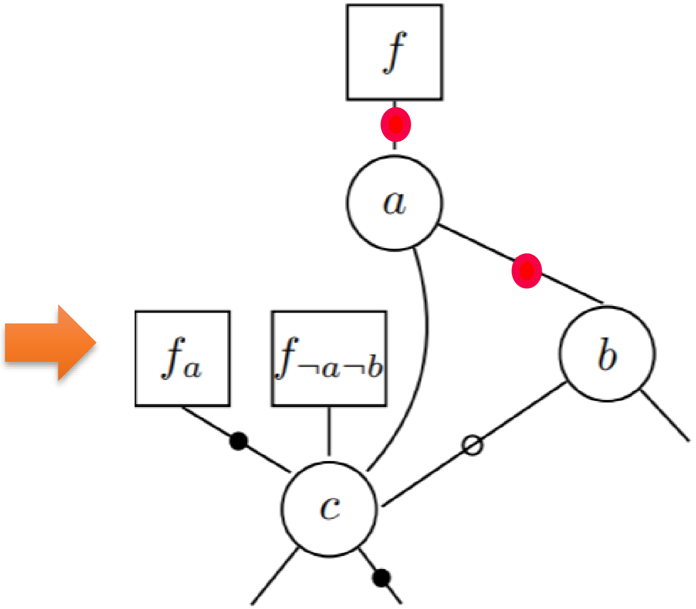
Construct the BDD for the formula $\boldsymbol{f} = (\boldsymbol{a} \wedge \neg \boldsymbol{c}) \vee (\neg \boldsymbol{a} \wedge (\boldsymbol{b} \vee (\neg \boldsymbol{b} \wedge \boldsymbol{c})))$.
Use the variable order $a < b < c < d$

Details: Next slide

$f_a = \neg c$
$f_{ac} = \bot$
$f_{a\neg c} = \top$

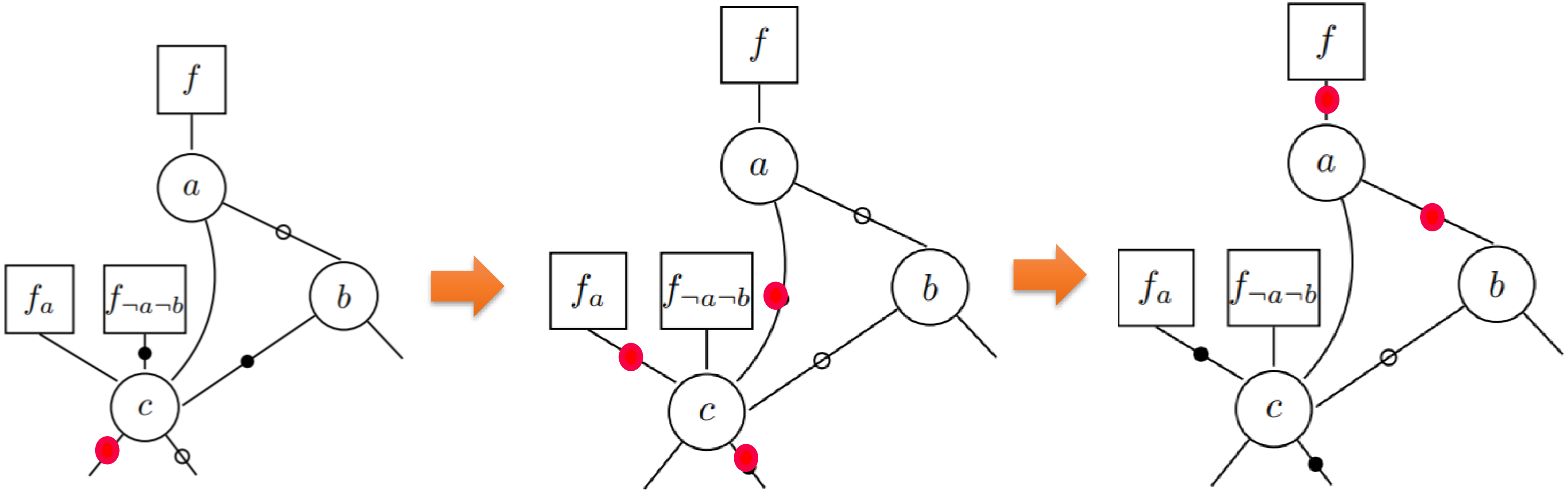
$f_{\neg a} = b \vee (\neg b \wedge c)$
$f_{\neg ab} = \top$
$f_{\neg a \neg b} = c = \neg f_a$





