Knowledge Compilation:
Principles and Applications

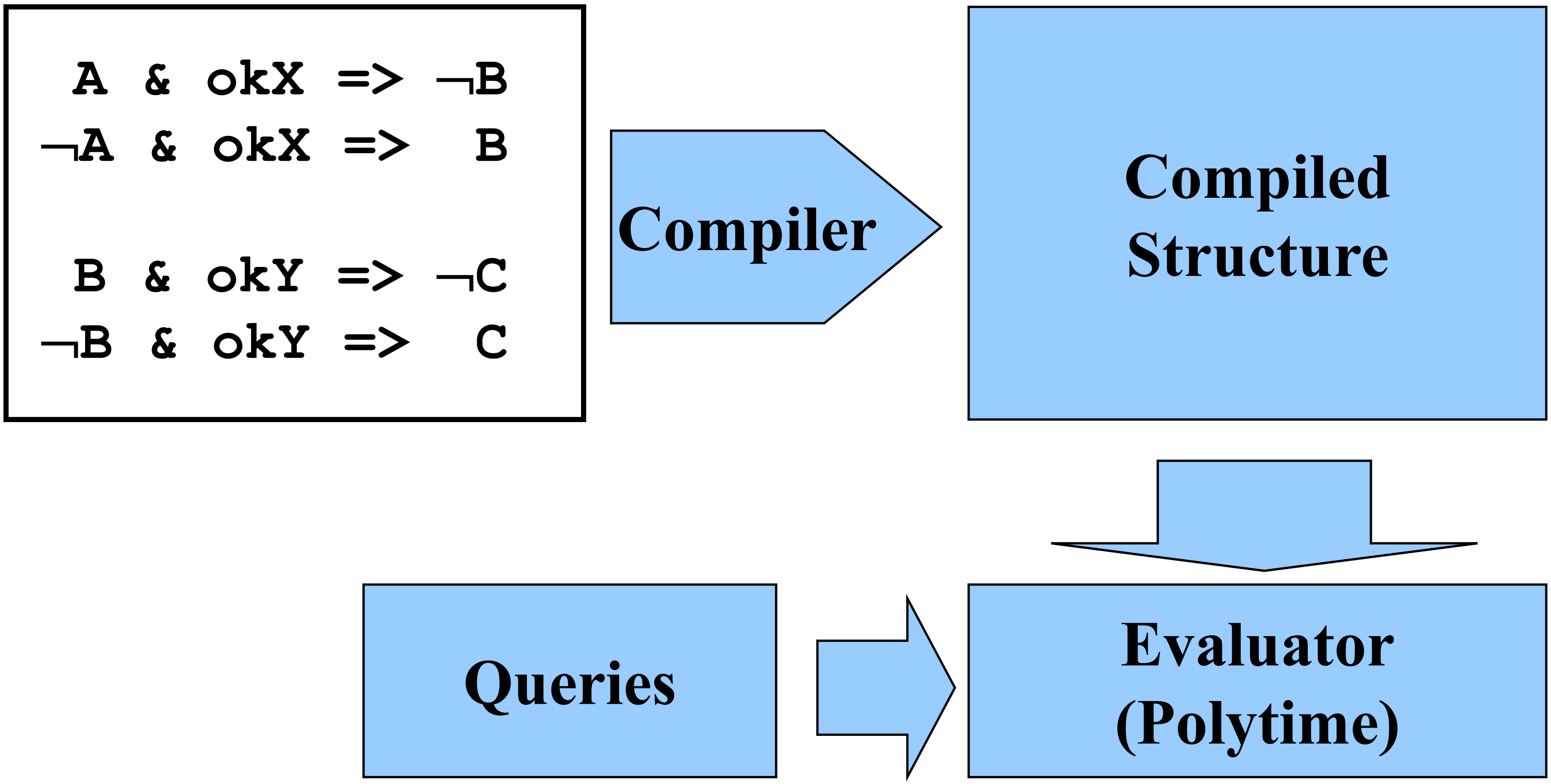
Adnan Darwiche
Computer Science Department, UCLA

Agenda

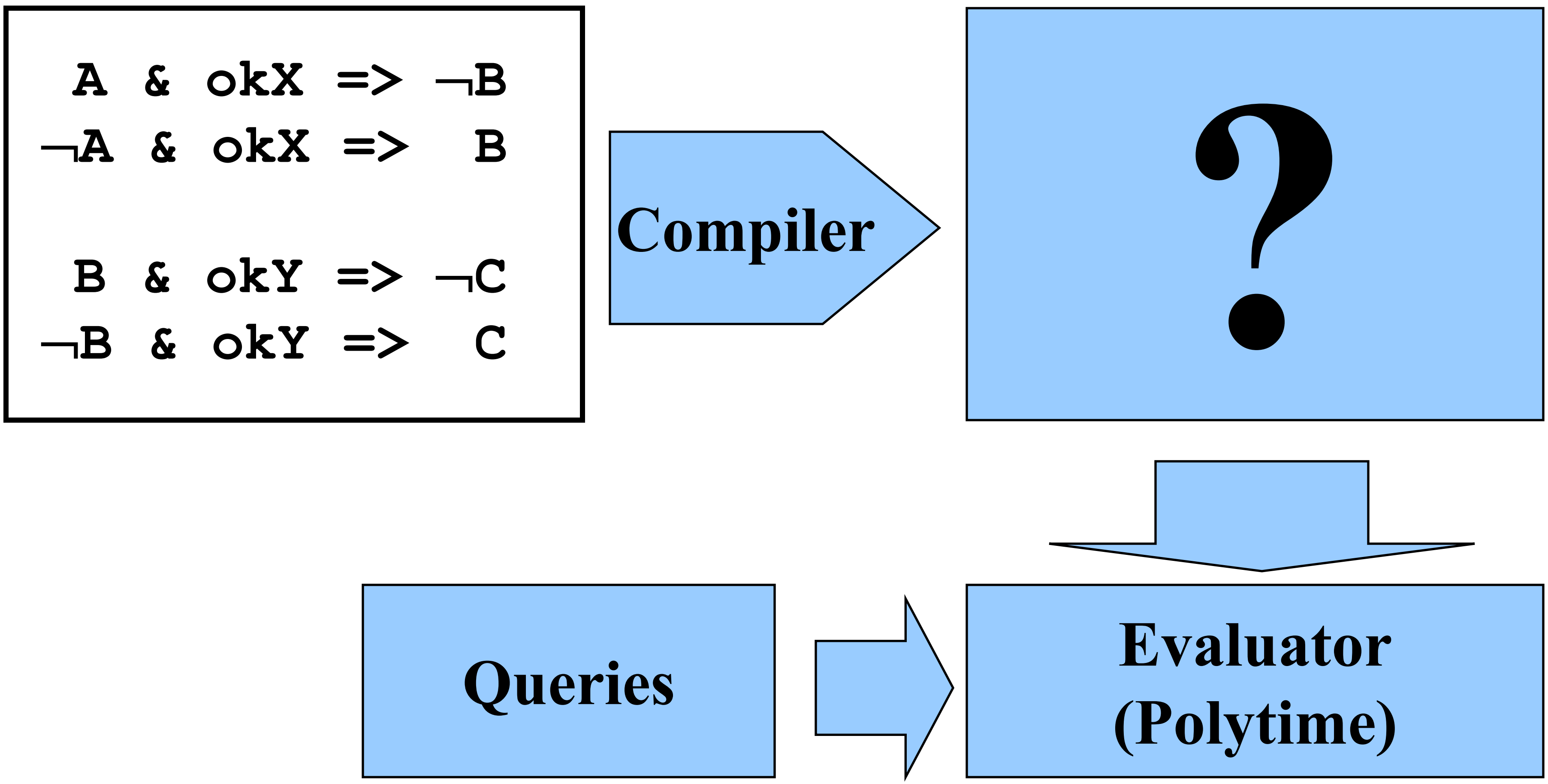
- Languages and Operations
- Knowledge Compilers
- Applications:
  - Explaining and Verifying AI Systems
  - Probabilistic Reasoning
  - Machine Learning

Languages & Operations

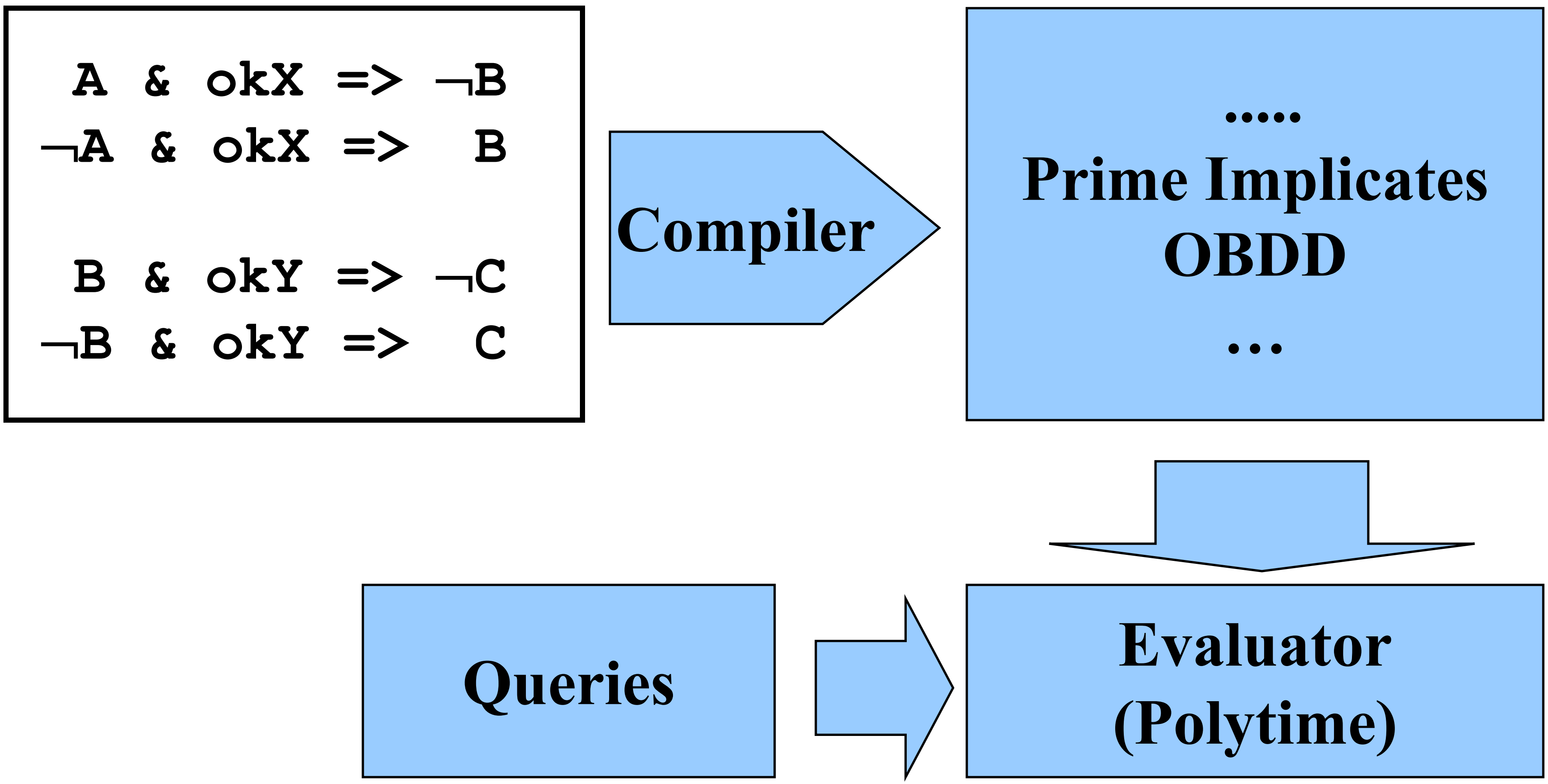
Knowledge Compilation



Knowledge Compilation



Knowledge Compilation



Knowledge Compilation Map

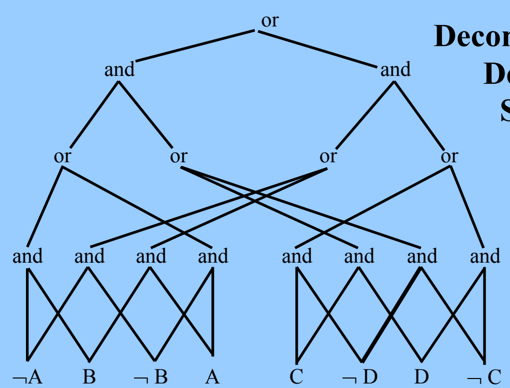
- What’s the space of possible target compilation languages?
  - Can it be synthesized in a semantically systematic way?
- How do the languages compare?
  - Succinctness (relative size)
  - Operations they support in polytime

Knowledge Compilation Map

- For a given application: identify needed operations
- Choose most succinct language that supports desired operations
- Compile knowledge base into chosen language

A Knowledge Compilation MAP

Negation Normal Form



Decomposability
Determinism
Smoothness
Flatness
Decision
Ordering

Succinctness

Polytime Operations

Consistency (CO)
Validity (VA)
Clausal entailment (CE)
Sentential entailment (SE)
Implicant testing (IP)
Equivalence testing (EQ)
Model Counting (CT)
Model enumeration (ME)

Projection (exist. quantification)
Conditioning
Conjoin, Disjoin, Negate

Propositional Logic

- **Literal** $X, \neg X$
- **Clause** $(X \vee \neg Y \vee \neg Z)$
- **Term** $(\neg X \wedge Y \wedge Z)$
- **CNF**: Conjunctive Normal Form

$$(X \vee \neg Y \vee \neg Z) \wedge \ldots \wedge (Y \vee \neg W)$$

- **DNF**: Disjunctive Normal Form

$$(\neg X \wedge Y \wedge Z) \vee \ldots \vee (X \wedge \neg Z \wedge W)$$

Propositional Logic

- **Truth assignment (TA)**

$$X : true, Y : false, Z : true, W : false$$

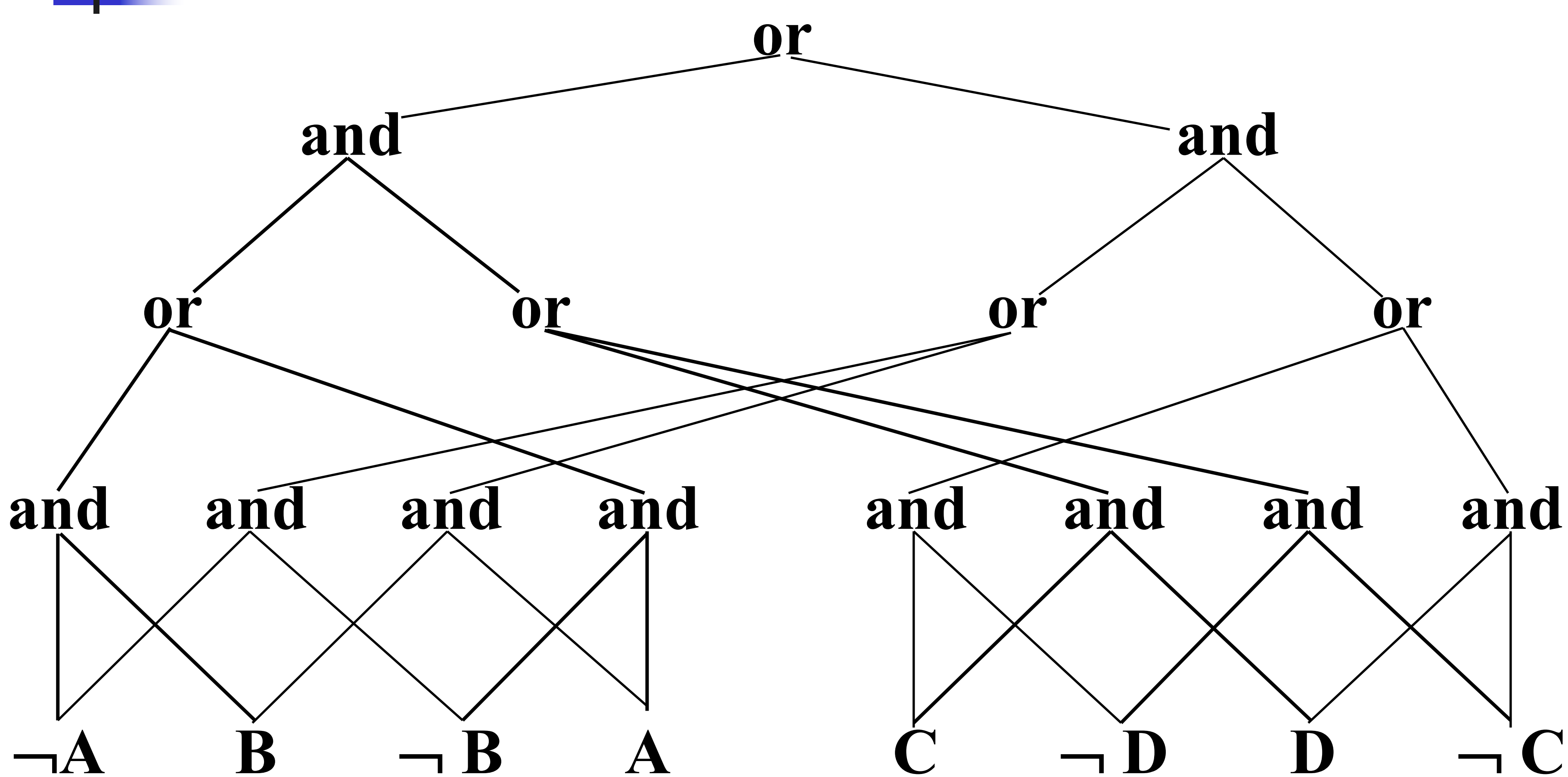
- **TA satisfies sentence (model)**

$$(X \vee \neg Y \vee \neg Z) \wedge \ldots \wedge (Y \vee \neg W)$$

- **Following TA is not a model**

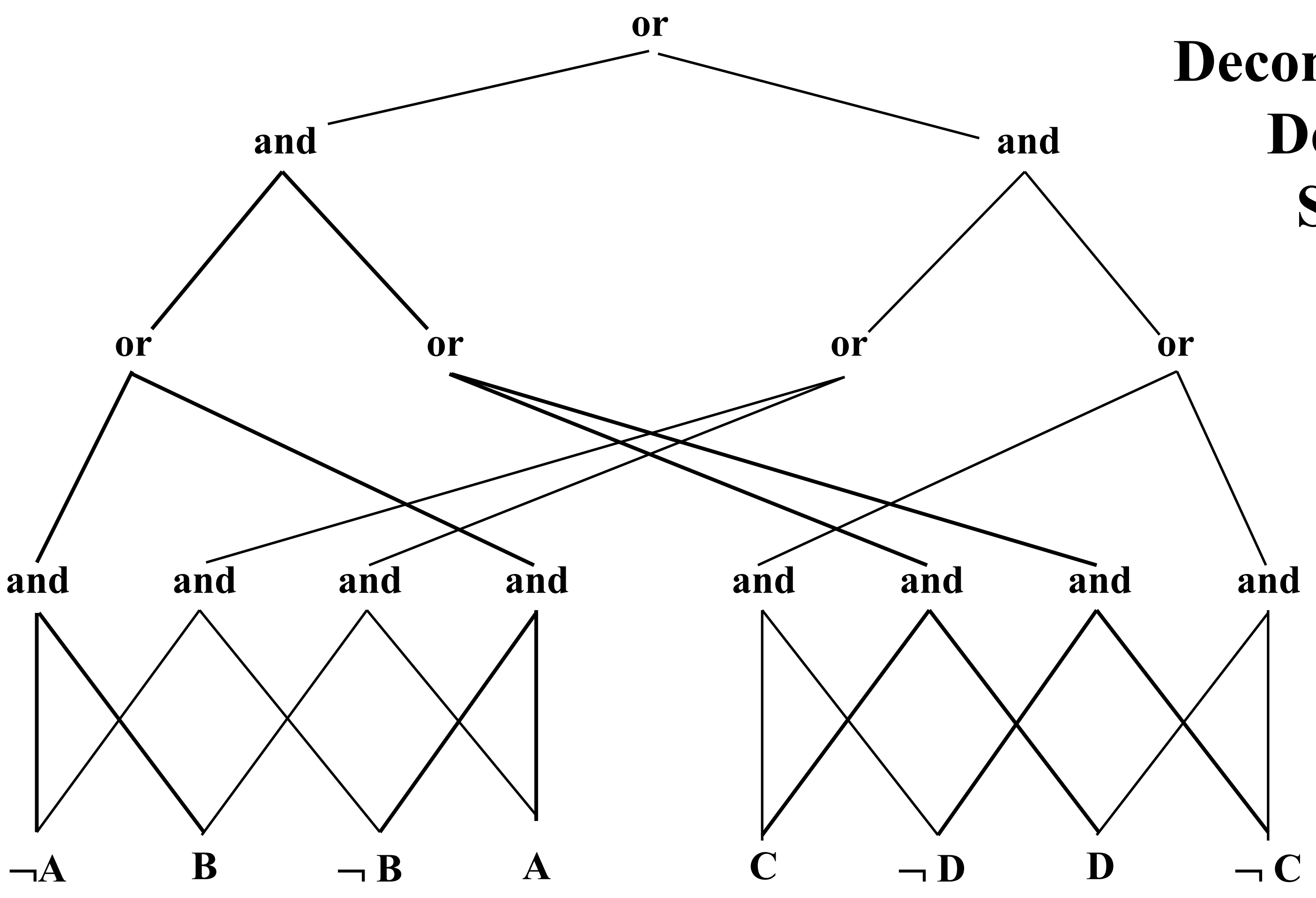
$$X : true, Y : false, Z : true, W : true$$

Negation Normal Form



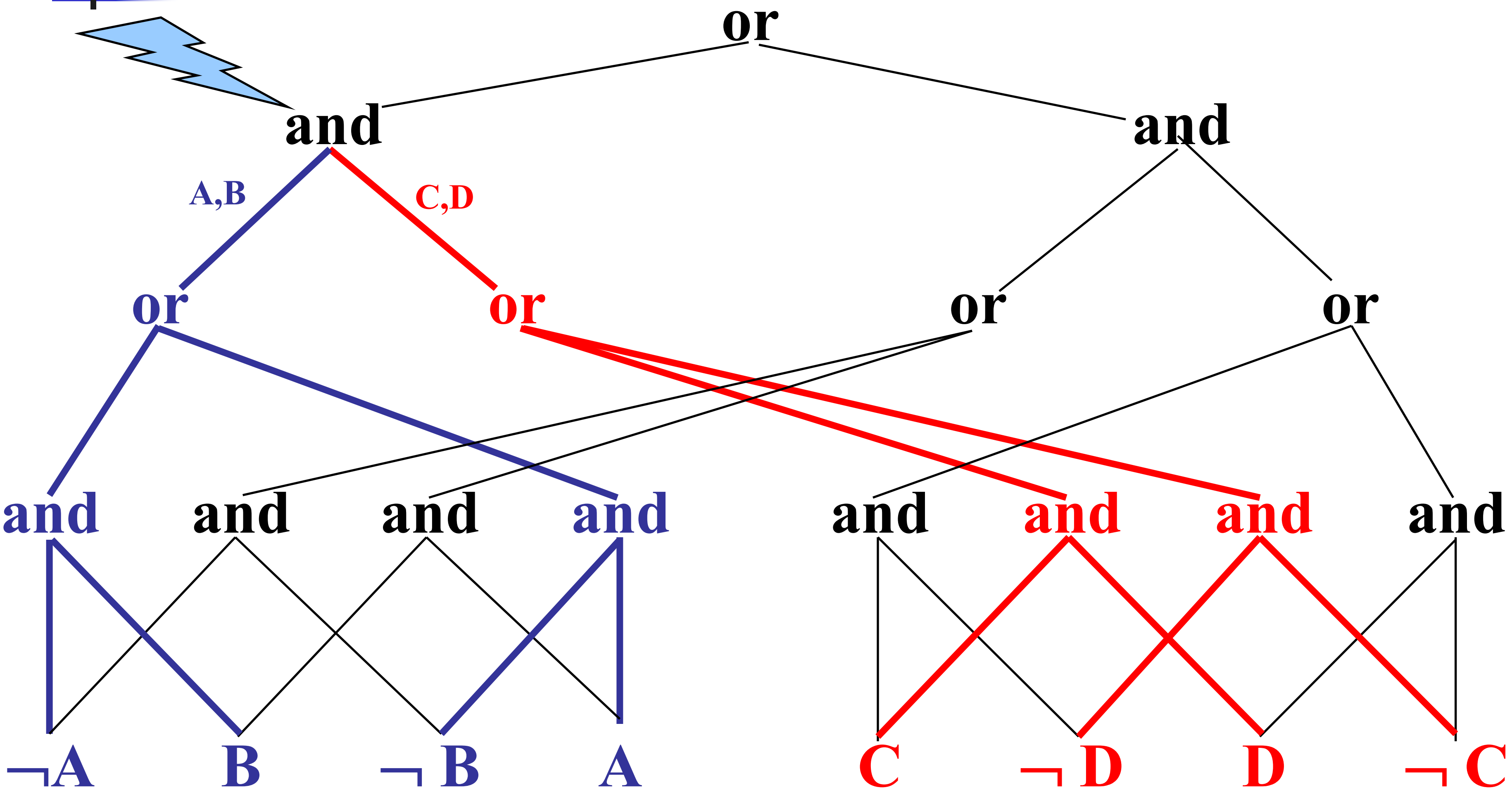
rooted DAG (Circuit)

Negation Normal Form

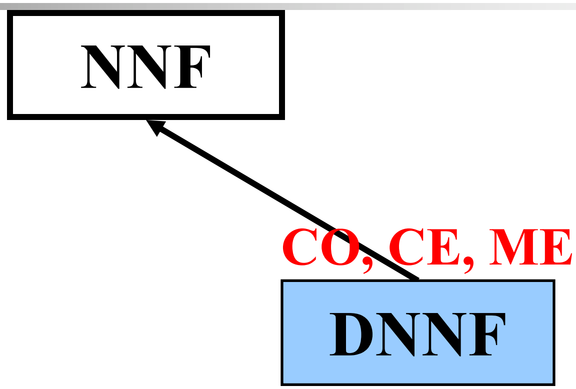


Decomposability
Determinism
Smoothness
Flatness
Decision
Ordering

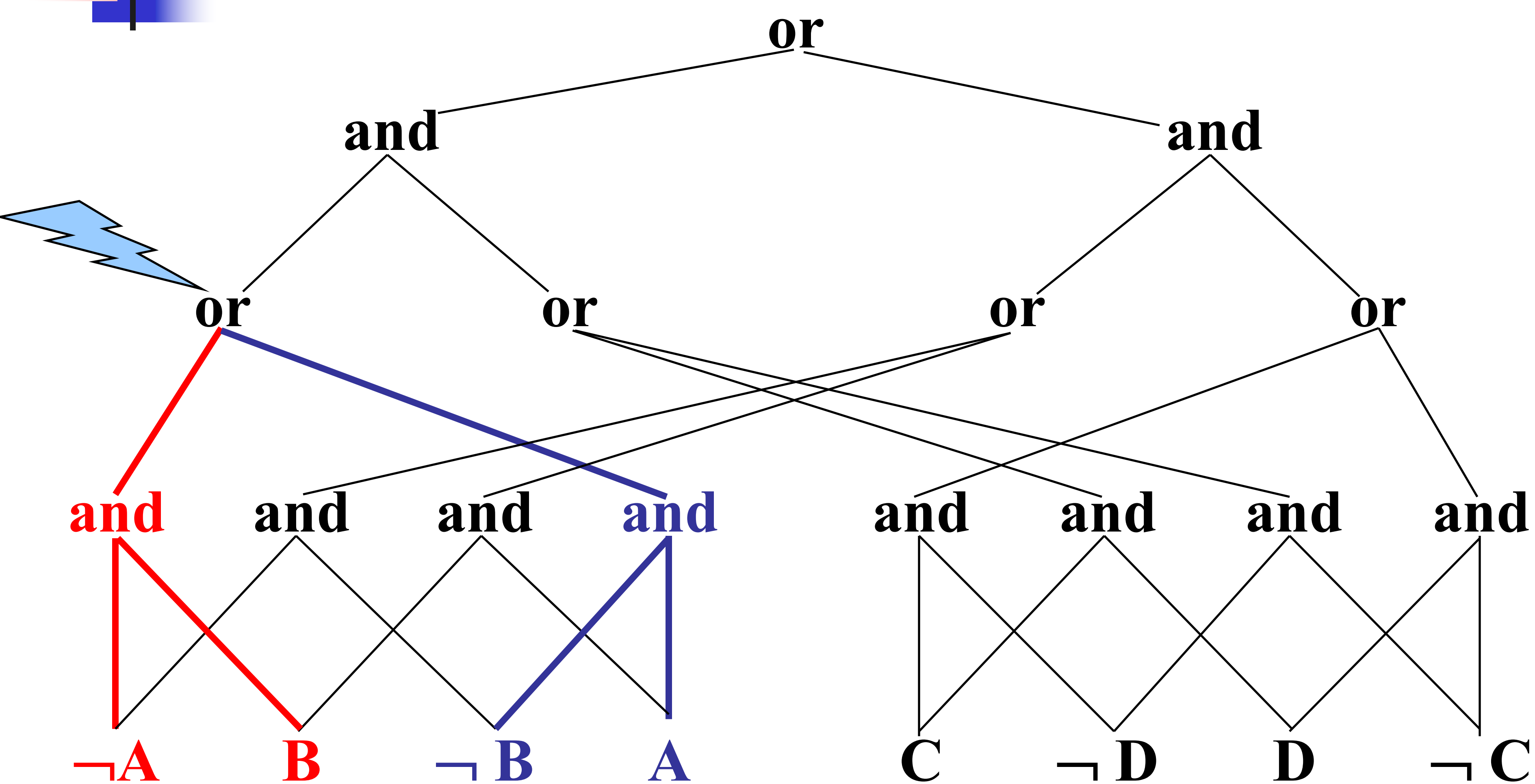
Decomposability



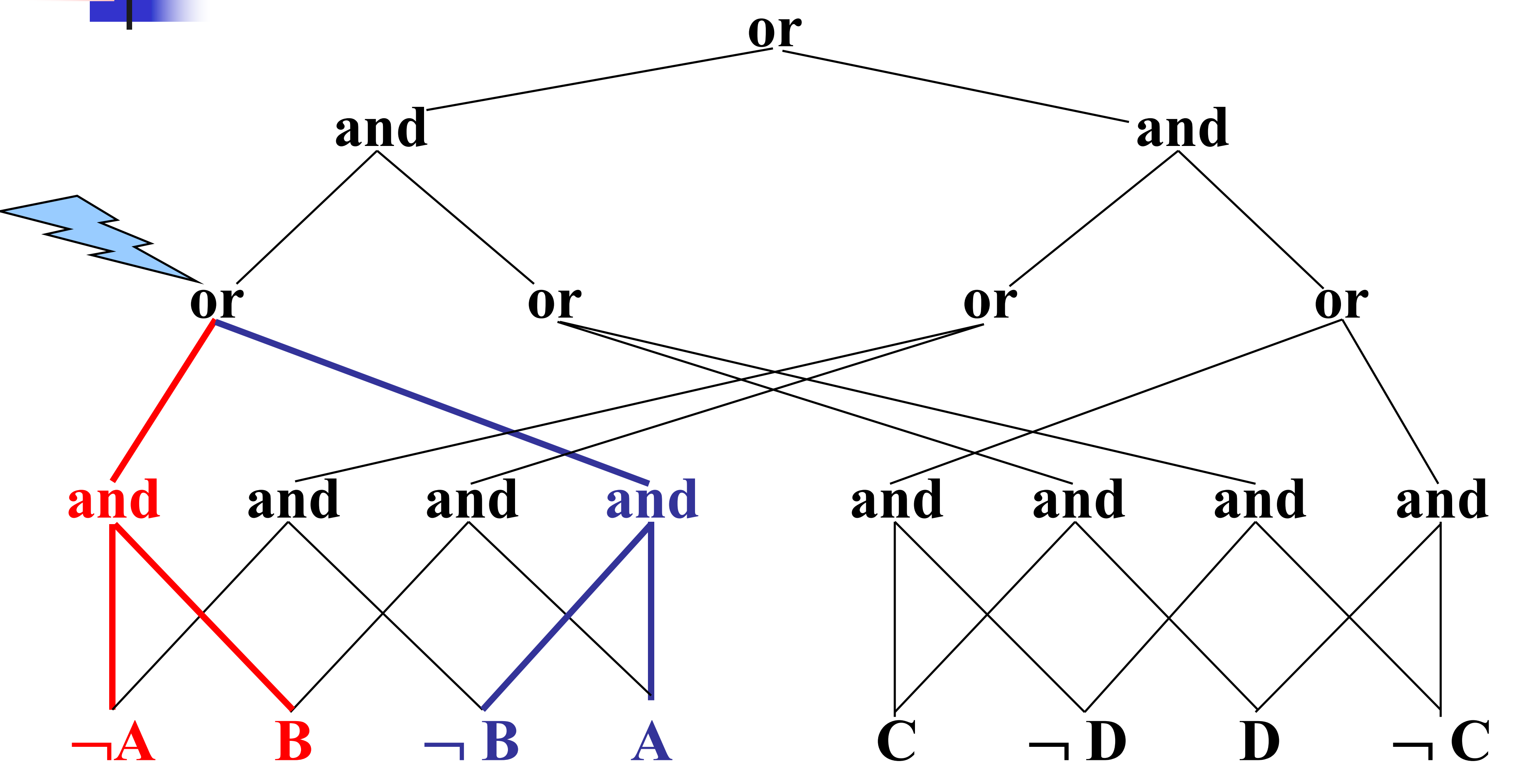
NNF Subsets



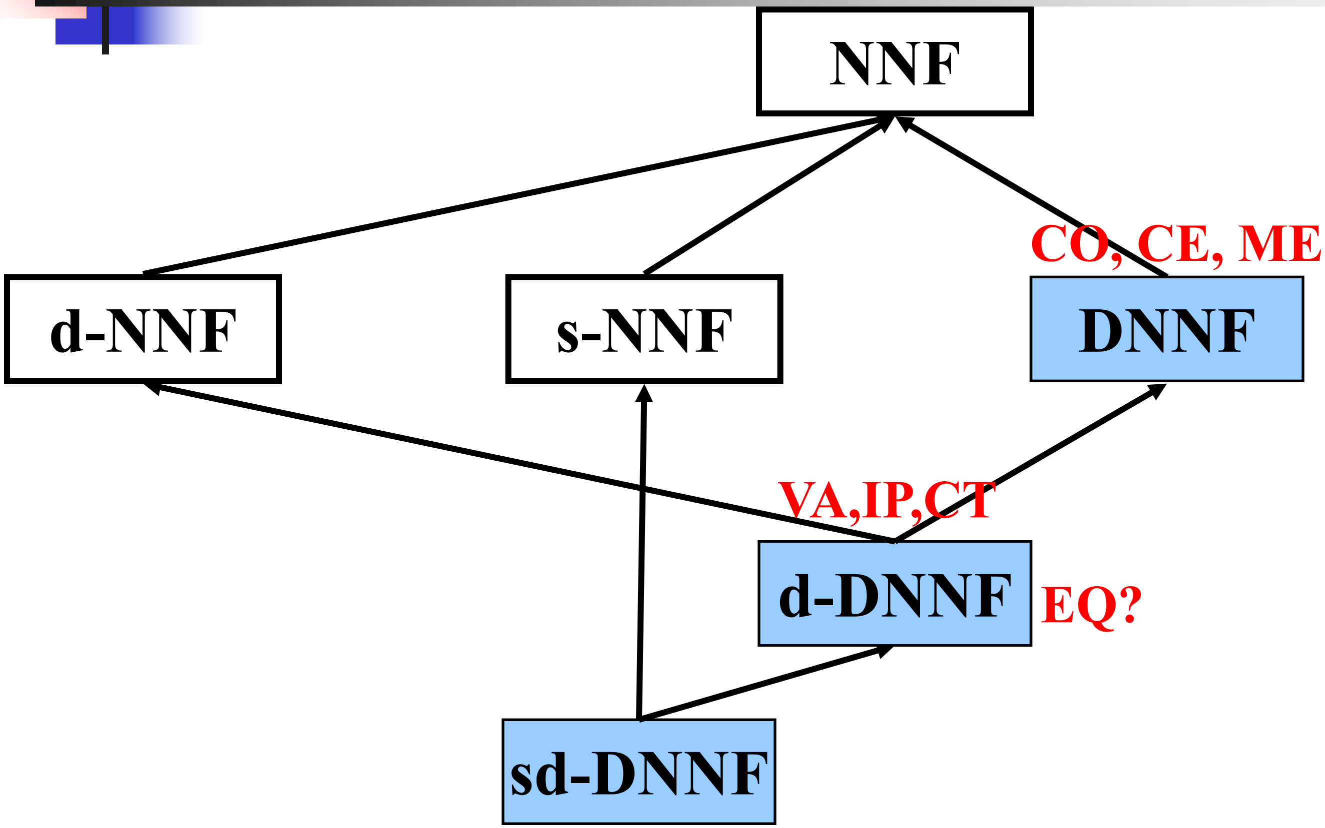
Determinism



Smoothness

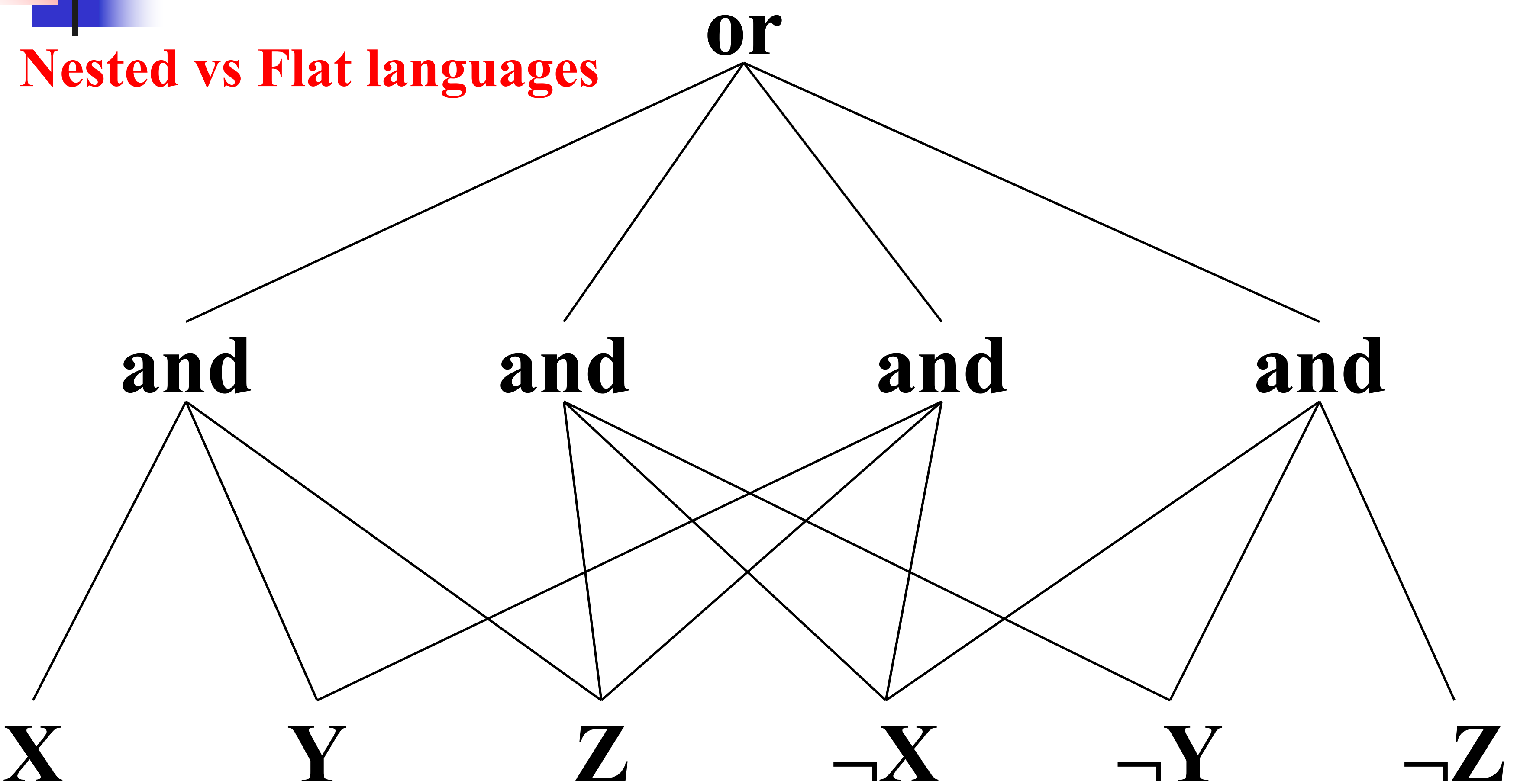


NNF Subsets



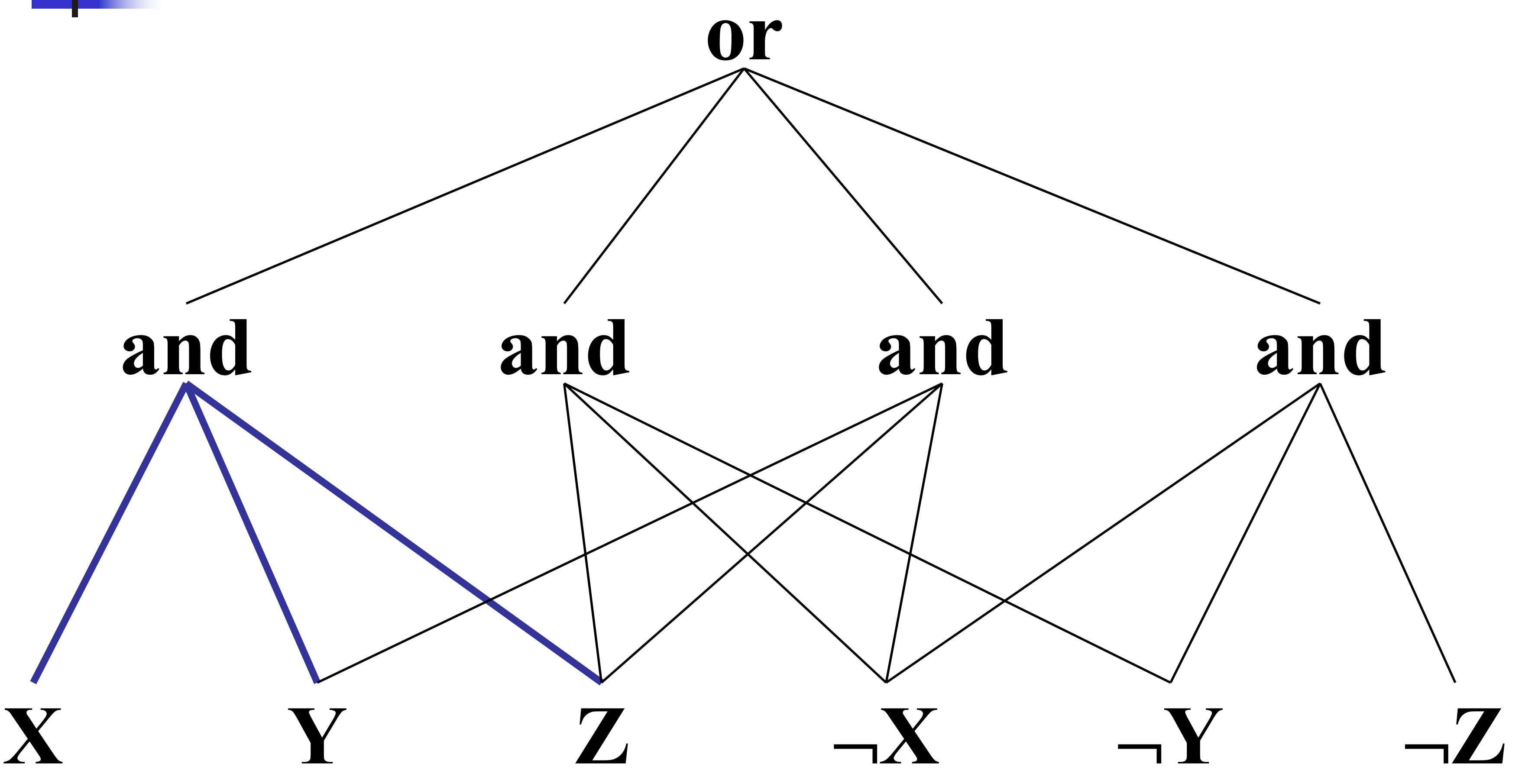
Flatness

Nested vs Flat languages



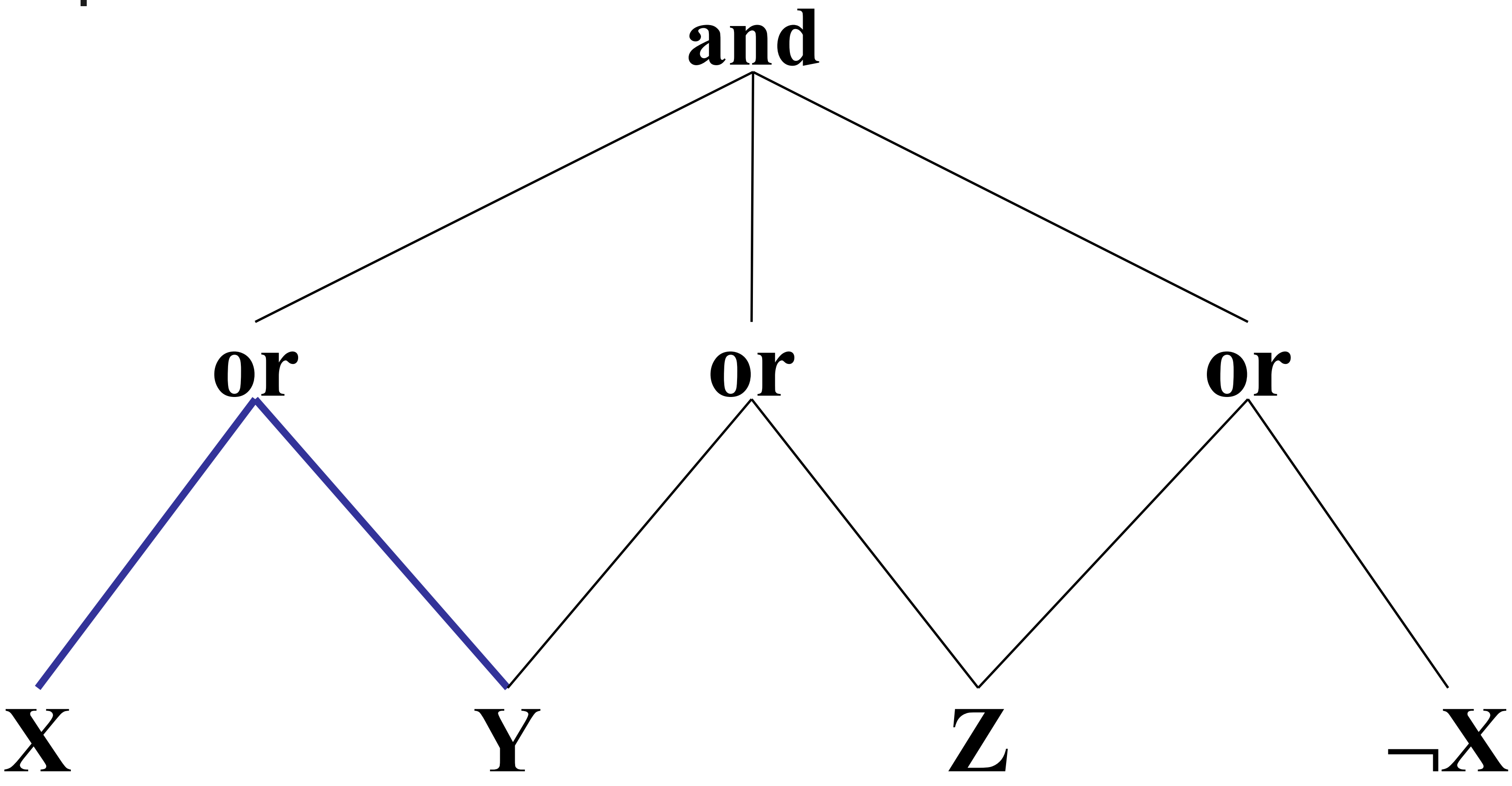
$(X \wedge Y \wedge Z) \vee (Z \wedge \neg X \wedge \neg Y) \vee (Y \wedge Z \wedge \neg X) \vee (\neg X \wedge \neg Y \wedge \neg Z)$