From Formula to BDD

Construct the BDD for the formula $\boldsymbol{f} = (\boldsymbol{a} \land \neg \boldsymbol{c}) \lor (\neg \boldsymbol{a} \land (\boldsymbol{b} \lor (\neg \boldsymbol{b} \land \boldsymbol{c})))$.
Use the variable order $a < b < c < d$



From Formula to BDD

Construct the BDD for the formula

$$\boldsymbol{f = (a \leftrightarrow b) \land (c \leftrightarrow d)}$$

Use the variable order $a < b < c < d$

From Formula to BDD

Construct the BDD for the formula

$$\boldsymbol{f = (a \leftrightarrow b) \wedge (c \leftrightarrow d)}$$

Use the variable order $a < b < c < d$

$f_a = b \wedge (c \leftrightarrow d)$

$f_{ab} = c \leftrightarrow d$

$f_{abc} = d$

$f_{abcd} = \top$

$f_{abc\neg d} = \bot$

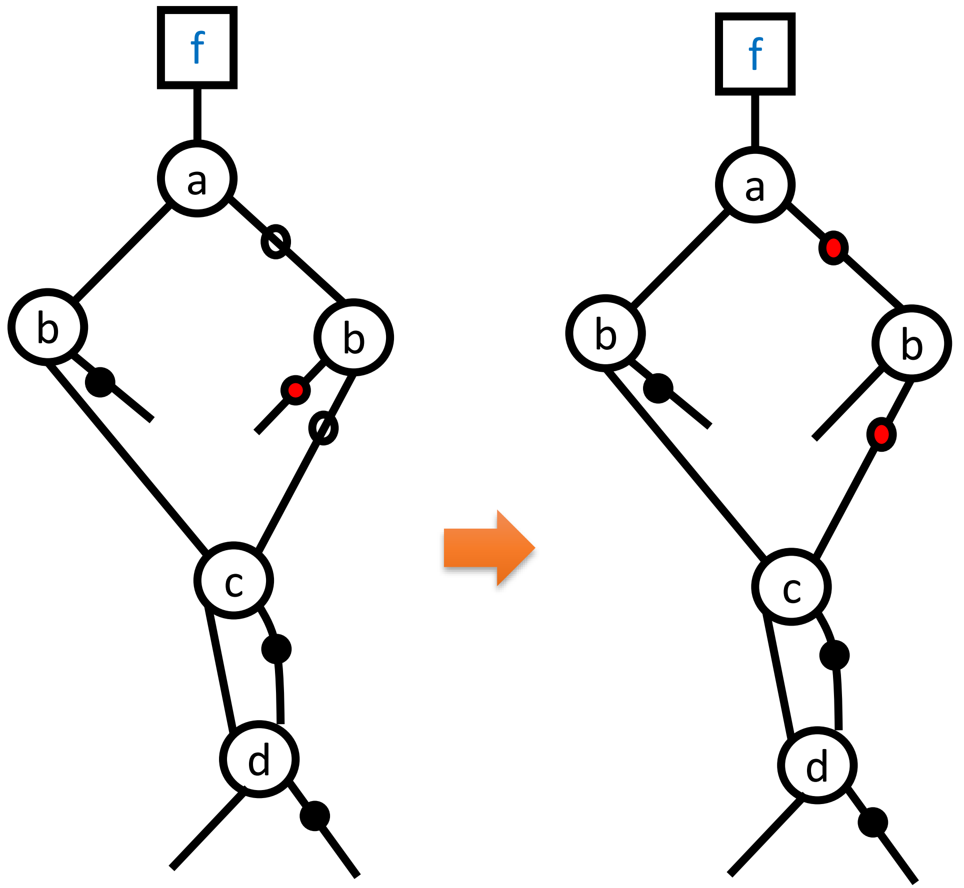
$f_{ab\neg c} = \neg d = \neg f_{abc}$

$f_{a\neg b} = \bot$

$f_{\neg a} = \neg b \wedge (c \leftrightarrow d)$

$f_{\neg ab} = \bot$

$f_{\neg a \neg b} = c \leftrightarrow d = f_{ab}$



From Formula to BDD

Construct the BDD for the formula

$$\boldsymbol{f} = (\boldsymbol{a} \leftrightarrow \boldsymbol{b}) \land (\boldsymbol{c} \leftrightarrow \boldsymbol{d})$$