Simple Conjunction

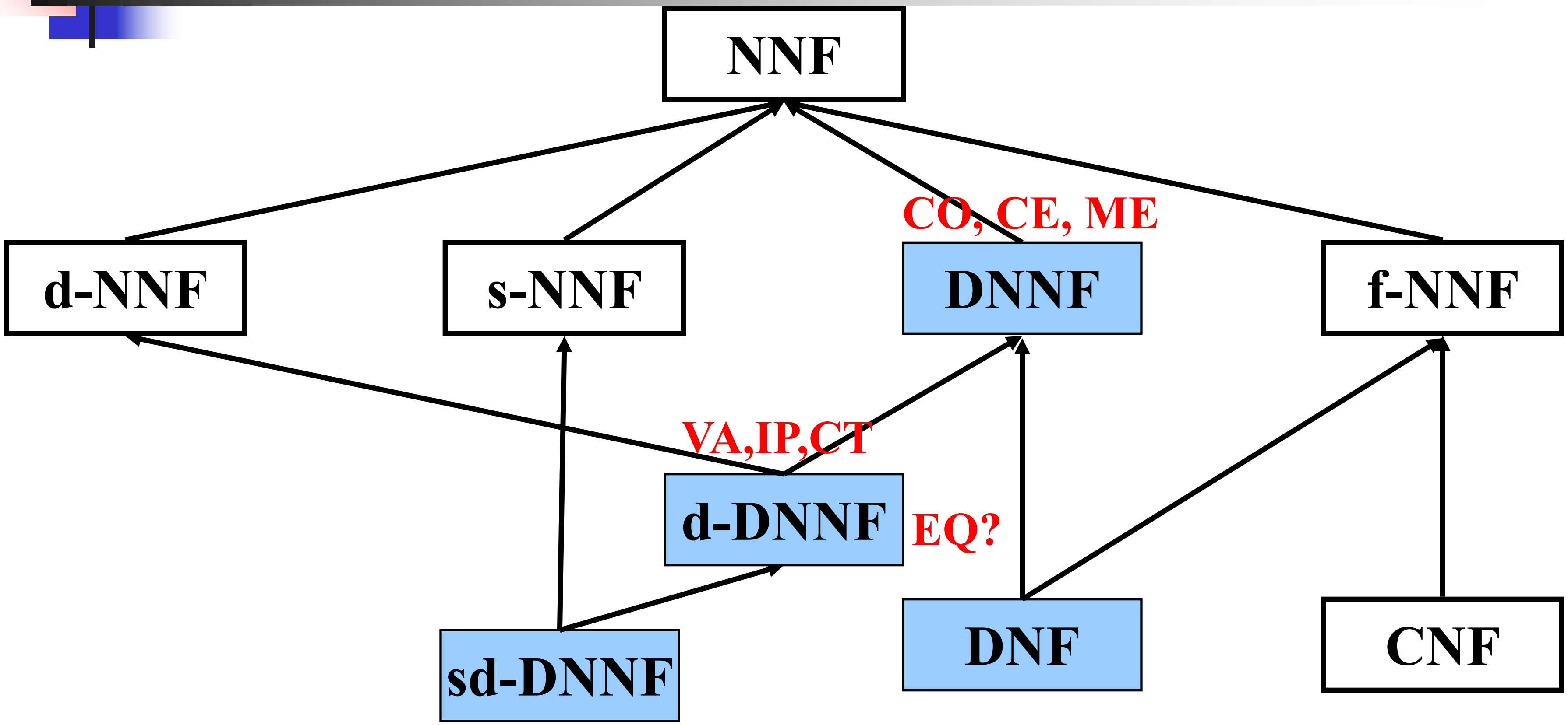


Simple conjunction implies decomposability

Simple Disjunction



NNF Subsets



NNF
d-NNF
s-NNF
CO, CE, ME
DNNF
f-NNF
VA,IP,CT
d-DNNF
EQ?
sd-DNNF
DNF
CNF

Prime Implicates (PI)

Resolution

$$\frac{(\alpha \vee X), (\beta \vee \neg X)}{(\alpha \vee \beta)}$$

n that:

er

e CNF, it must

he CNF

- CNF:

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee D)$$

- PI:

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee D) \wedge$$
$$(\neg A \vee C) \wedge (\neg A \vee D) \wedge (\neg B \vee D)$$

Prime Implicants (IP)

Consensus h that:

$$\frac{(\alpha \wedge X), (\beta \wedge \neg X)}{(\alpha \wedge \beta)}$$

it must imply a

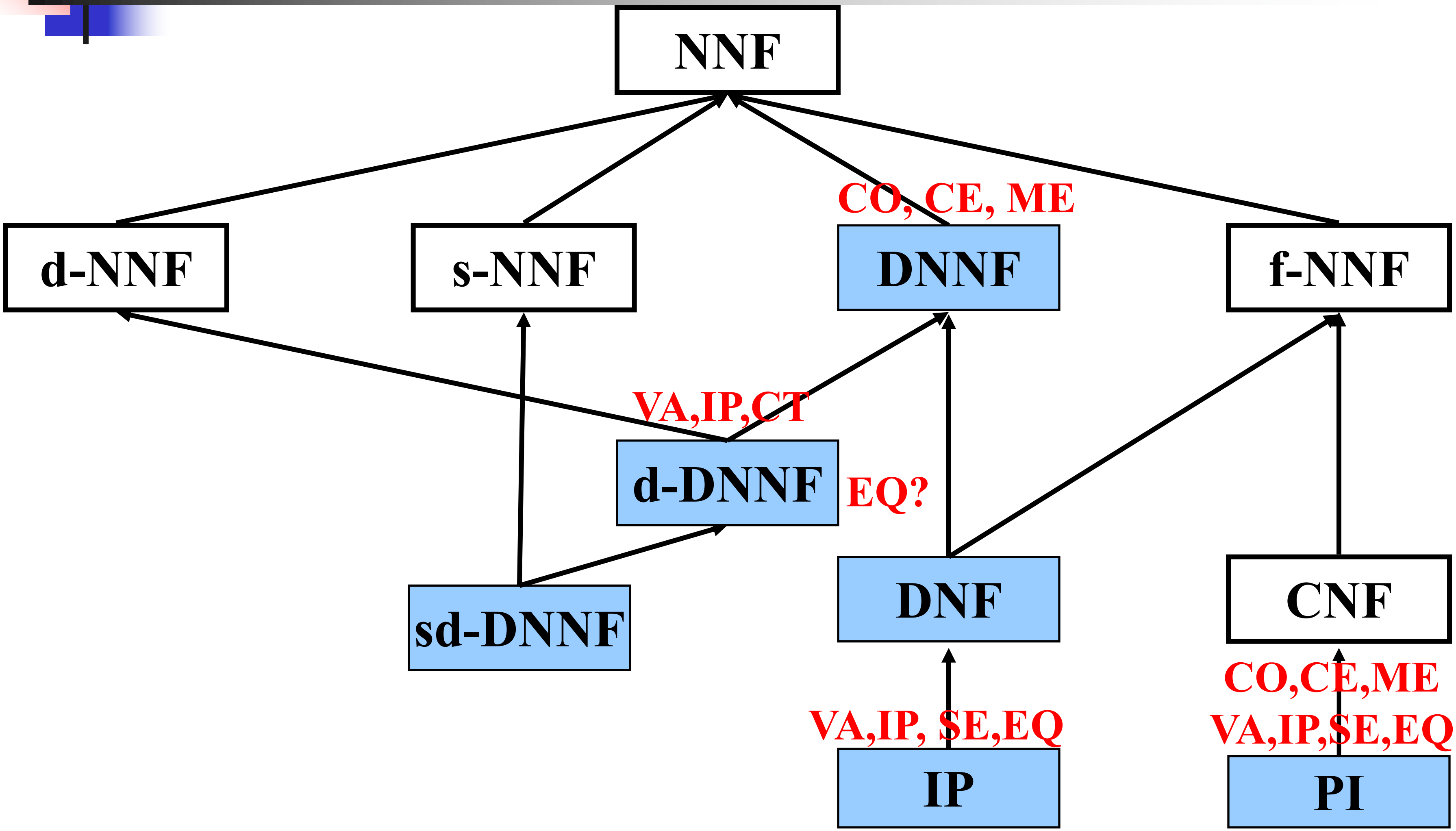
- DNF:

$$(A \wedge B) \vee (\neg B \wedge C)$$

- IP:

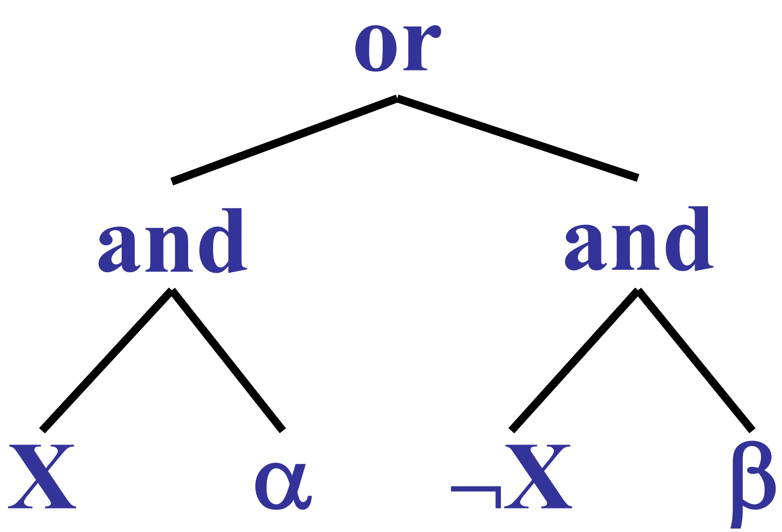
$$(A \wedge B) \vee (\neg B \wedge C) \vee (A \wedge C)$$

NNF Subsets



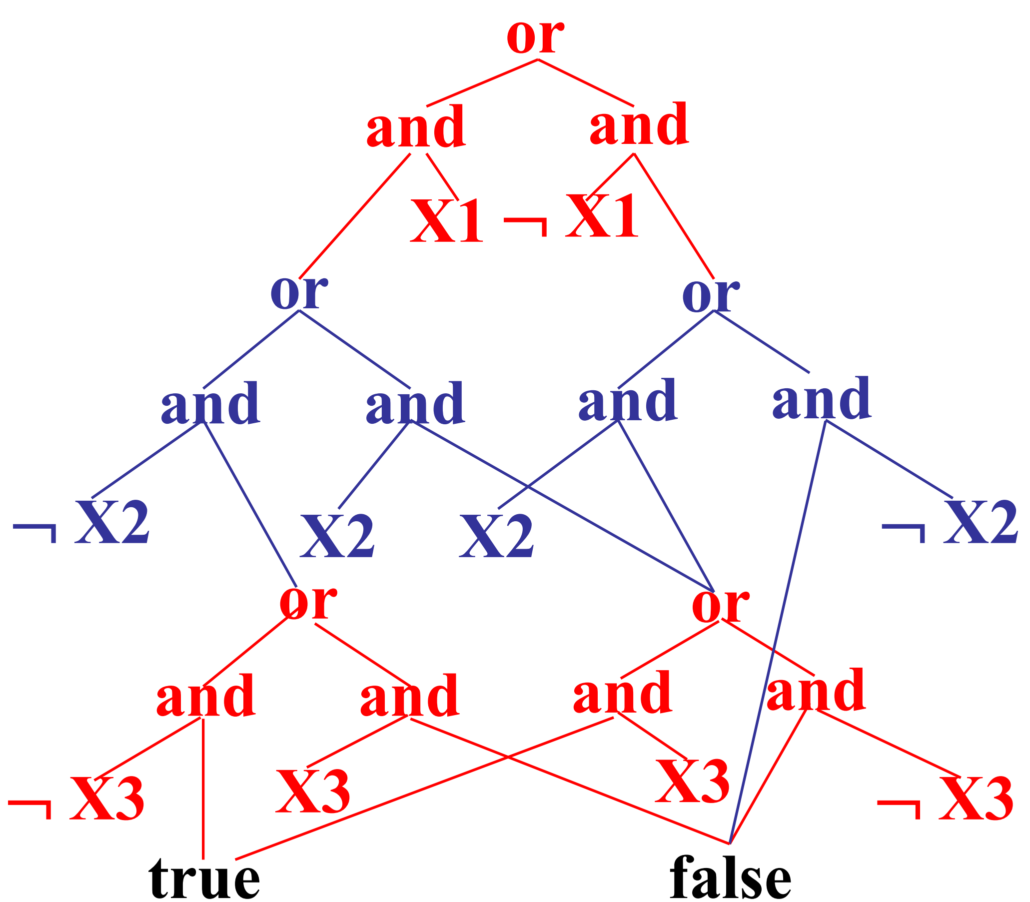
NNF
d-NNF
s-NNF
CO, CE, ME
DNNF
f-NNF
VA,IP,CT
d-DNNF
EQ?
sd-DNNF
DNF
CNF
CO,CE,ME
VA,IP, SE,EQ
VA,IP,SE,EQ
IP
PI

Decision

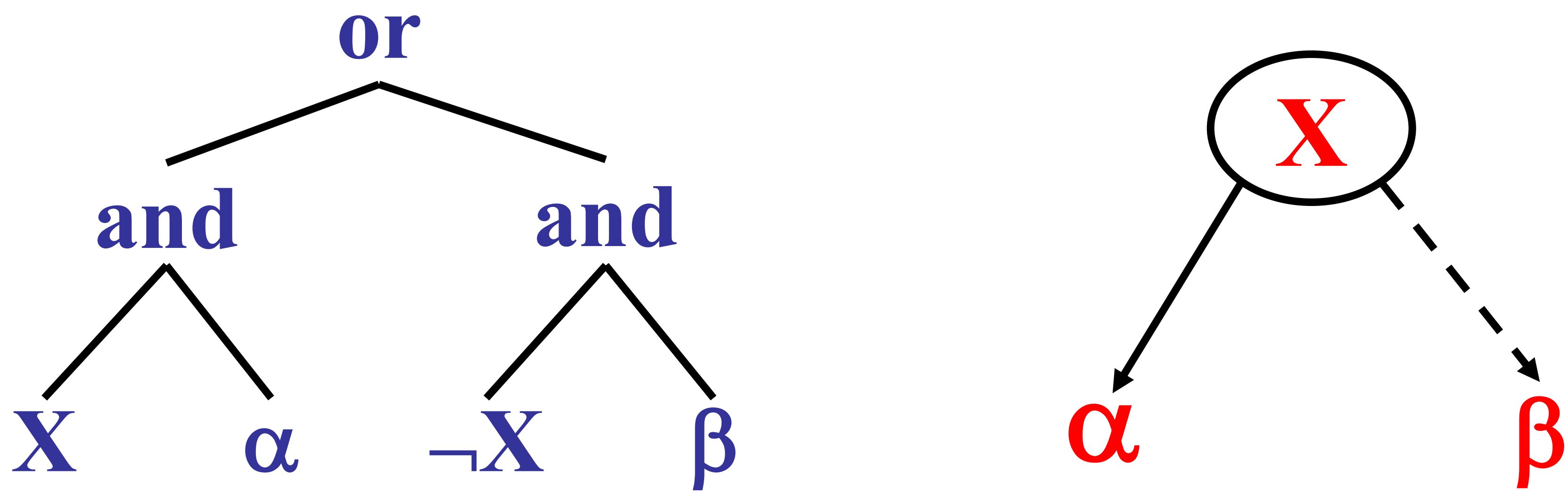


α, β: **Are decision nodes**

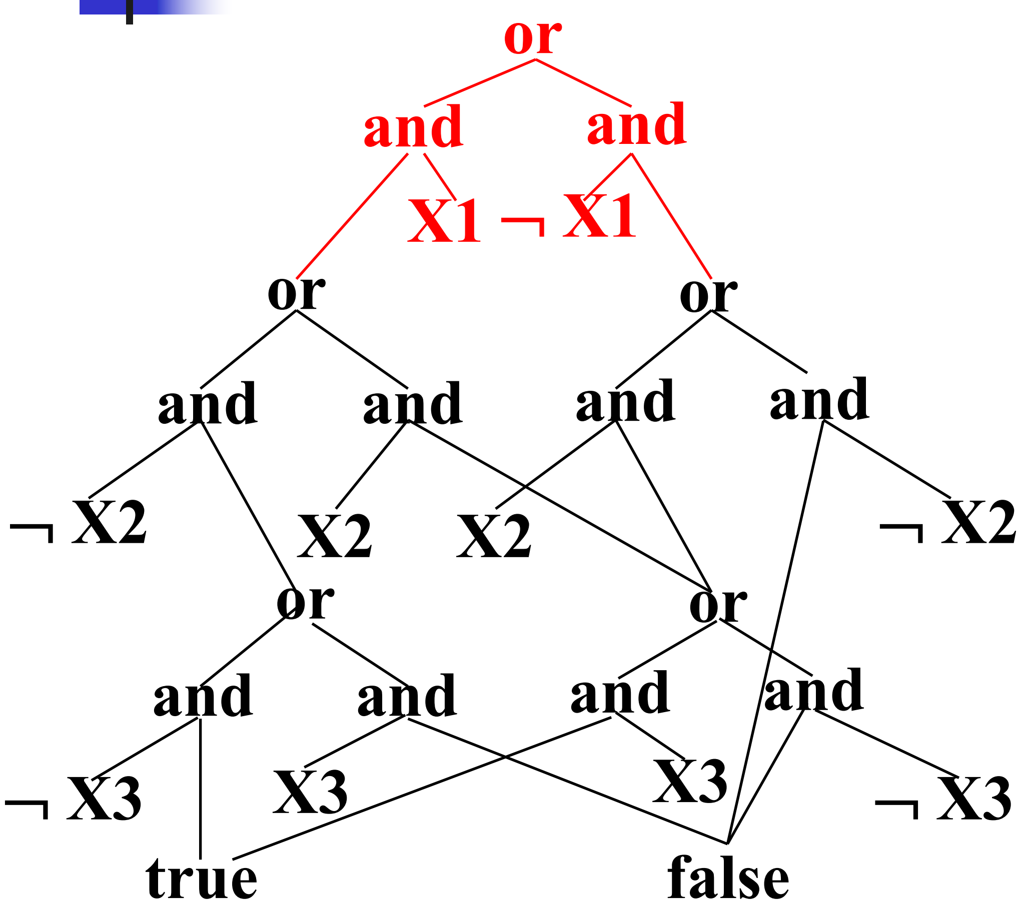
Decision

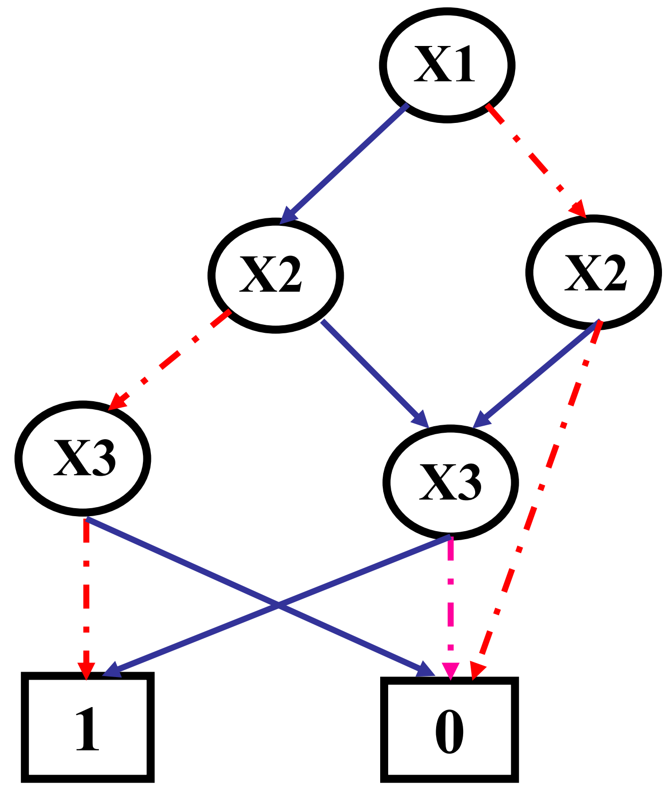

or
and
and
X1
¬ X1
or
or
and
and
and
and
¬ X2
X2
X2
¬ X2
or
or
and
and
and
and
¬ X3
X3
X3
¬ X3
true
false

Decision



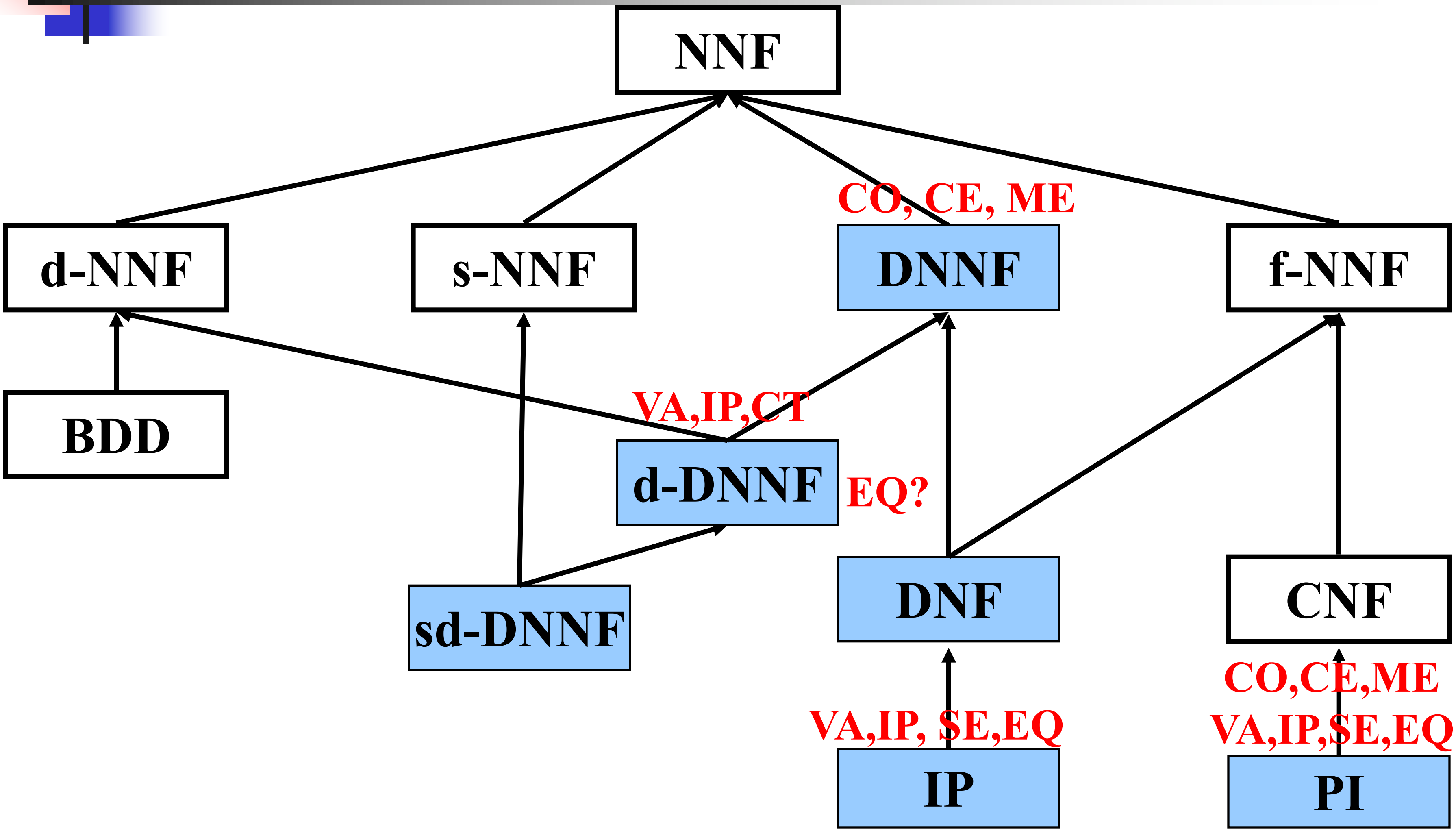
Binary Decision Diagrams (BDDs)