Use the variable order $a < c < b < d$

From Formula to BDD

Construct the BDD for the formula

$$\boldsymbol{f = (a \leftrightarrow b) \land (c \leftrightarrow d)}$$

Use the variable order $a < c < b < d$

$f_a = b \land (c \leftrightarrow d)$

$f_{ac} = b \land d$

$f_{acb} = d$

$f_{acbd} = \top$

$f_{acb\neg d} = \bot$

$f_{ac\neg b} = \bot$

$f_{a\neg c} = b \land \neg d$

$f_{a\neg cb} = \neg d = \neg f_{acb}$

$f_{a\neg c\neg b} = \bot$

$f_{\neg a} = \neg b \land (c \leftrightarrow d)$

$f_{\neg ac} = \neg b \land d$

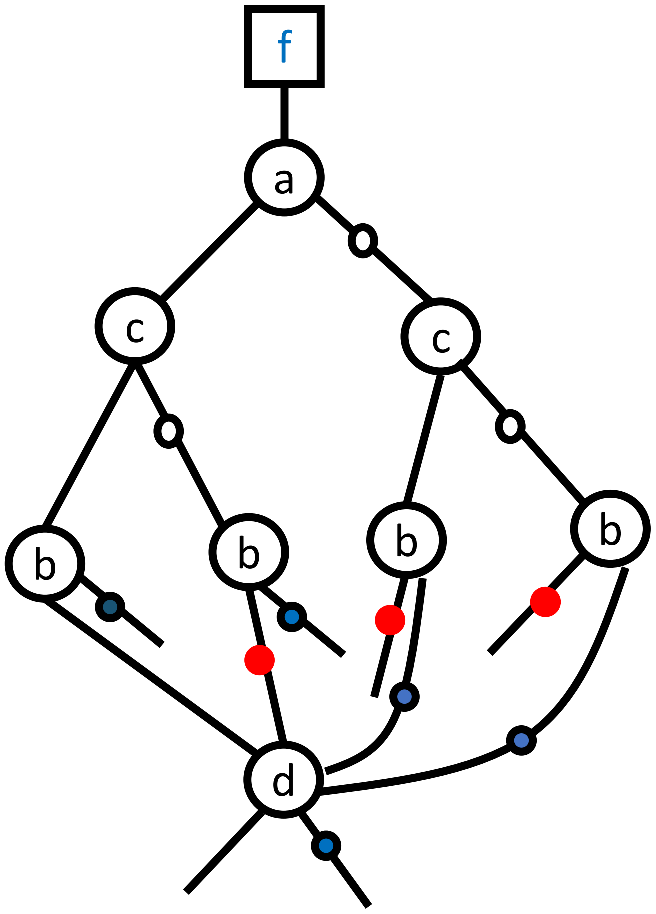