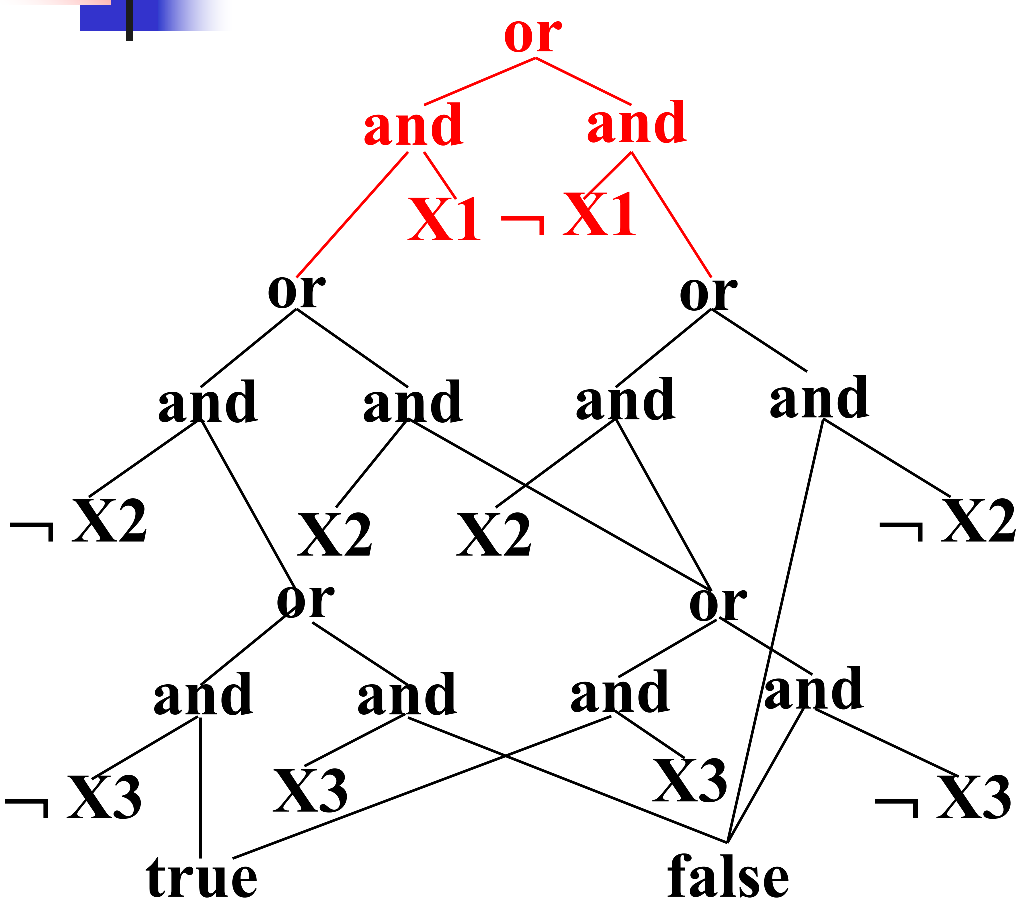


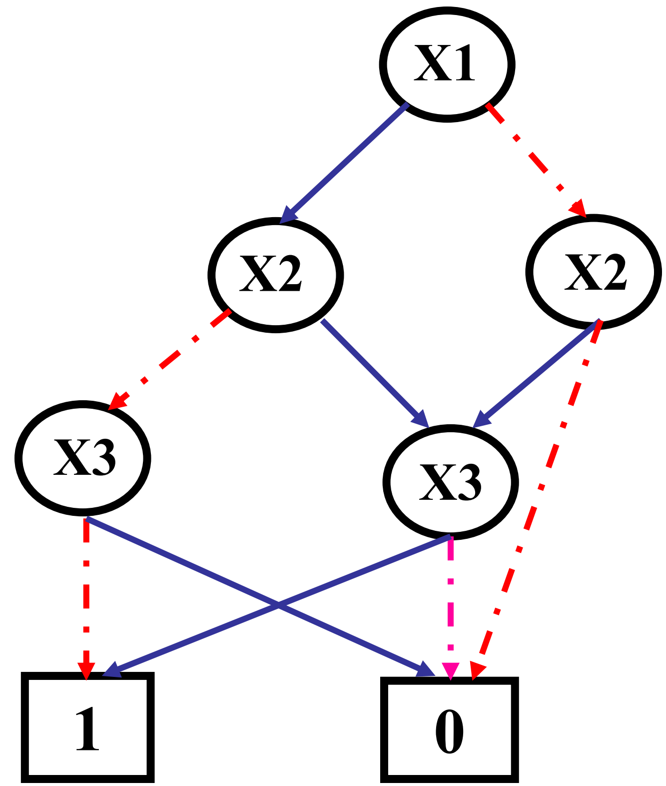
Decision implies determinism

NNF Subsets


NNF
d-NNF
s-NNF
CO, CE, ME
DNNF
f-NNF
BDD
VA,IP,CT
d-DNNF
EQ?
sd-DNNF
DNF
CNF
CO,CE,ME
VA,IP, SE,EQ
VA,IP,SE,EQ
IP
PI

Binary Decision Diagrams (BDDs)

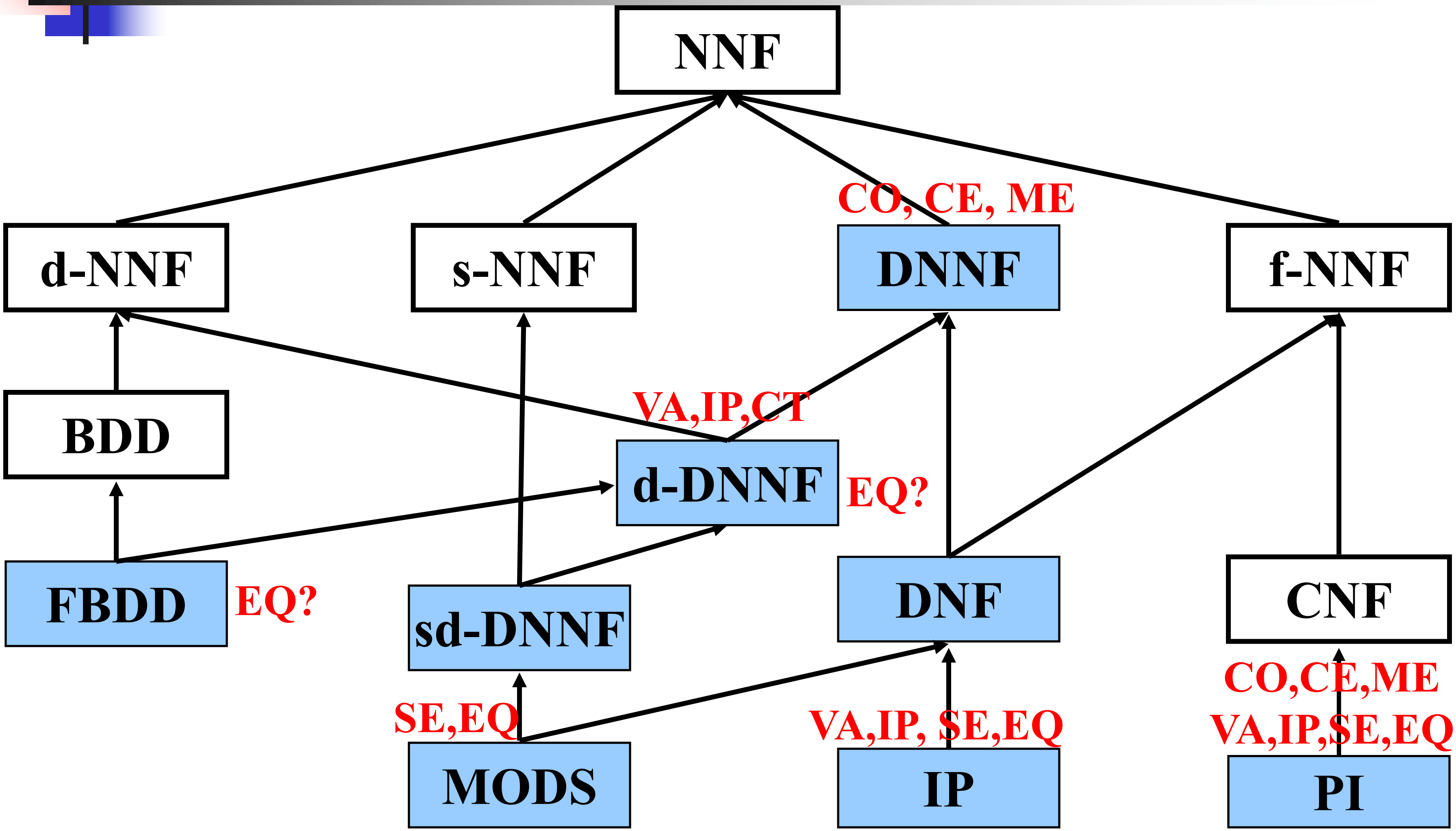




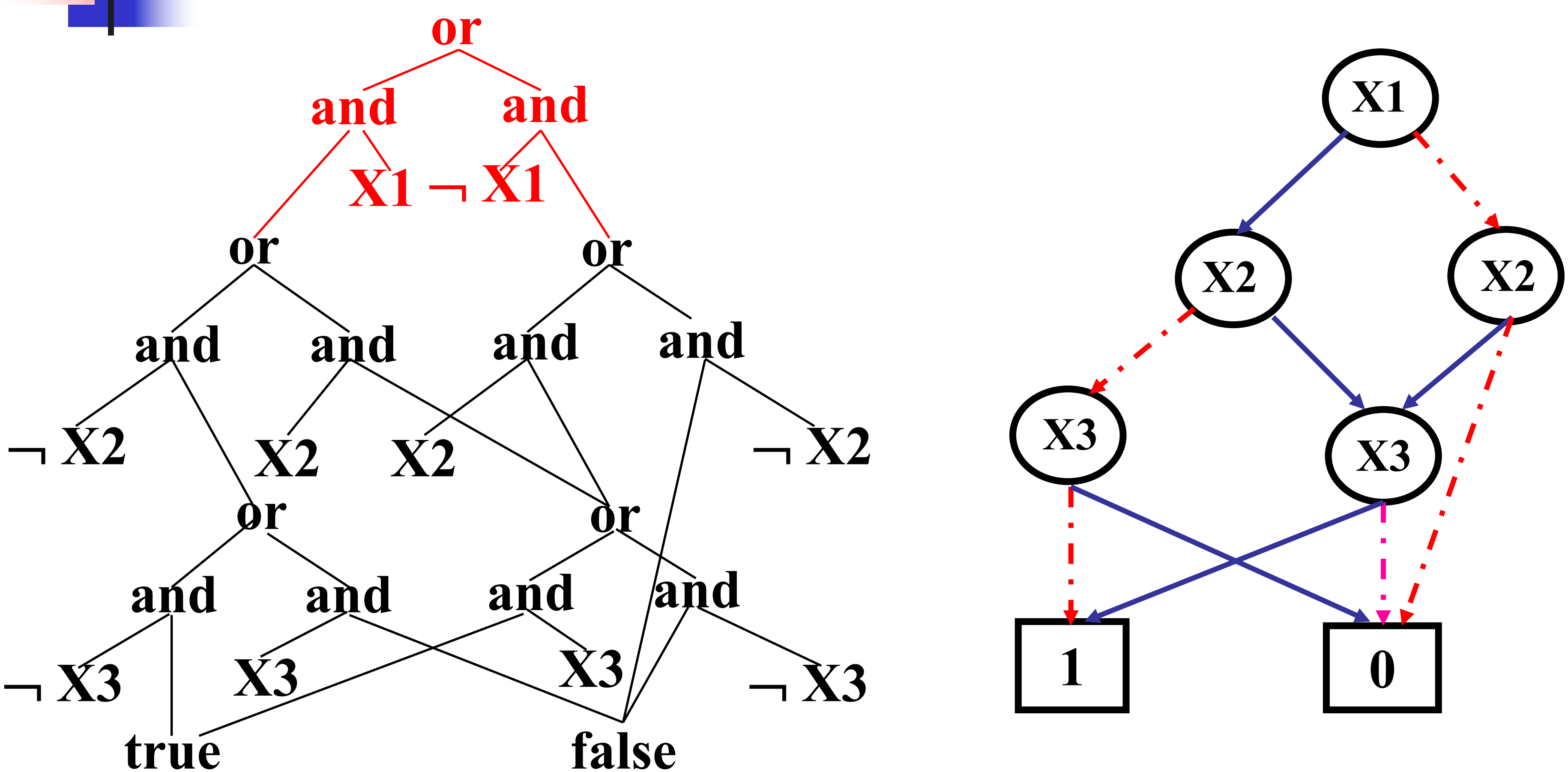
Decision + decomposability = FBDD

Test once property

NNF Subsets

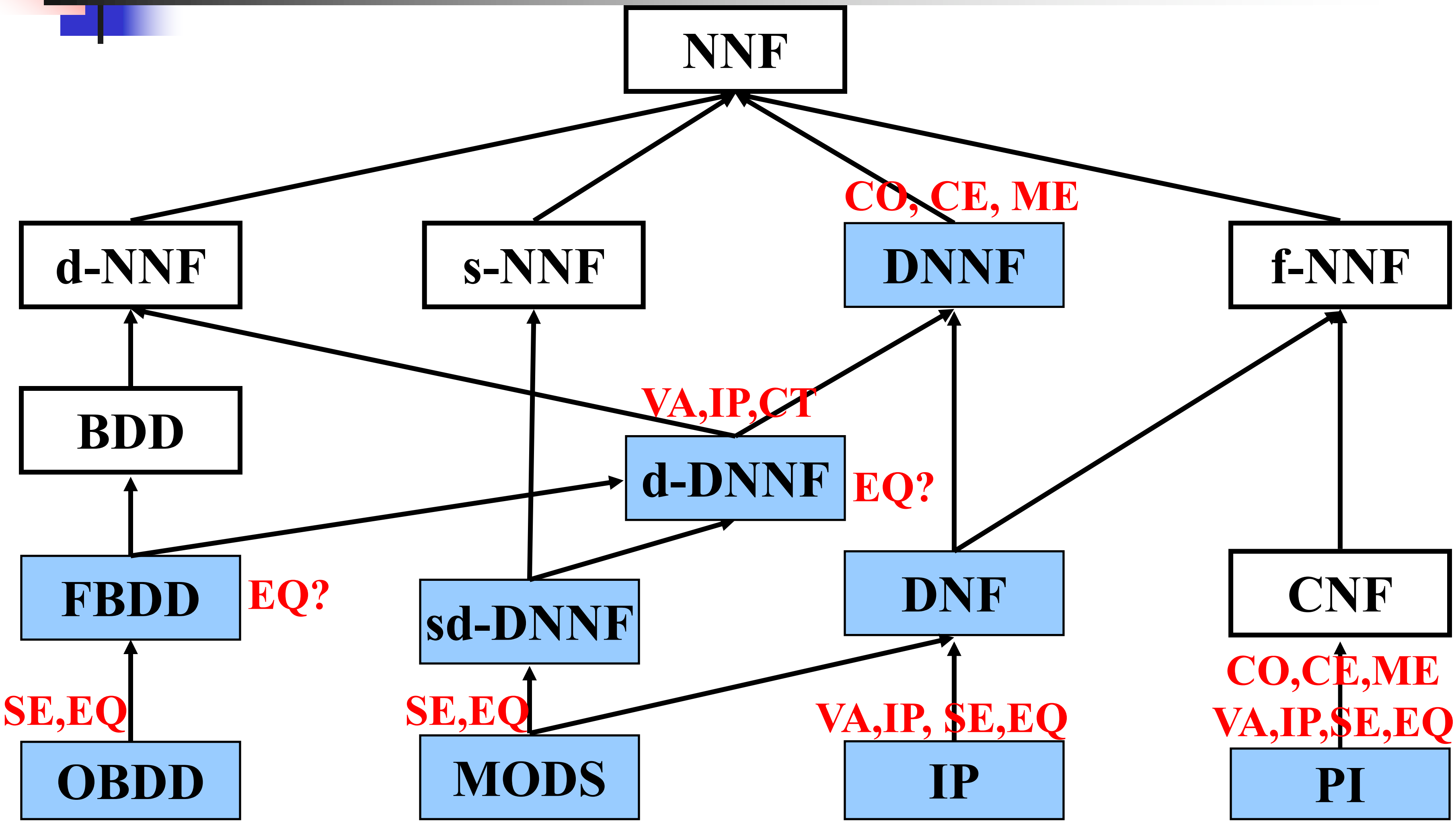

NNF
d-NNF
s-NNF
DNNF
f-NNF
CO, CE, ME
BDD
VA,IP,CT
d-DNNF
EQ?
FBDD
EQ?
sd-DNNF
DNF
CNF
CO,CE,ME
VA,IP,SE,EQ
SE,EQ
VA,IP, SE,EQ
MODS
IP
PI

Binary Decision Diagrams (BDDs)



Decision + decomposability + ordering = OBDD

NNF Subsets



NNF
d-NNF
s-NNF
CO, CE, ME
DNNF
f-NNF
BDD
VA,IP,CT
d-DNNF
EQ?
FBDD
EQ?
sd-DNNF
DNF
CNF
CO,CE,ME
VA,IP,SE,EQ
SE,EQ
SE,EQ
VA,IP, SE,EQ
OBDD
MODS
IP
PI

Language Succinctness

L1 at least as succinct as L2

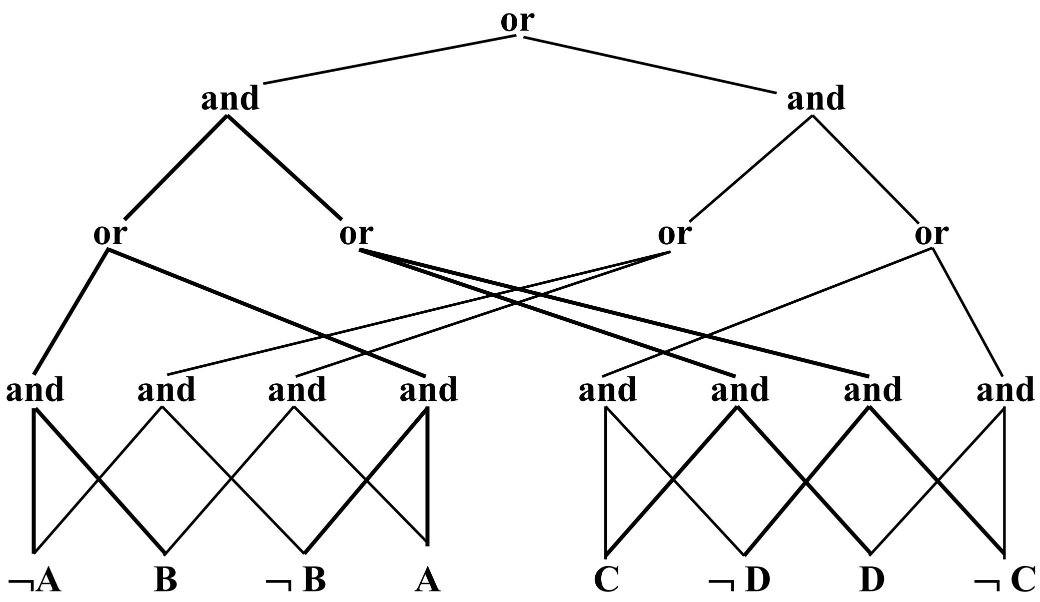
L1 <= L2

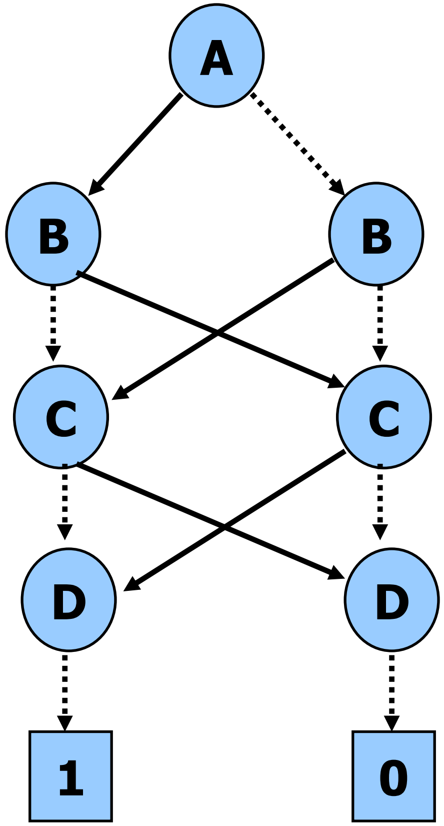
Size p(n) Size n

L1 is more succinct than L2

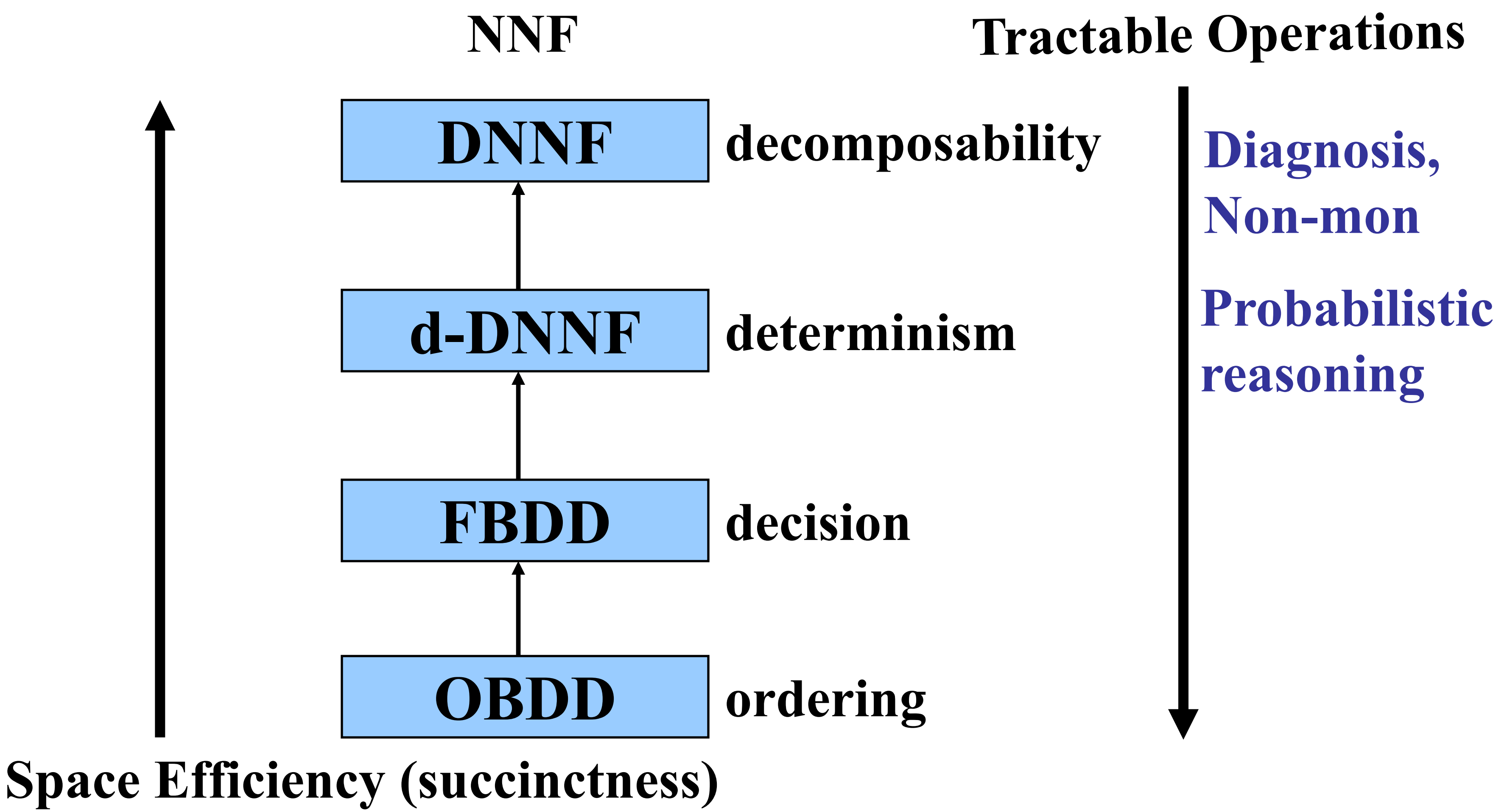
L1 < L2

Odd Parity Function





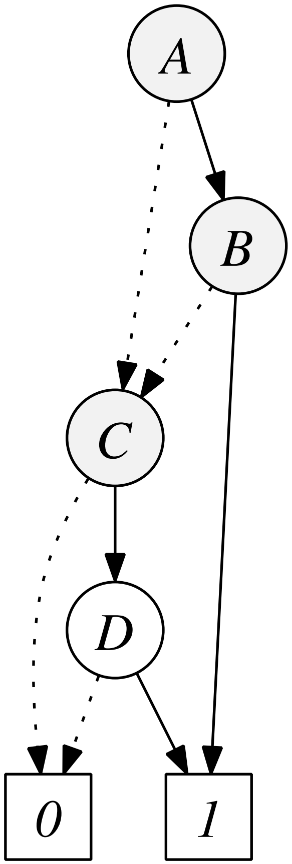
Tractability & Succinctness



Separating Functions

- OBDD/FBDD:
  - Hidden weighted bit function hwb(x_1,..,x_n)
- DNNF/DNF:
  - odd parity function parity(x_1,..,x_n)
- DNNF/OBDD:
  - Distinct integers function distinct(x_1,..,x_n)

From OBDD to SDD



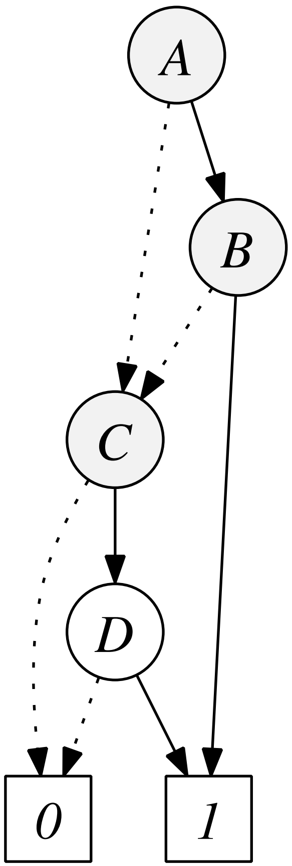
A

B

C

D

From OBDD to SDD



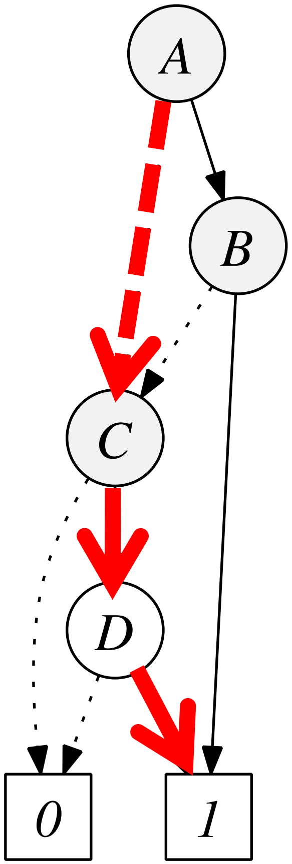
A =f

B =f

C =t

D =t

From OBDD to SDD



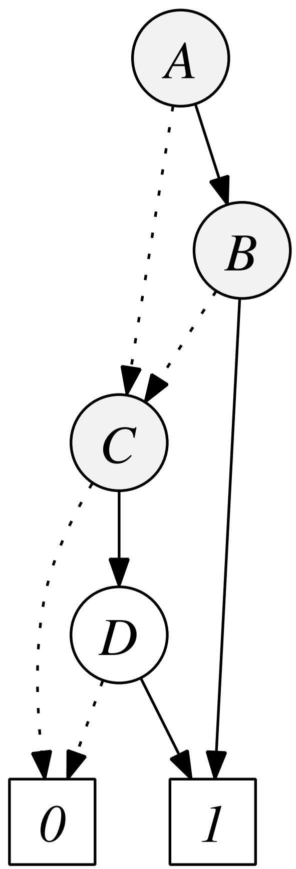
A=f

B=f

C=t

D=t

From OBDD to SDD

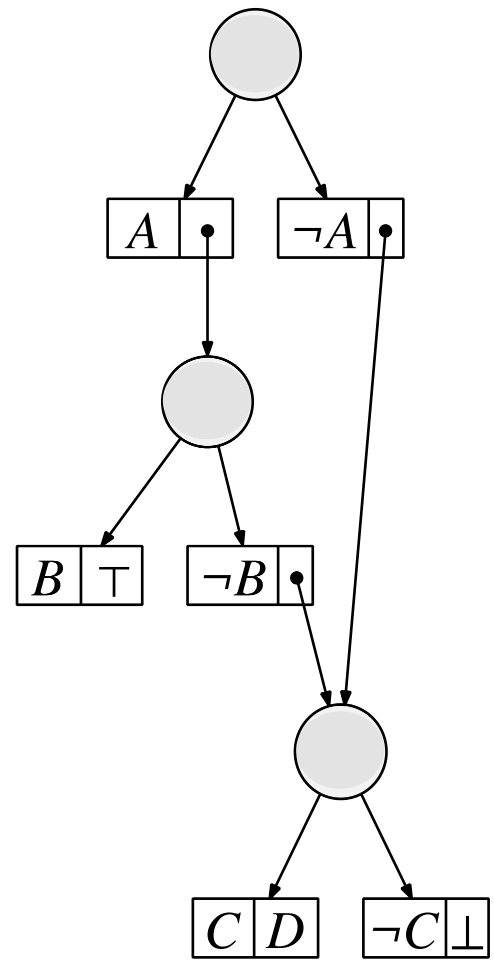


A

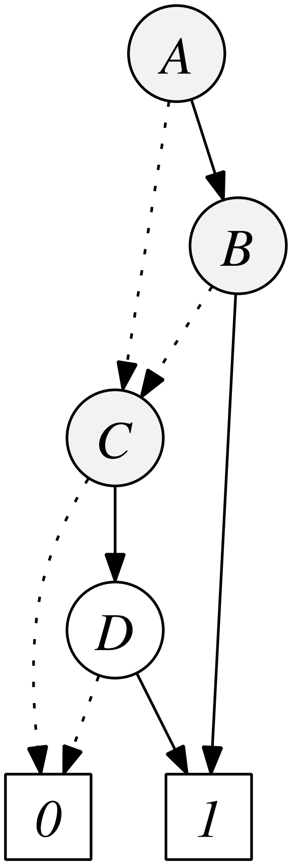
B

C

D



From OBDD to SDD

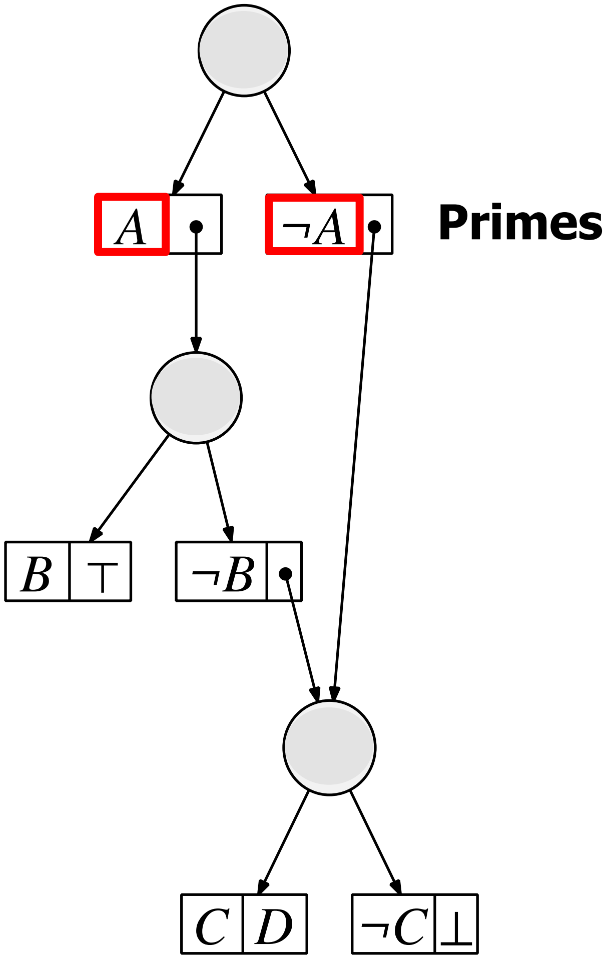


A

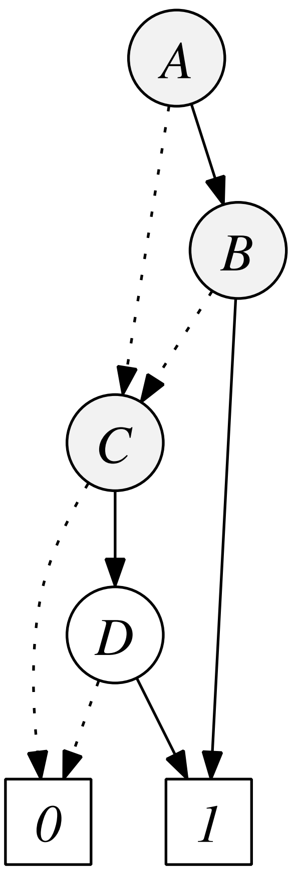
B

C

D



From OBDD to SDD

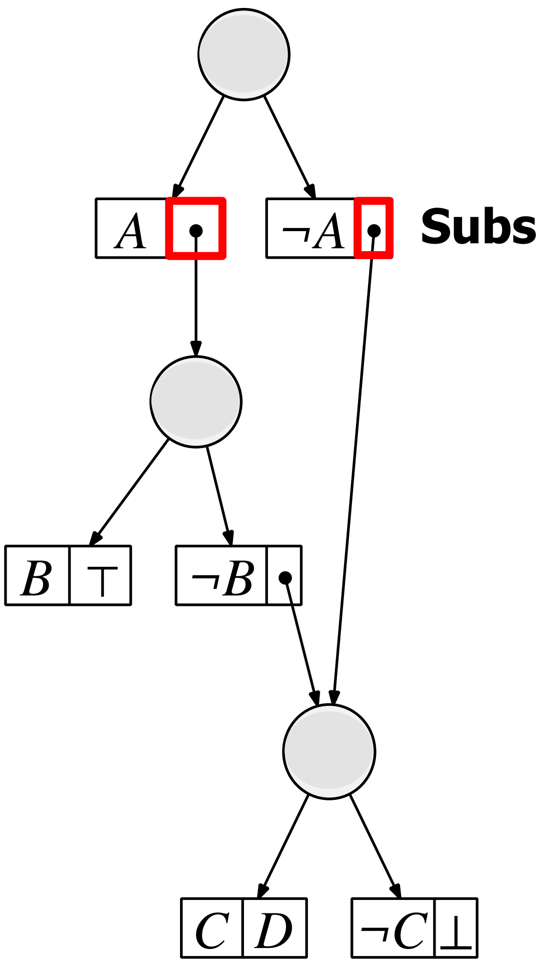


A

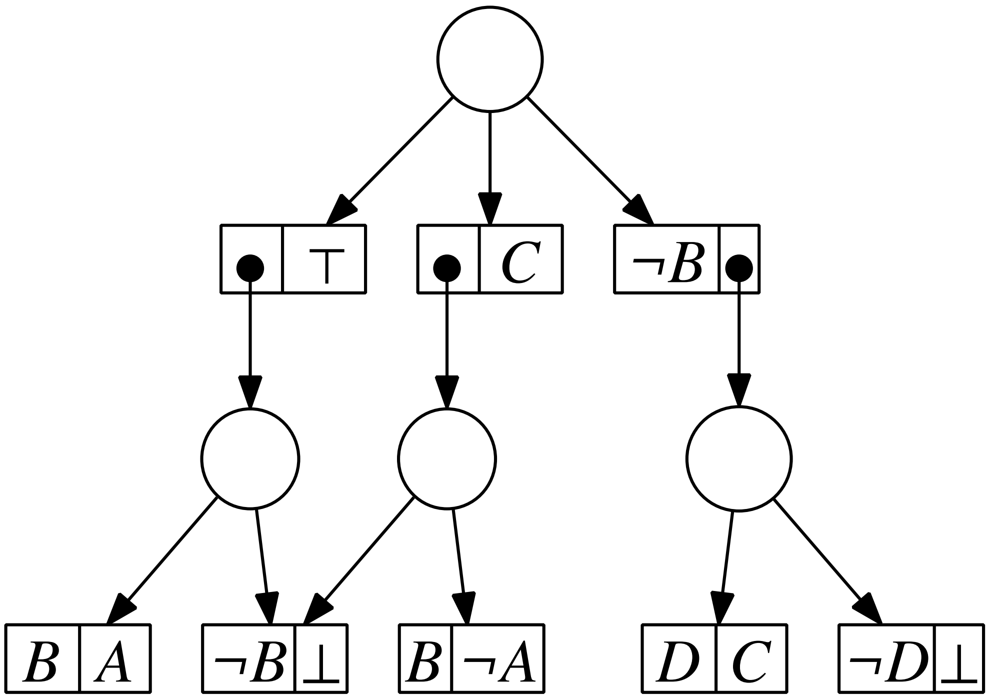
B

C

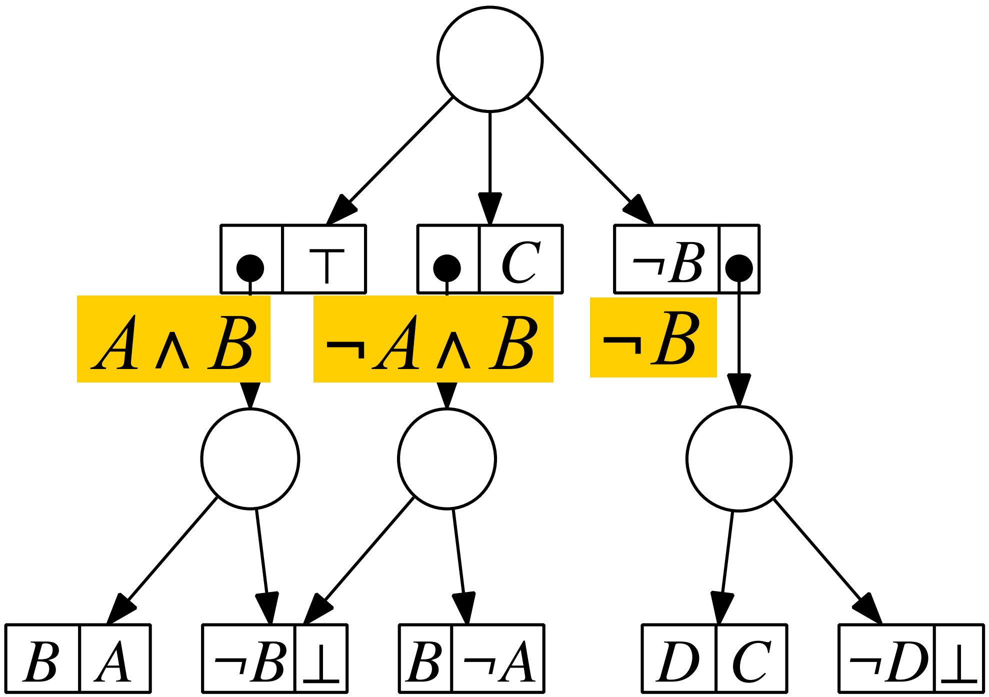
D



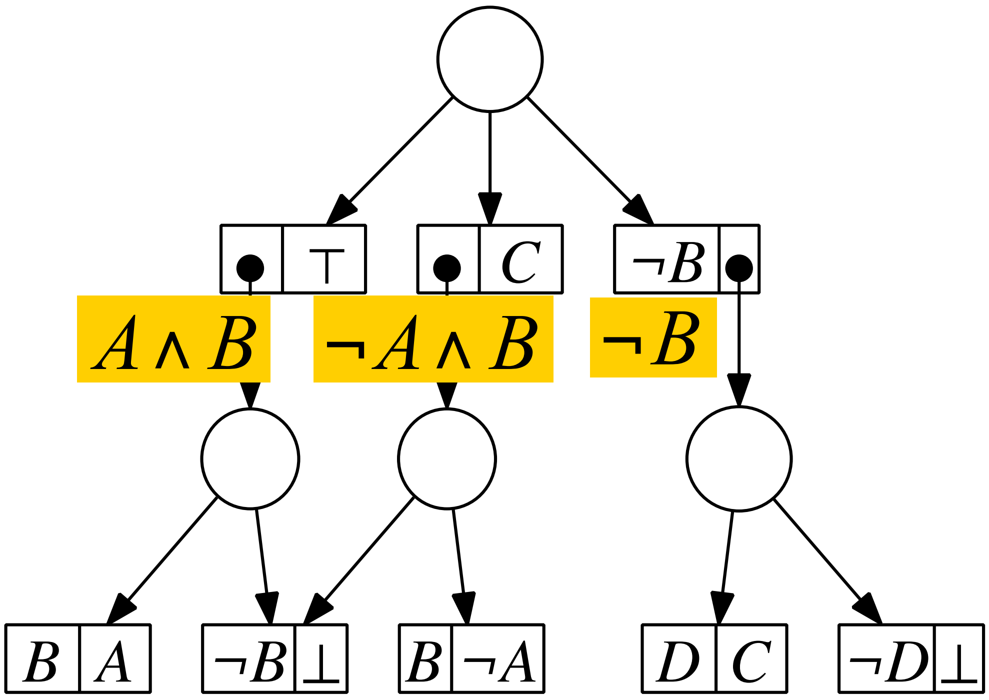
SDD: Sentential Decision Diagram

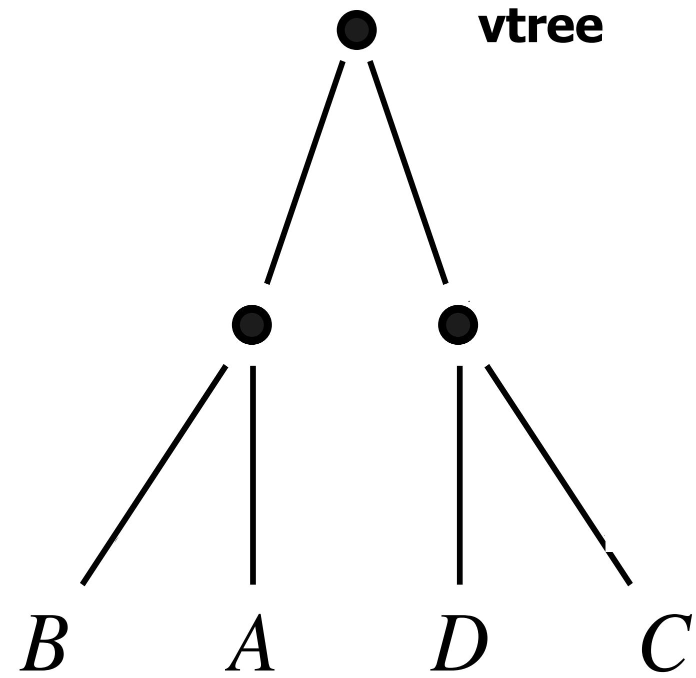


SDD:
Sentential Decision Diagram

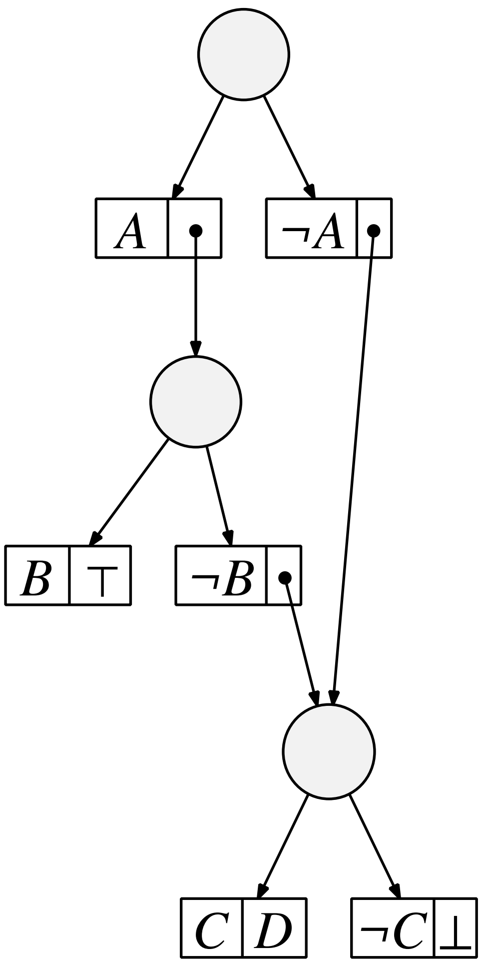


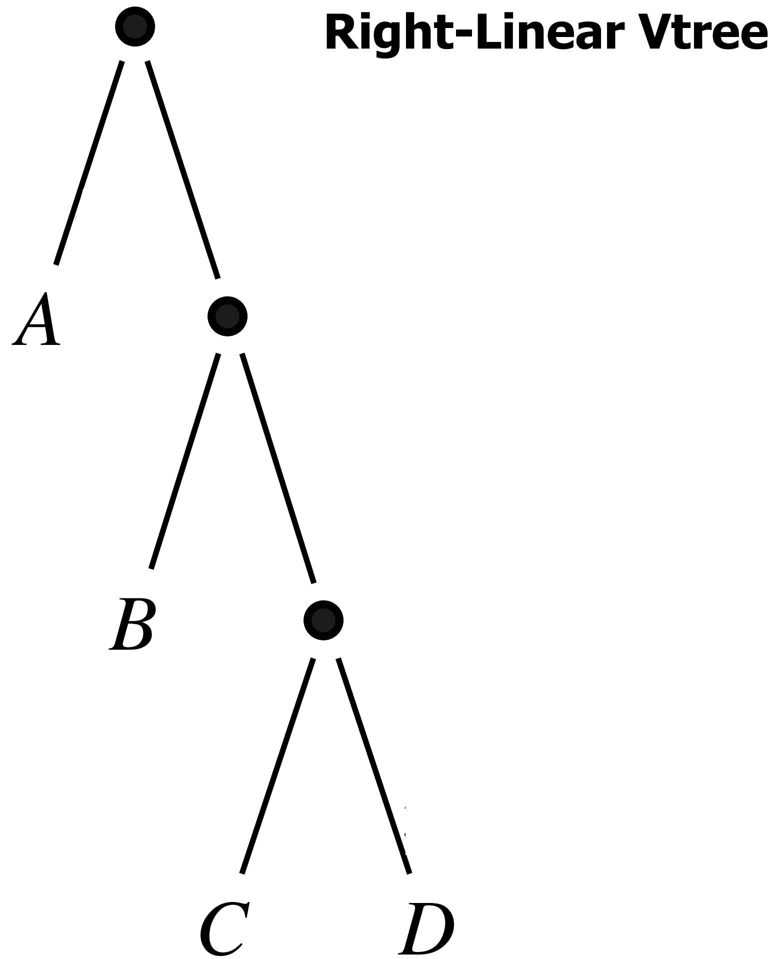
SDD:
Sentential Decision Diagram





OBDD as SDD





X-Partitions: The insight!