$f_{\neg acb} = \bot$

$f_{\neg ac\neg b} = d = f_{acb}$

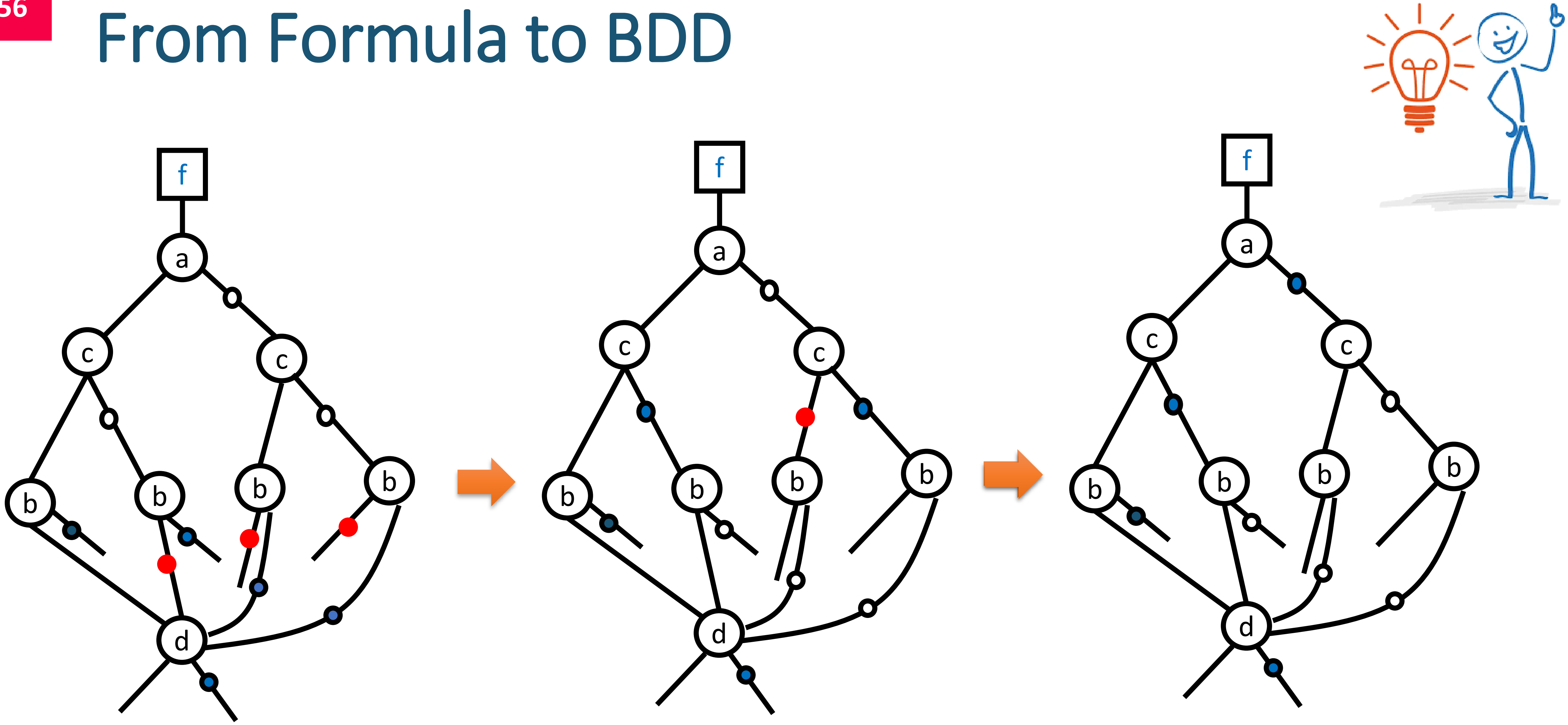
$f_{\neg a\neg c} = \neg b \land \neg d$

$f_{\neg a\neg cb} = \bot$

$f_{\neg a\neg c\neg b} = \neg d = \neg f_{acb}$



From Formula to BDD



Disadvantages of BDDs

Size of BDDs strongly depends on variable order

- Hard to optimize
- Problem to find the optimal variable order is NP complete

Example before: $f = (x_1 \leftrightarrow x_1{}') \wedge (x_2 \leftrightarrow x_2{}') \wedge (x_3 \leftrightarrow x_3{}') \wedge (x_4 \leftrightarrow x_4{}') \wedge \cdots$