- Write function f(**X**,**Y**) as

  $h_1(\mathbf{X})\, g_1(\mathbf{Y}) + \ldots + h_n(\mathbf{X})\, g_n(\mathbf{Y})$

  such that:

  $h_1(\mathbf{X}), \ldots, h_n(\mathbf{X})$ is a partition

  (mutually exclusive and exhaustive; h_i <> false)

X-Partitions

- **X**-partition

  $h_1(\mathbf{X})\, g_1(\mathbf{Y}) + \ldots + h_n(\mathbf{X})\, g_n(\mathbf{Y})$

  written as $\{\, (h_1, g_1), \ldots, (h_n, g_n) \,\}$

- We call h_i <u>primes</u>, g_i <u>subs</u>

Compression & Canonicity

- **X**-partition is compressed if no equal subs

  $\{ (h_1, g_1), (h_2, g_2) \ldots, (h_n, g_n) \}$

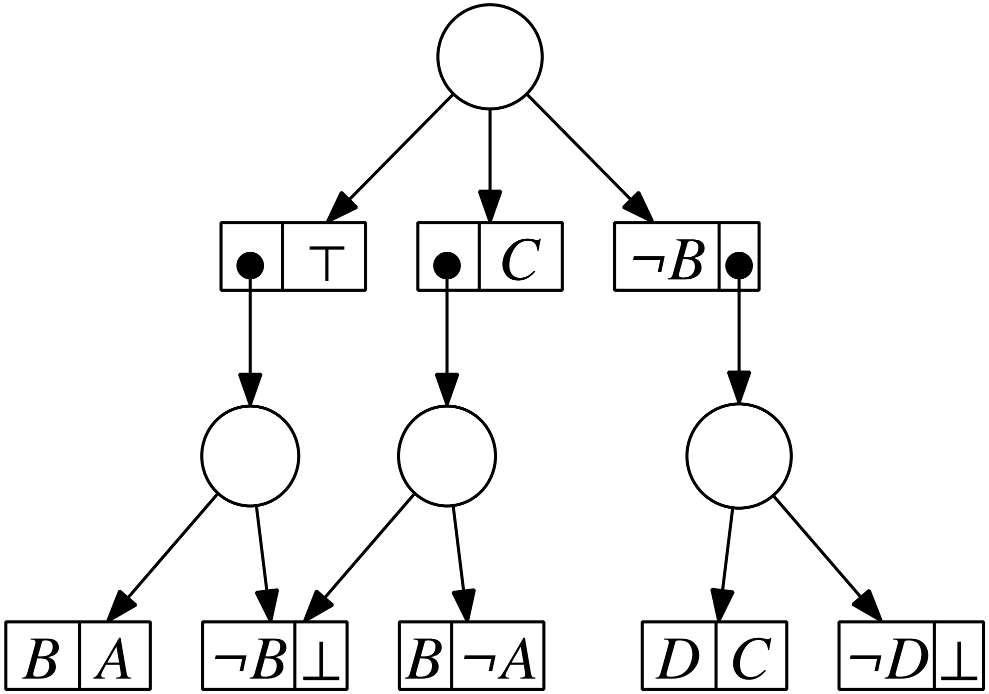
- If equal subs $g_1 = g_2$, compress to:

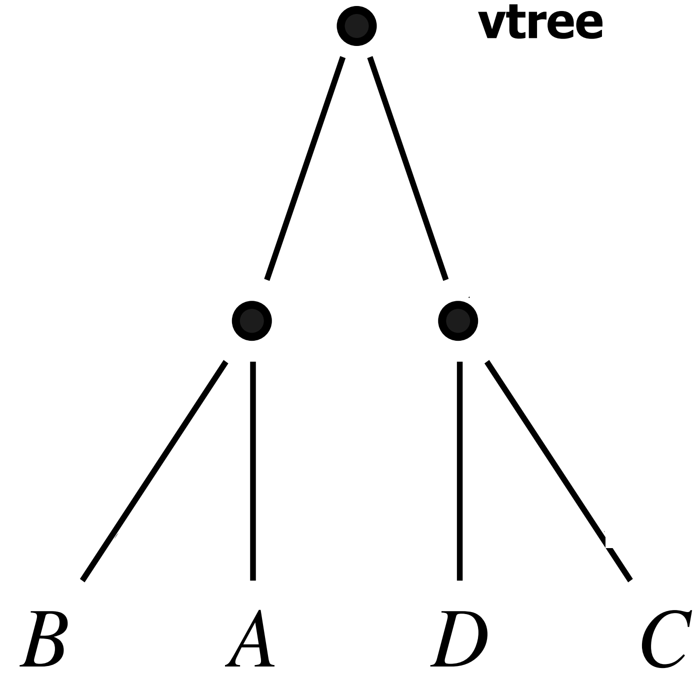
  $\{ (h_1 + h_2, g_1), \ldots, (h_n, g_n) \}$

- Every function f(**X**,**Y**) has a unique compressed **X**-partition

SDD

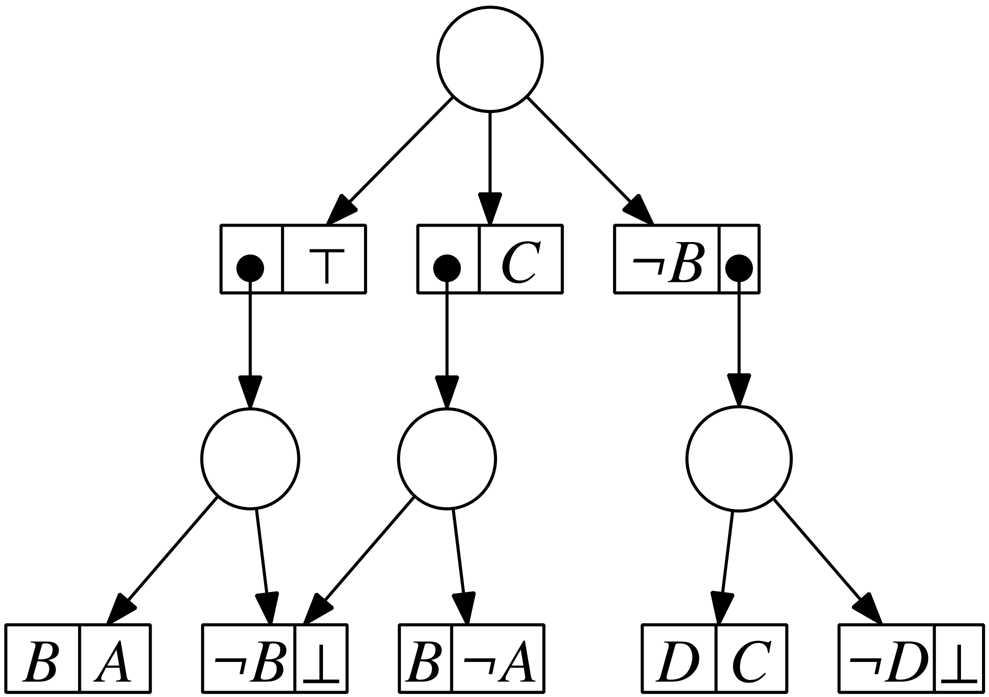
$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$

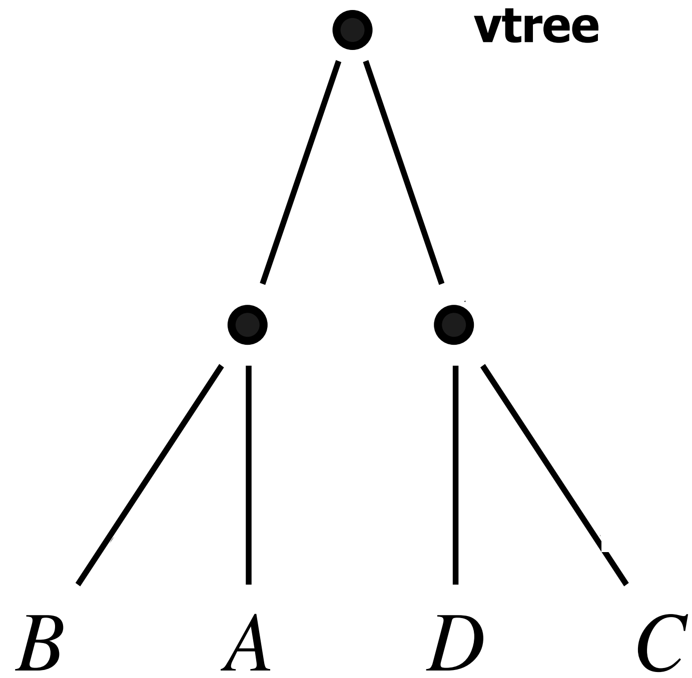




SDD

$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$





Compressed X-Partition of function f with X = {A,B}

$$\{(A \wedge B, true), (\neg A \wedge B, C), (\neg B, C \wedge D)\}$$

OBDDs are SDDs

- When **X=**{X} (single variable), an **X**-partition corresponds to a Shannon decomposition

- Primes: $\boxed{X, \neg X}$

- Subs: $\boxed{f \mid X, f \mid \neg X}$

- **X**-partition: $\boxed{\{(X, f \mid X), (\neg X, f \mid \neg X)\}}$

Polytime Apply Operation

- **X**-partition of *f*(**X**,**Y**): $\{(p_1,q_1),\ldots,(p_n,q_n)\}$
- **X**-partition of *g*(**X**,**Y**): $\{(r_1,s_1),\ldots,(r_m,s_m)\}$

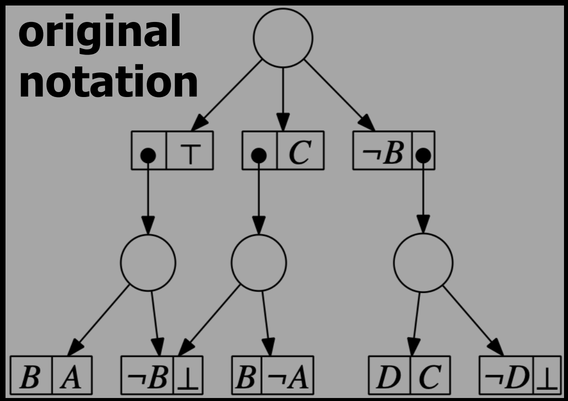
- **X**-partition of

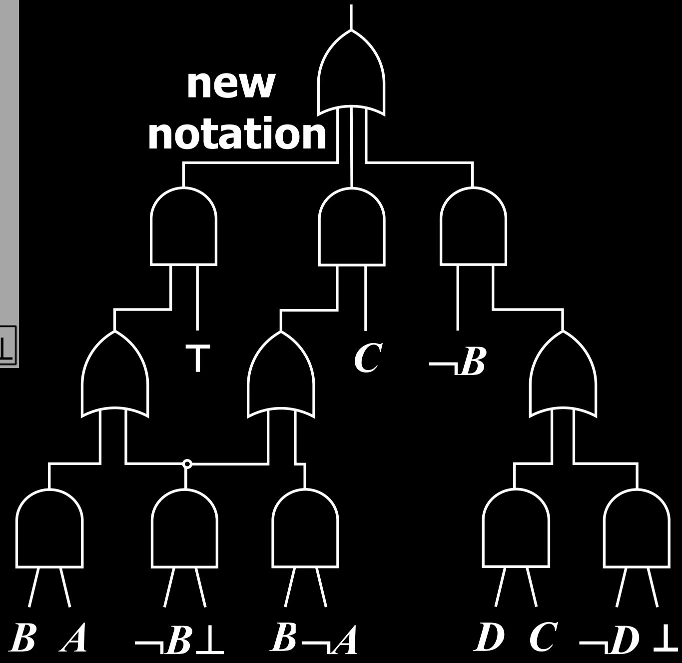
$$f \circ g = \{(p_i \wedge r_j, q_i \circ s_j) \mid p_i \wedge r_j \neq false\}$$

- Result may not be compressed

SDD vs OBDD

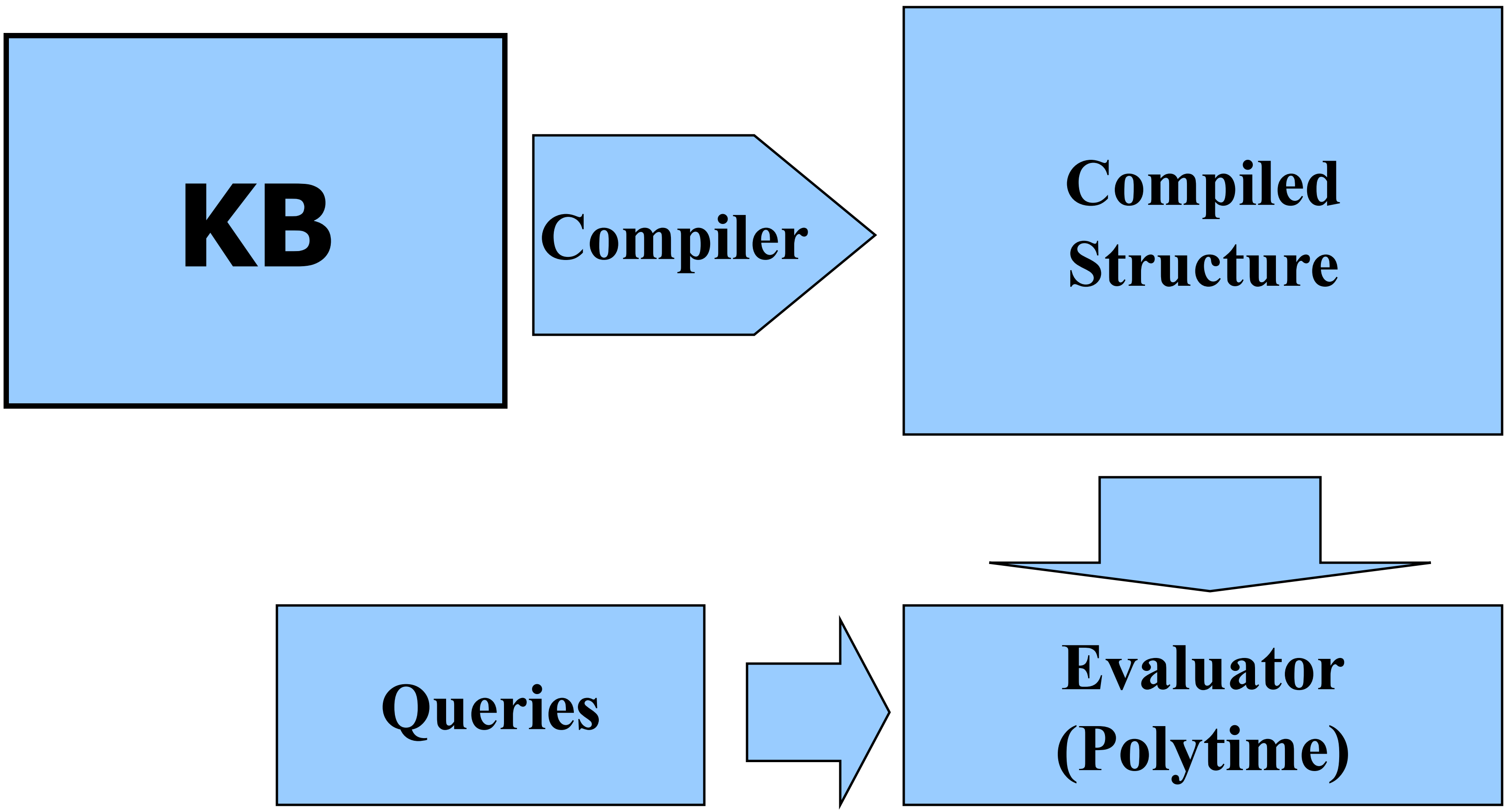
- SDD a strict superset of OBDD:
  - Characterized by trees, which include orders
  - Branch over sentences, which include literals
- SDDs maintain key properties of OBDDs:
  - Canonical when compressed
  - Polytime Apply Operation (no compression)
- SDDs: treewidth, OBDD: pathwidth
- SDDs more succinct than OBDDs


original
notation
• ⊤
• C
¬B •
B A
¬B ⊥
B ¬A
D C
¬D ⊥


new
notation
⊤
C
¬B
B A
¬B ⊥
B ¬A
D C
¬D ⊥

Operations

Knowledge Compilation



Queries

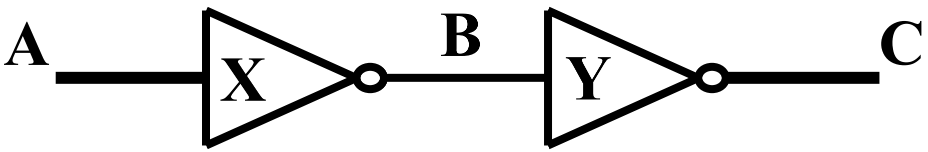
- Consistency (CO)
- Validity (VA)
- Sentential entailment (SE)
- Clausal entailment (CE): KB implies clause
- Implicant testing (IP): term implies KB
- Equivalence testing (EQ)
- Model Counting (CT)
- Model enumeration (ME)

Transformations

- Projection (existential quantification)
- Conditioning
- Conjoin
- Disjoin
- Negate

Decomposability

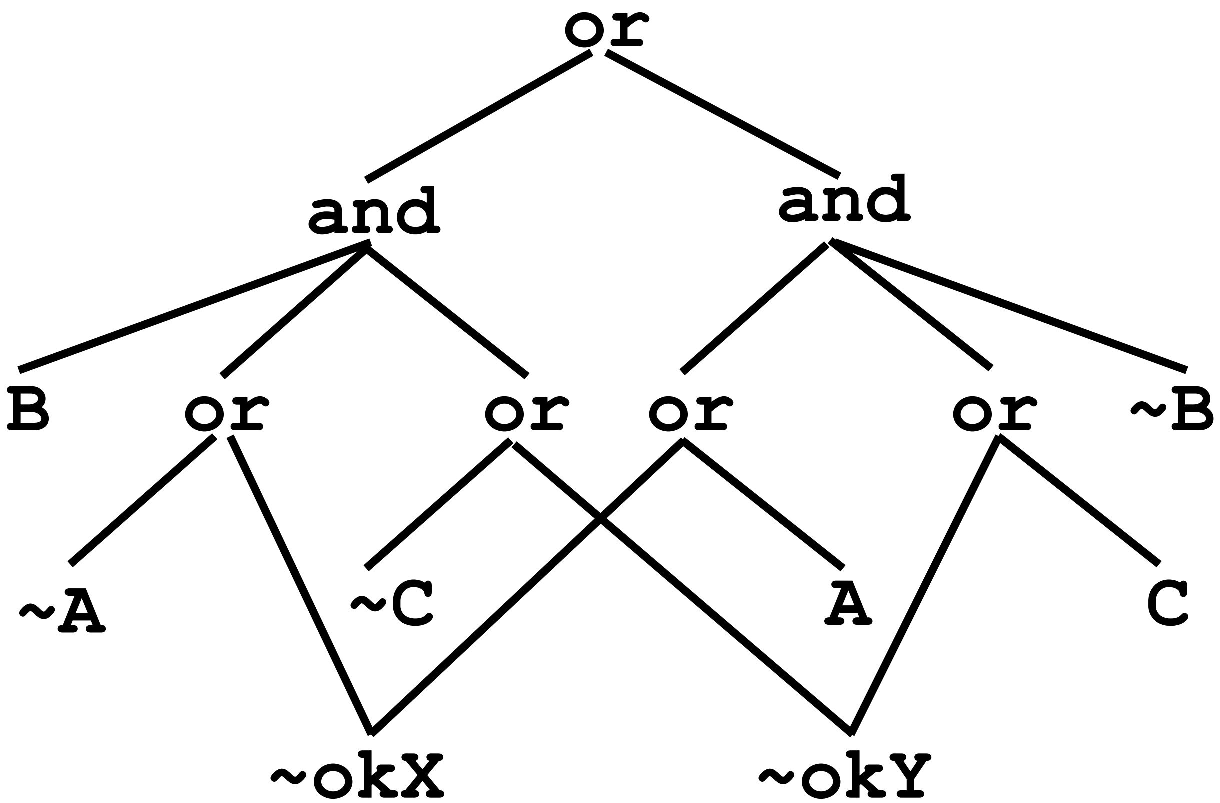
Example Knowledge Base




```
 A & okX => ~B
~A & okX =>  B

 B & okY => ~C
~B & okY =>  C
```

Decomposable

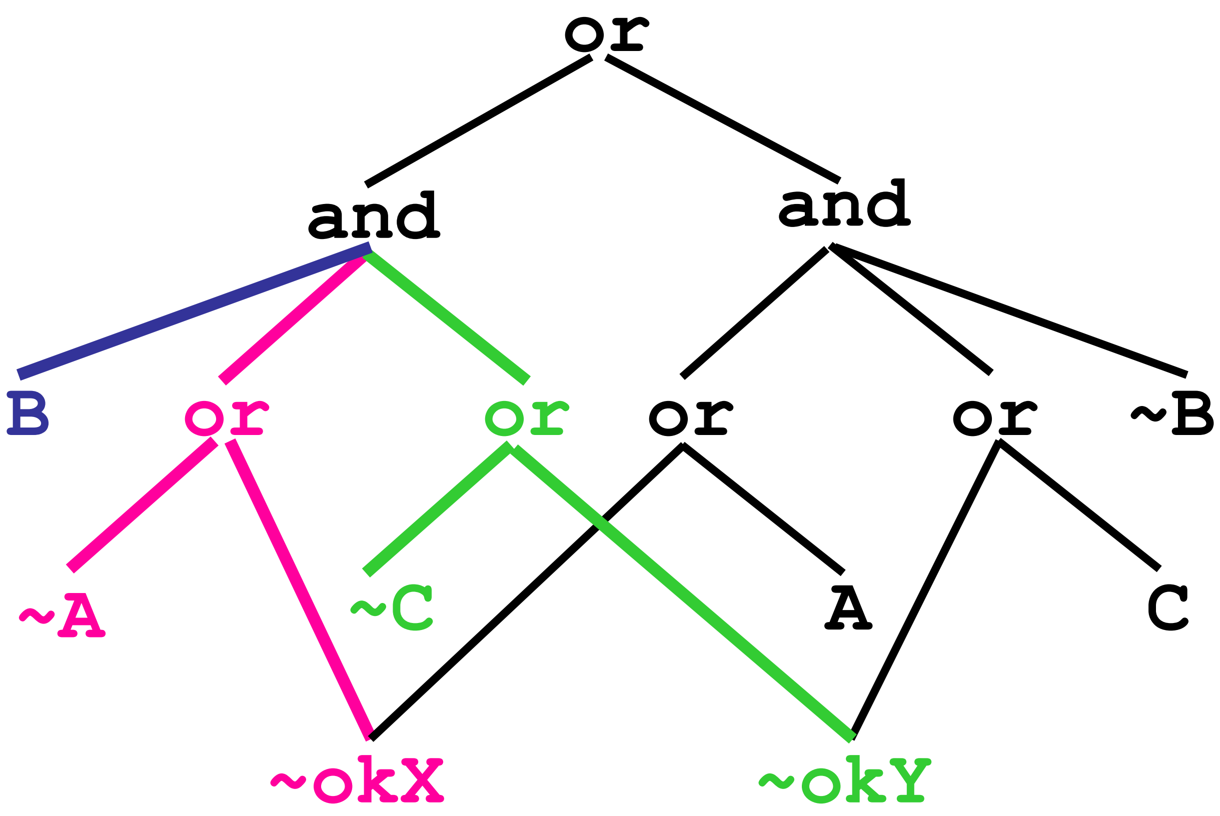


Decomposable

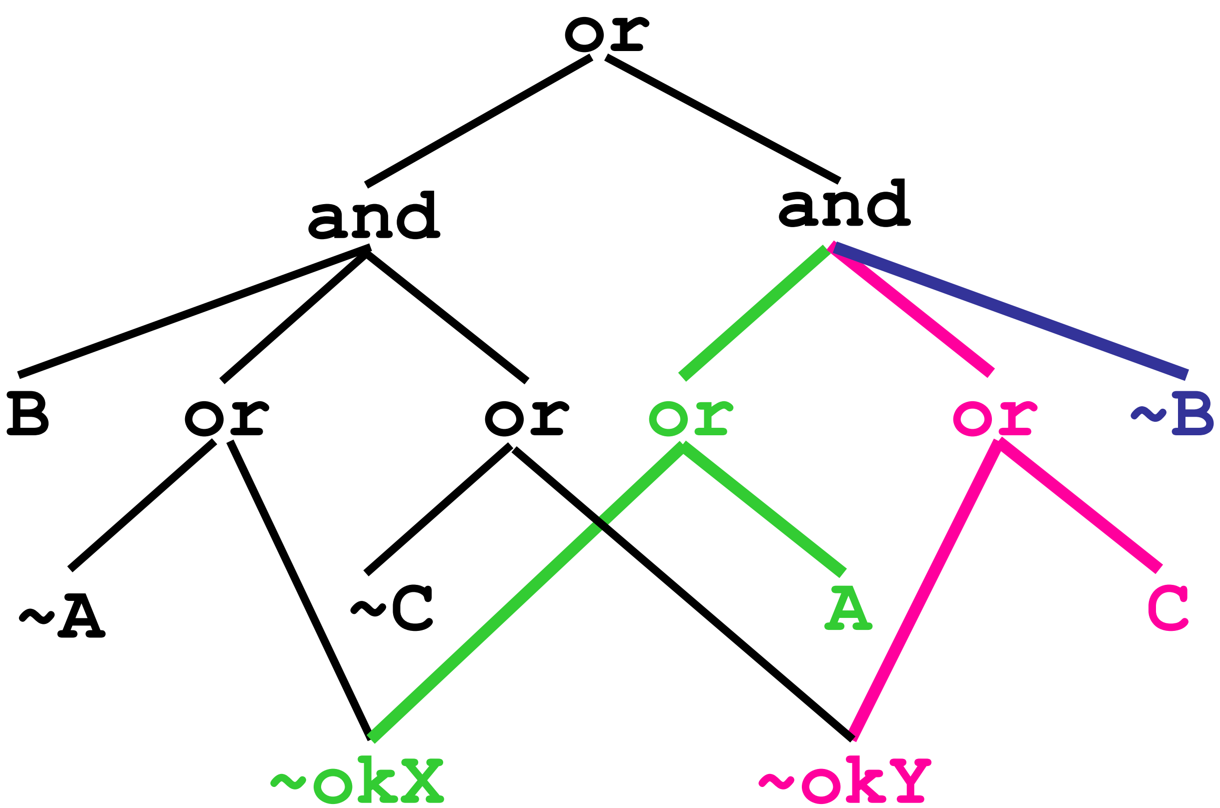
B

A, okX

C, okY



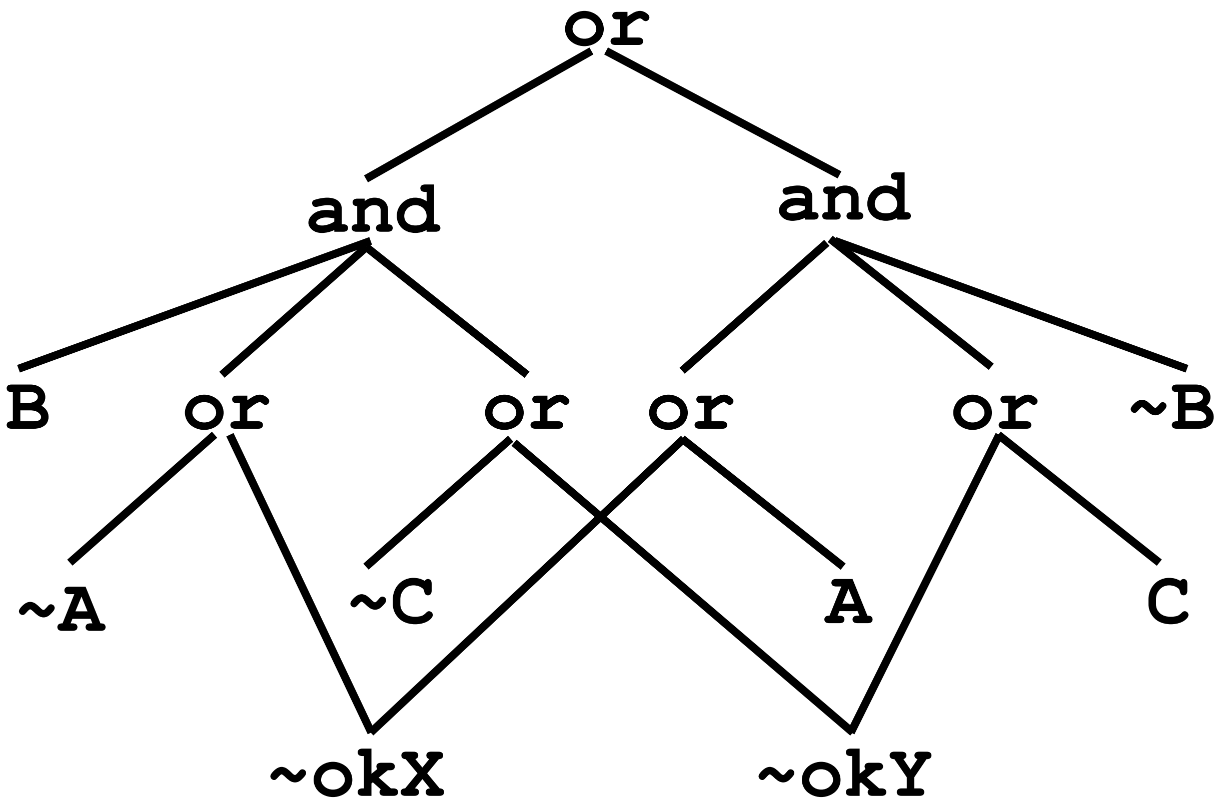
Decomposable



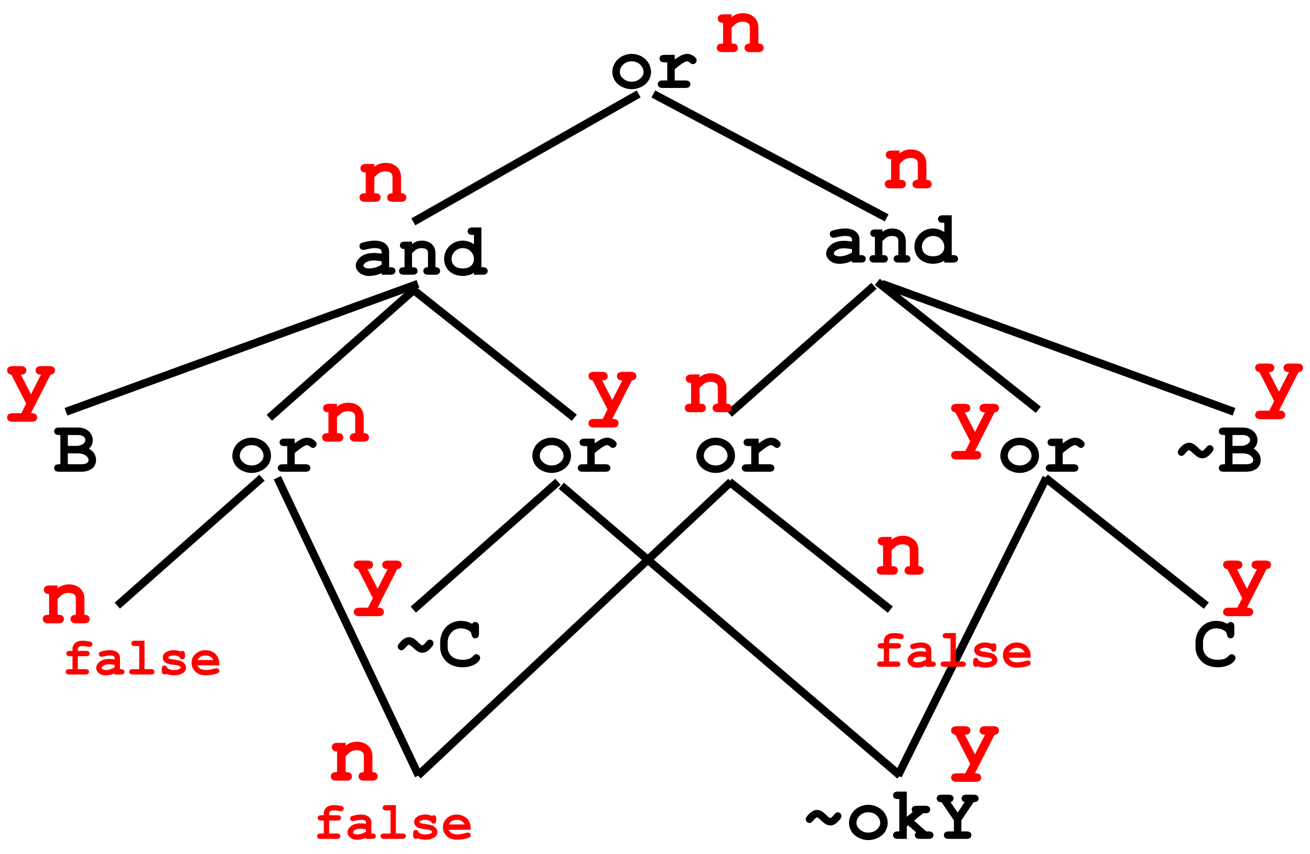
Satisfiability

- SAT(A or B) iff SAT(A) or SAT(B)
- SAT(A and B) iff SAT(A) and SAT(B)
- SAT(X) is true
- SAT(~X) is true
- SAT(True) is true
- SAT(False) is false

Satisfiability



Satisfiability

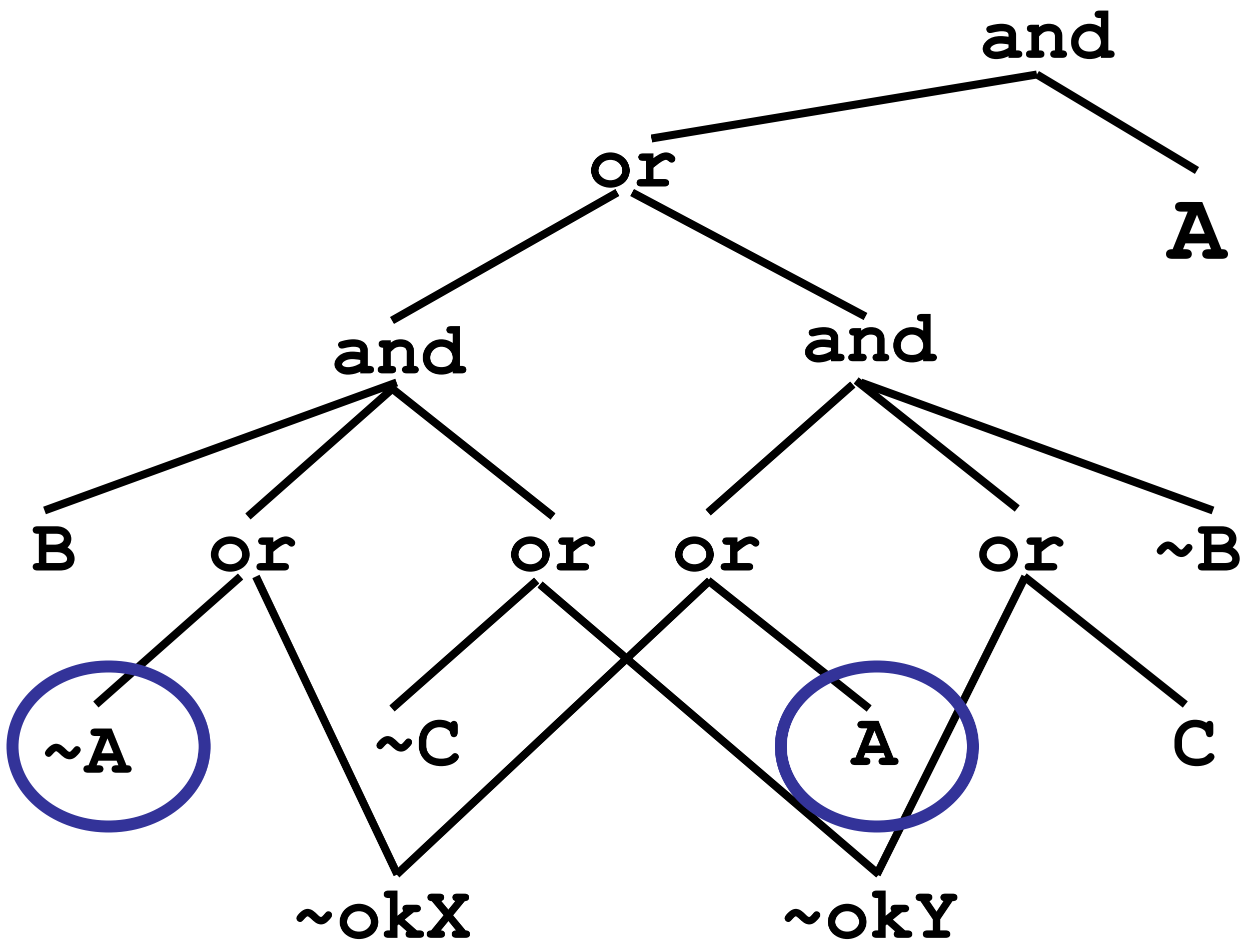


Clausal Entailment

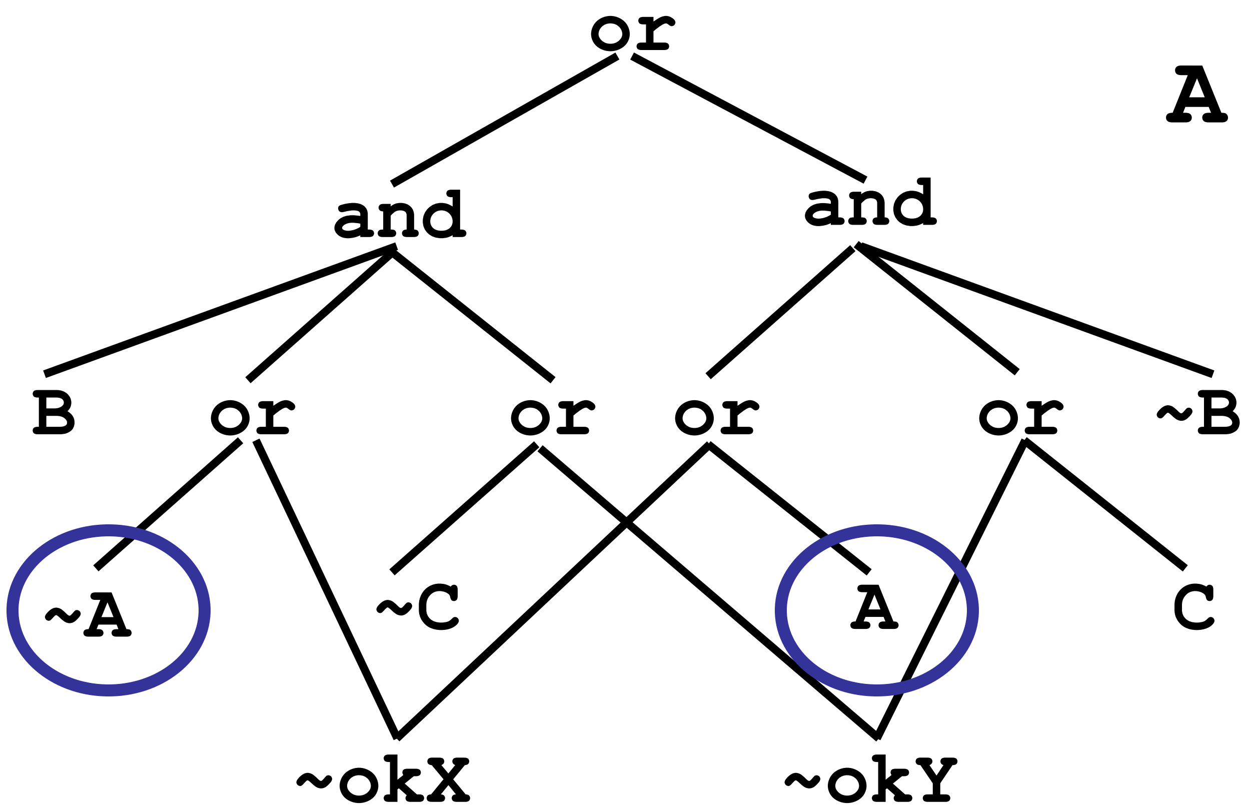
KB *entails* L_1 v L_2 v … v L_n ?

KB & $\sim L_1$ & $\sim L_2$ & … & $\sim L_n$ *SAT* ?