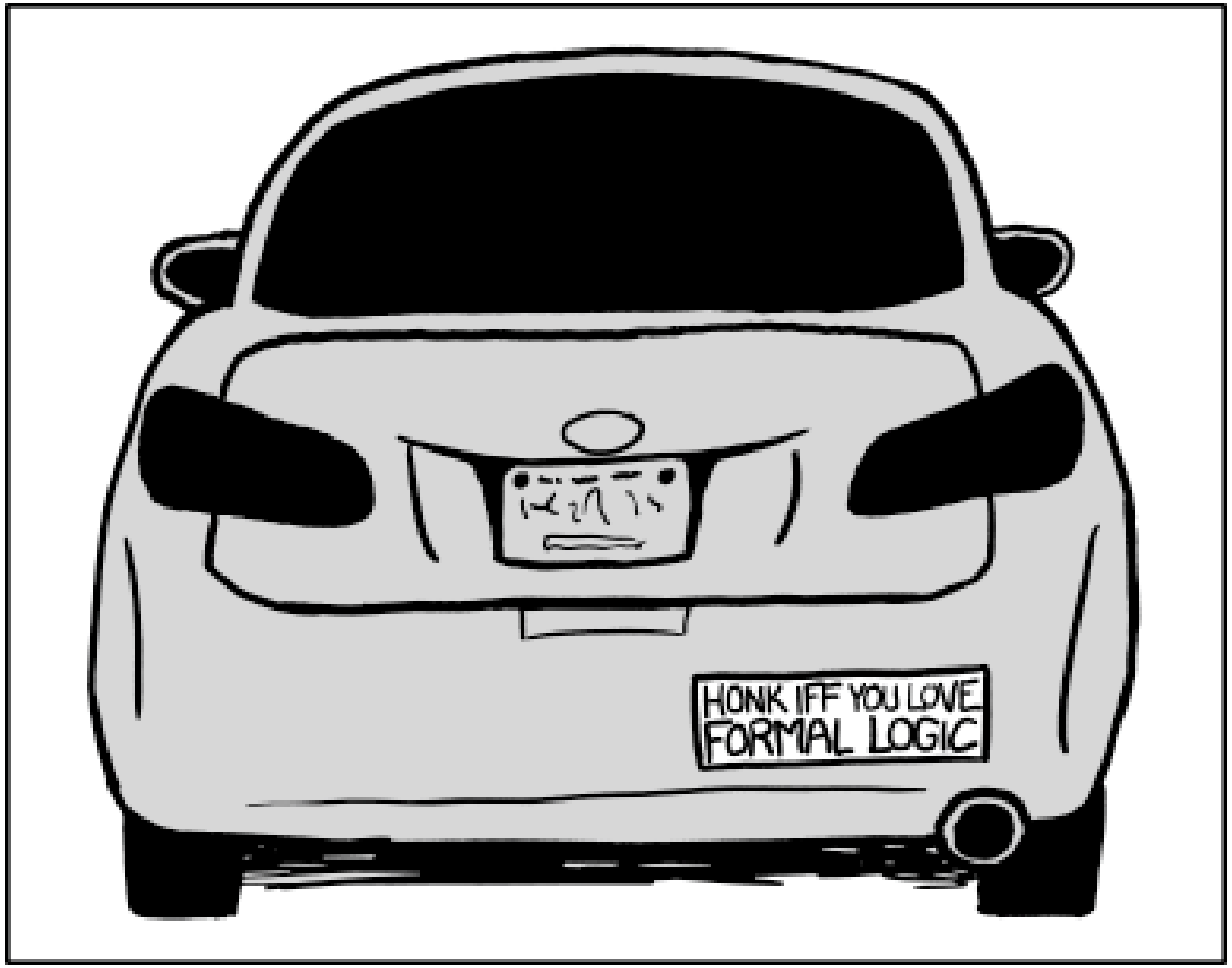
- Order: $x_1 < x_1' < x_2 < x_2' <$ ... ➡ size of BDD is $3 * n + 2$ nodes
- Order: $x_1 < x_2 < \cdots < x_1{}' < x_2' <$ ... ➡ size of BDD is $3 * 2^n - 1$ nodes

Learning Outcomes

After this lecture…

1. students can define and explain BDDs.
    - BDD = reduced and ordered binary decision diagram
    - Define and explain its elements and their meaning
2. students can represent a formula in propositional logic as BDD.
3. students can derive the formula that is represented by a BDD.
4. students can state properties of BDDs.
    - advantages, disadvantages

Thank You



SCIENCE
PASSION
TECHNOLOGY