Literal Conjoin

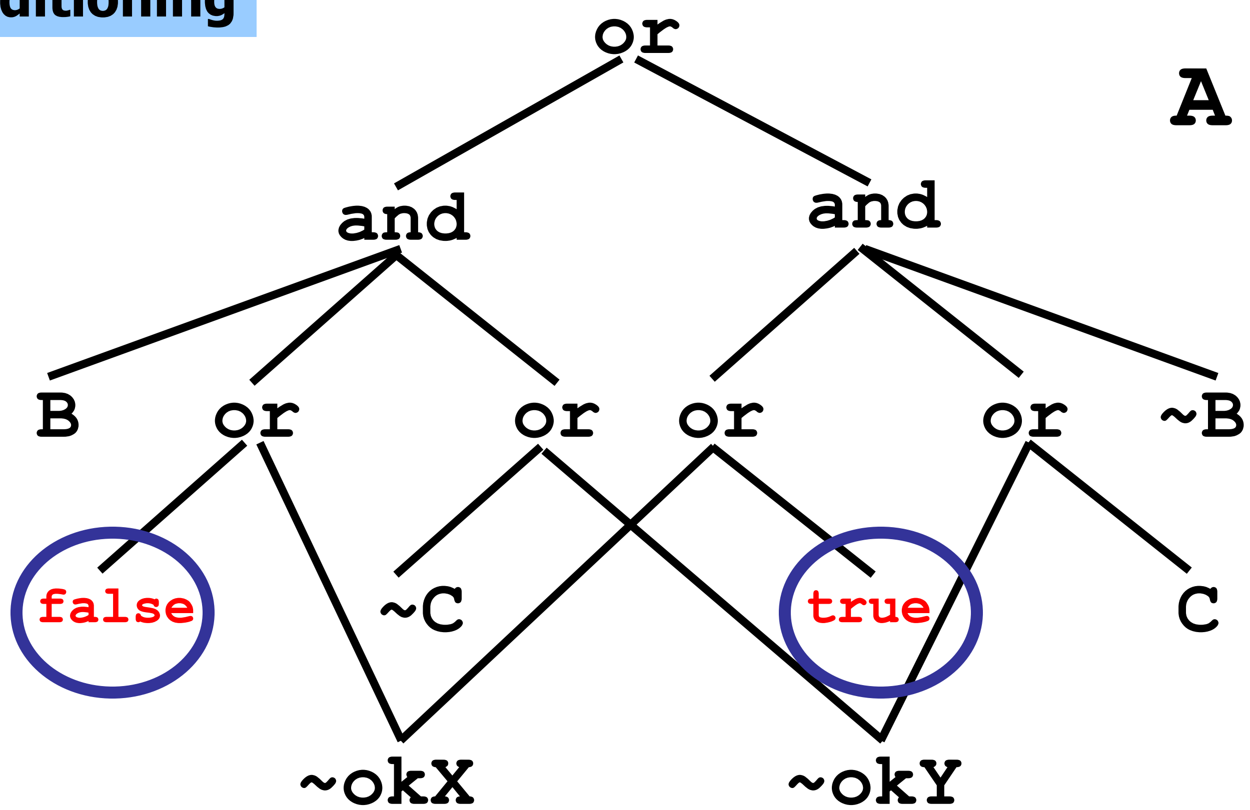


Literal Conjoin

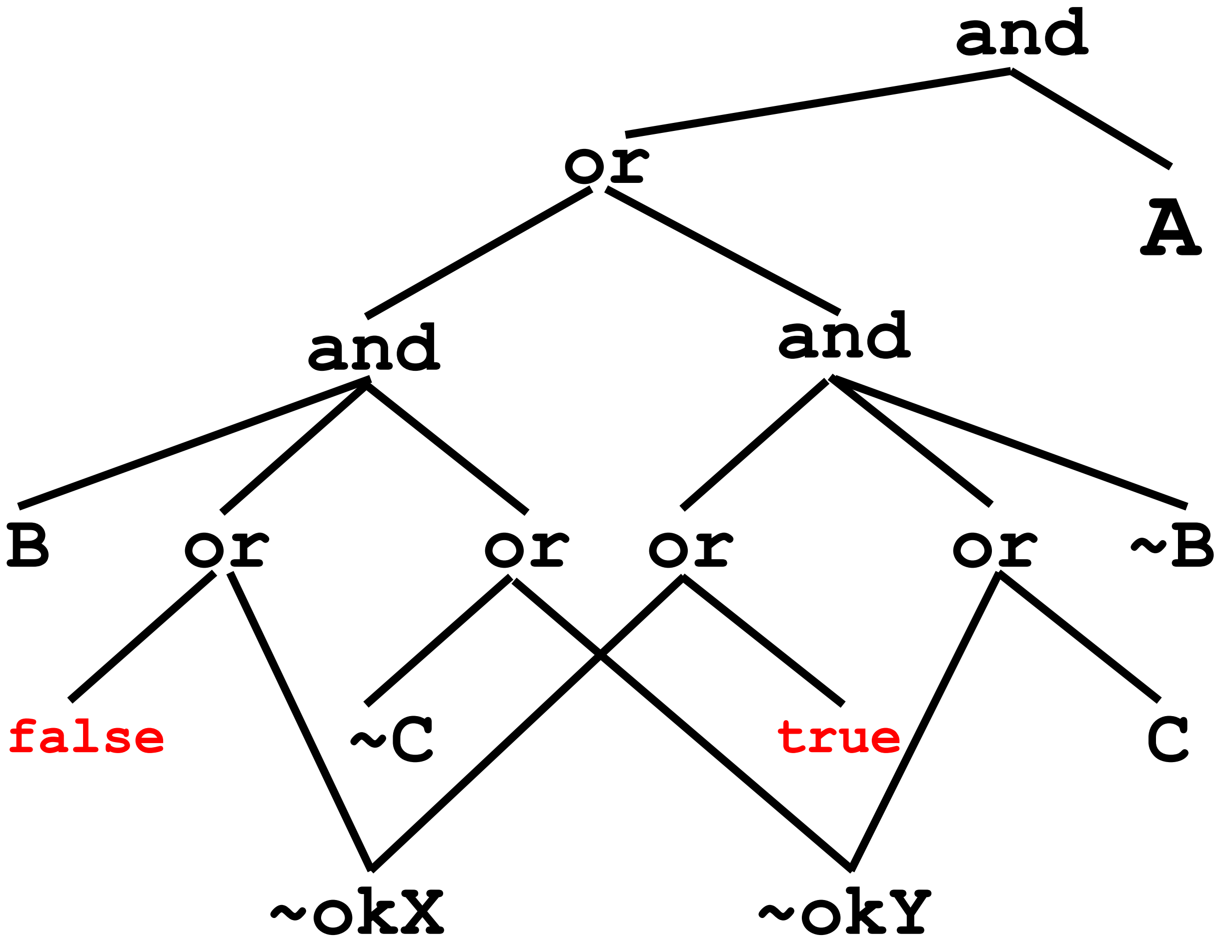


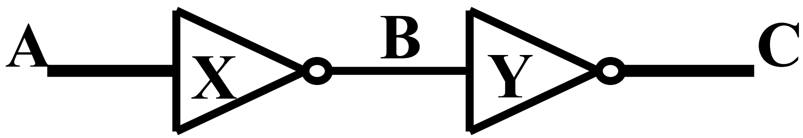
Literal Conjoin

Conditioning

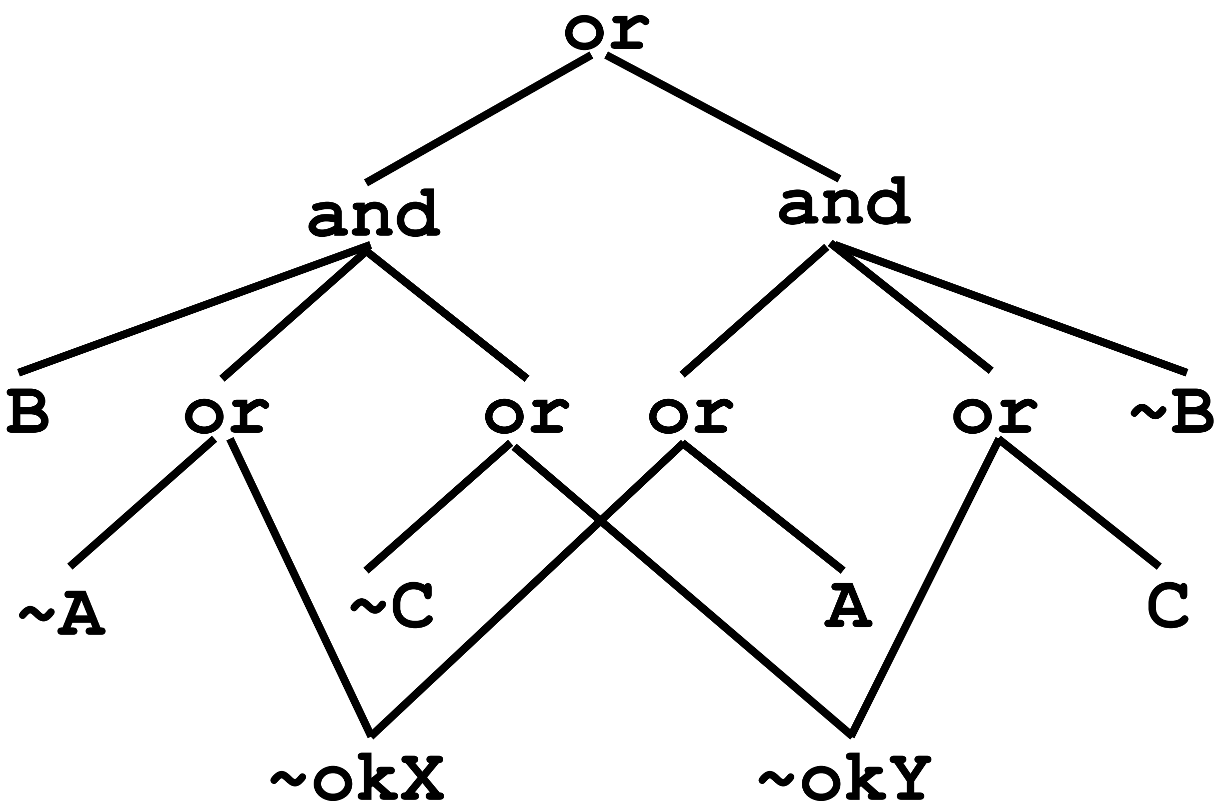


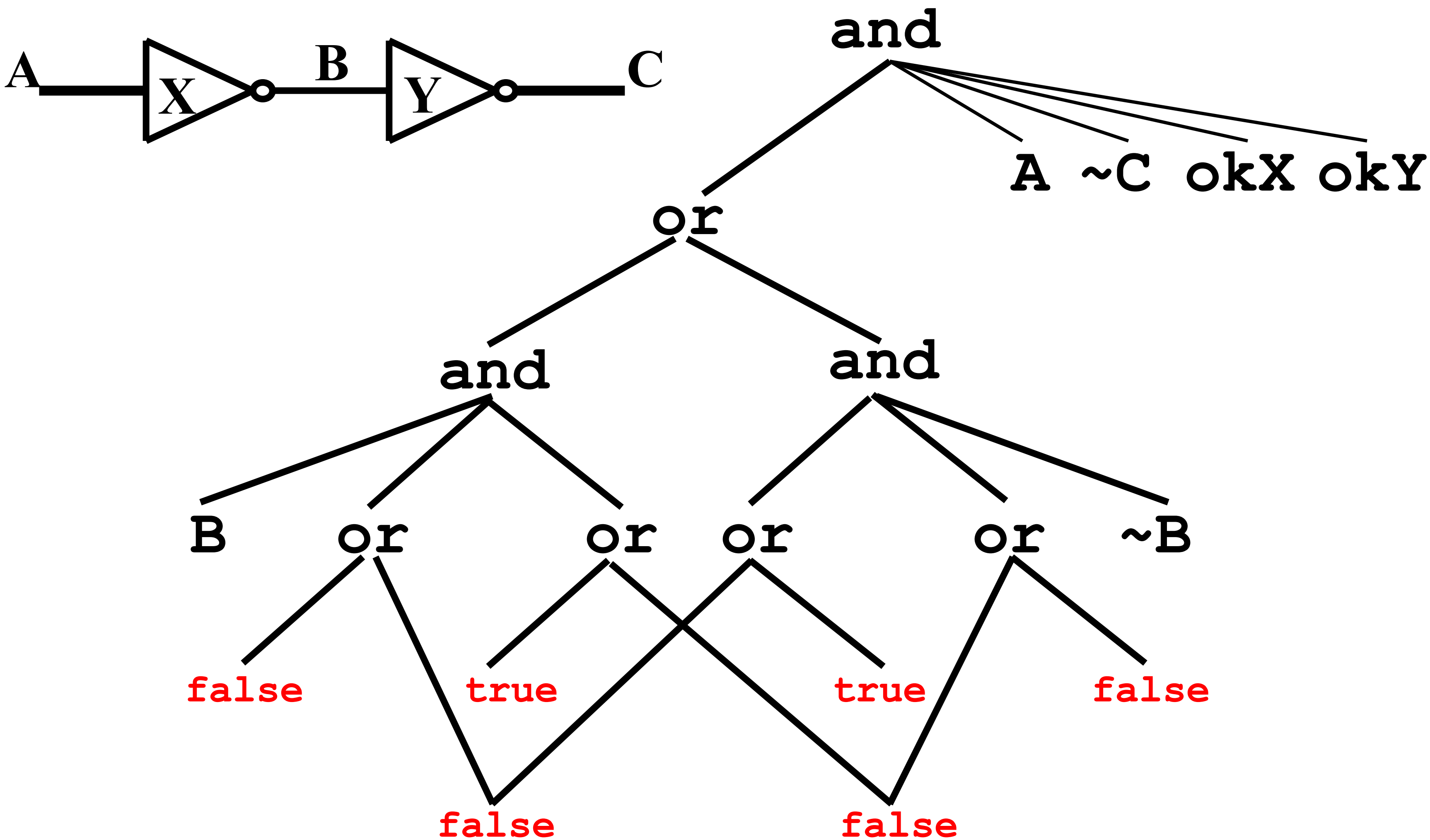
Literal Conjoin

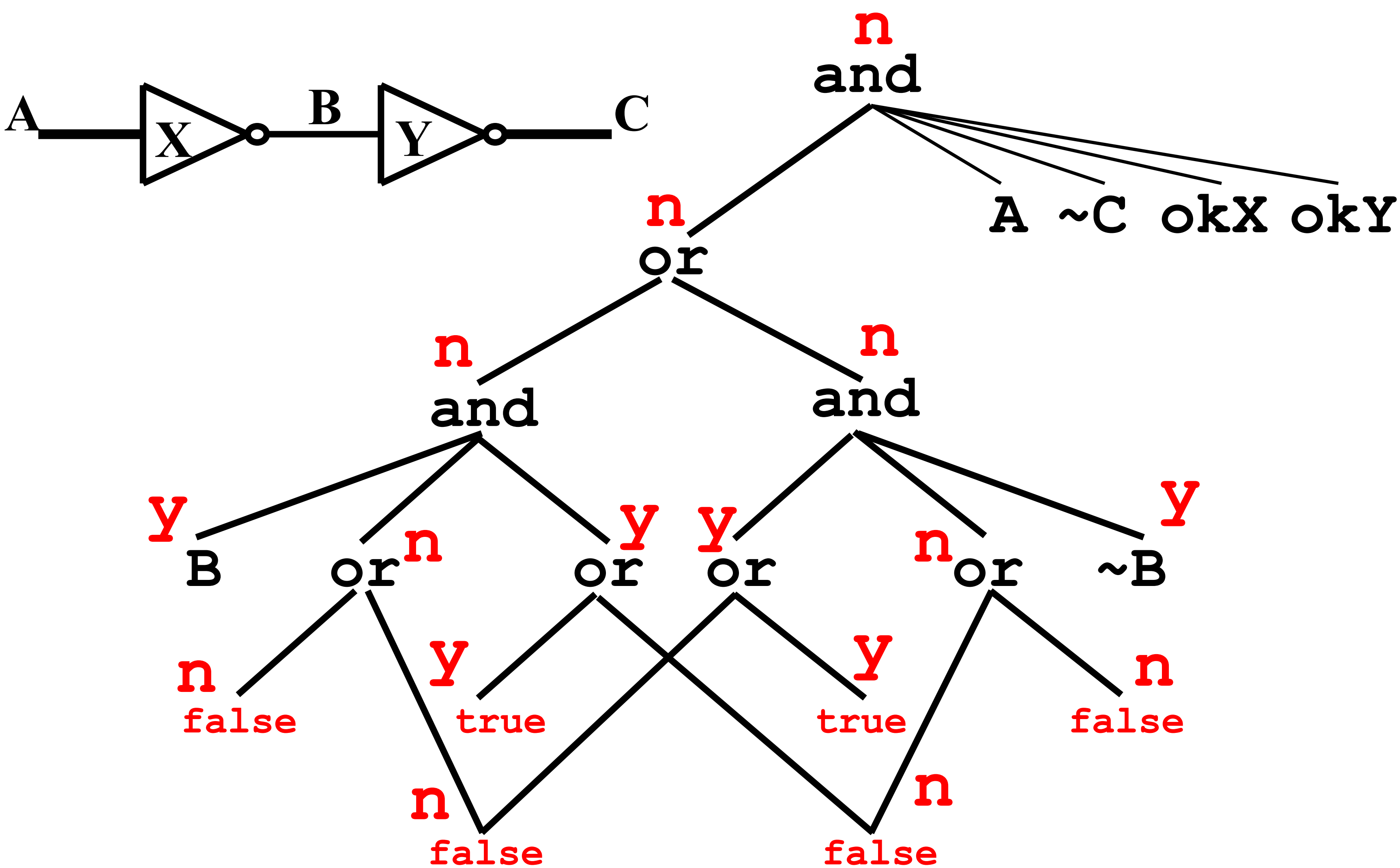




A ~C okX okY




A
X
B
Y
C
and
A
~C
okX
okY
or
and
and
B
or
or
or
or
~B
false
true
true
false
false
false



A
X
B
Y
C
n
and
A ~C okX okY
n
or
n
and
n
and
y
B
or
n
y
or
y
or
n
or
y
~B
n
false
y
true
y
true
n
false
n
false
n
false

Projection: Existential Quantification

Knowledge Base

$$\Delta = A \Rightarrow B, B \Rightarrow C, C \Rightarrow D$$

Existentially quantifying B,C

Forgetting B,C

Projecting on A,D

$$(\exists B \exists C \Delta) = A \Rightarrow D$$

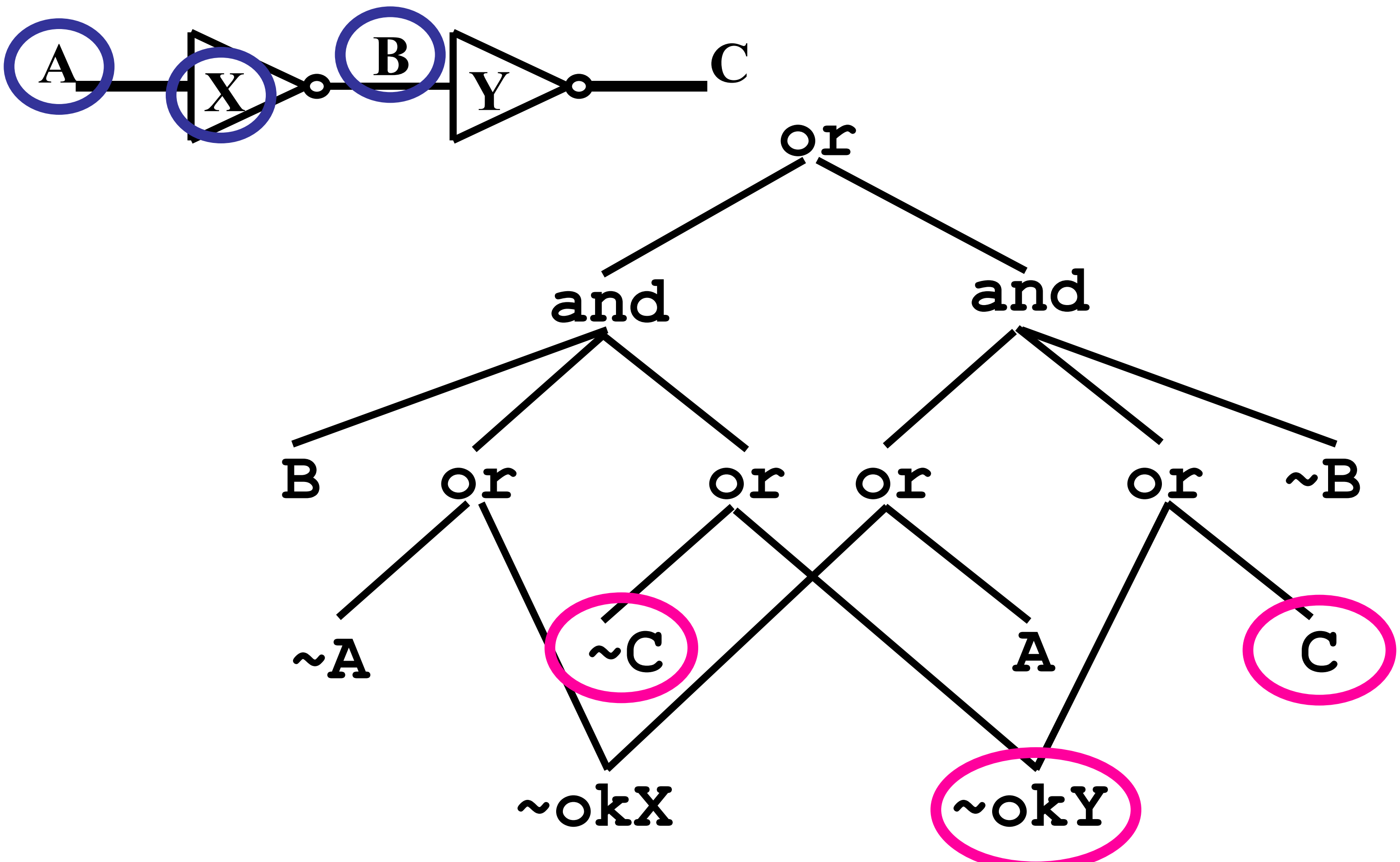
Projection: Existential Quantification

Formal Definition

$$\exists X \Delta = (\Delta \mid X) \vee (\Delta \mid \neg X)$$

- **If Knowledge base is a CNF:**
  - **Close under resolution**
  - **Remove all clauses that mention X**

Projection: Existential Quantification



Projection: Existential Quantification

$$\exists X(\Delta \wedge \Gamma)$$
$$= ((\Delta \wedge \Gamma) \mid X) \vee ((\Delta \wedge \Gamma) \mid \neg X)$$
$$= ((\Delta \mid X) \wedge (\Gamma \mid X)) \vee ((\Delta \mid \neg X) \wedge (\Gamma \mid \neg X))$$
$$= (\Delta \wedge (\Gamma \mid X)) \vee (\Delta \wedge (\Gamma \mid \neg X))$$
$$= \Delta \wedge ((\Gamma \mid X) \vee (\Gamma \mid \neg X))$$
$$= \Delta \wedge (\exists X \Gamma)$$

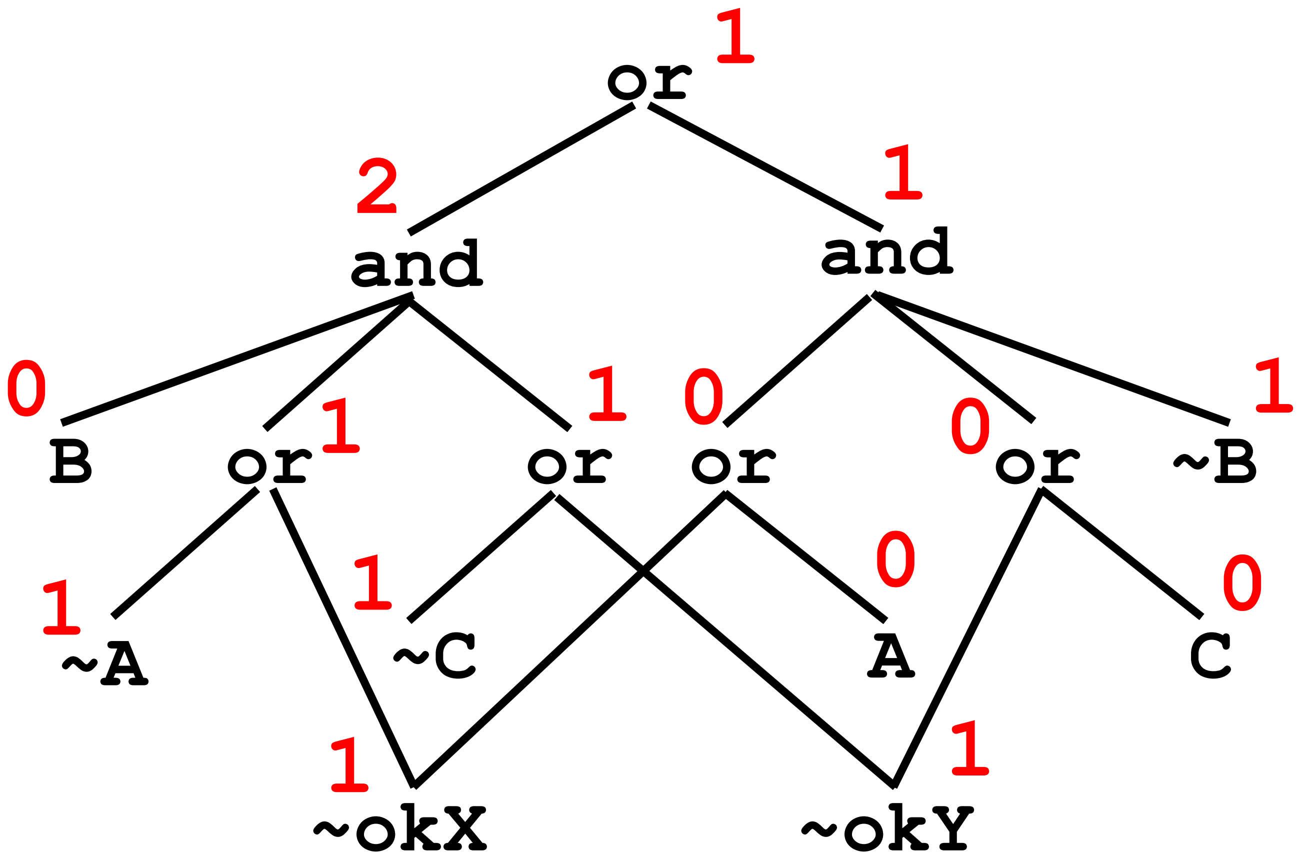
A ~B B ~okX true

Minimum Cardinality

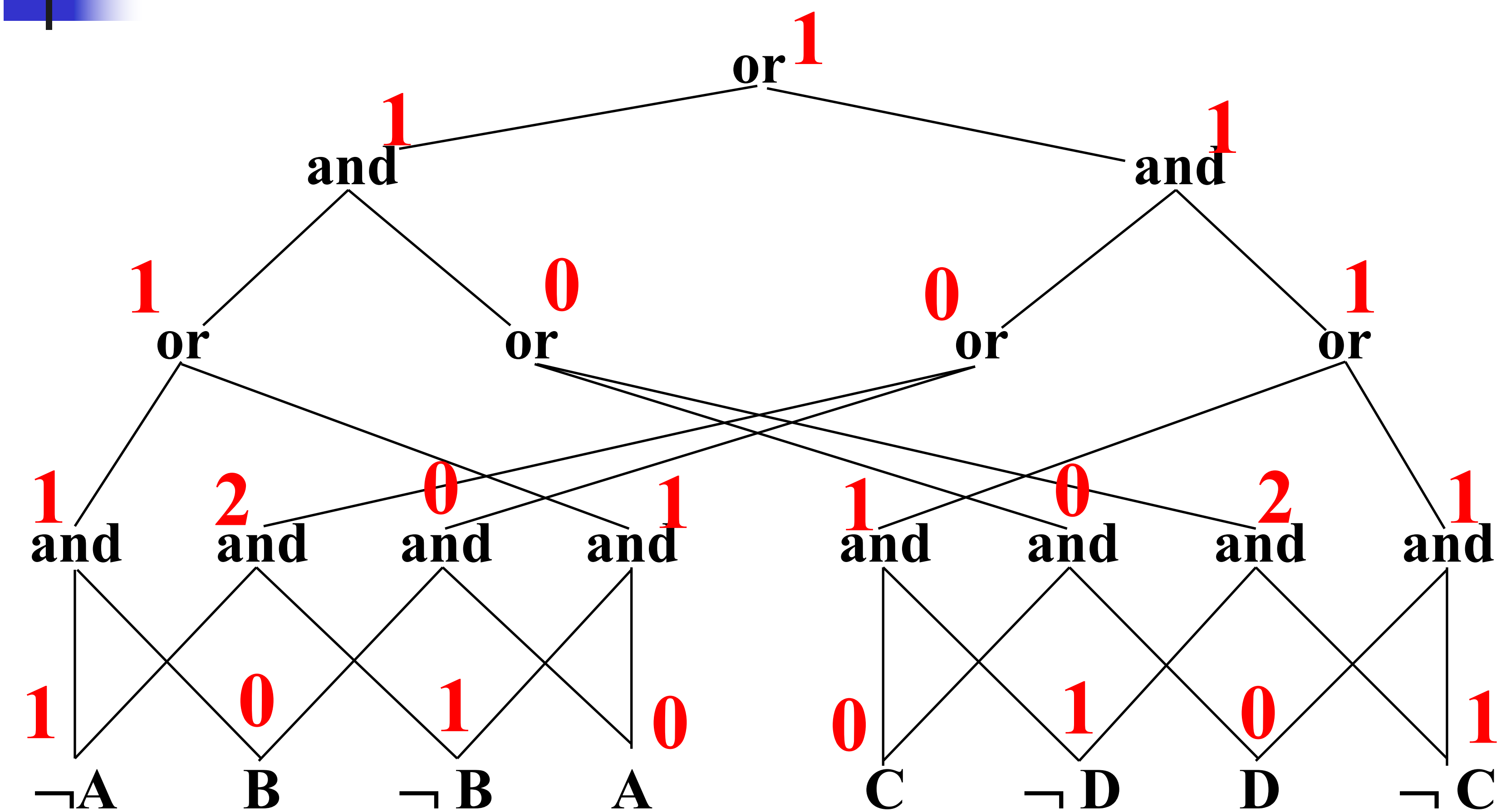
```
 A & okX => ~B
~A & okX =>  B
 B & okY => ~C
~B & okY =>  C
```

okX	okY	A	B	C	
true	true	true	false	true	1
true	false	true	false	false	3
.	.	.	.	.	.
.	.	.	.	.	.
					1

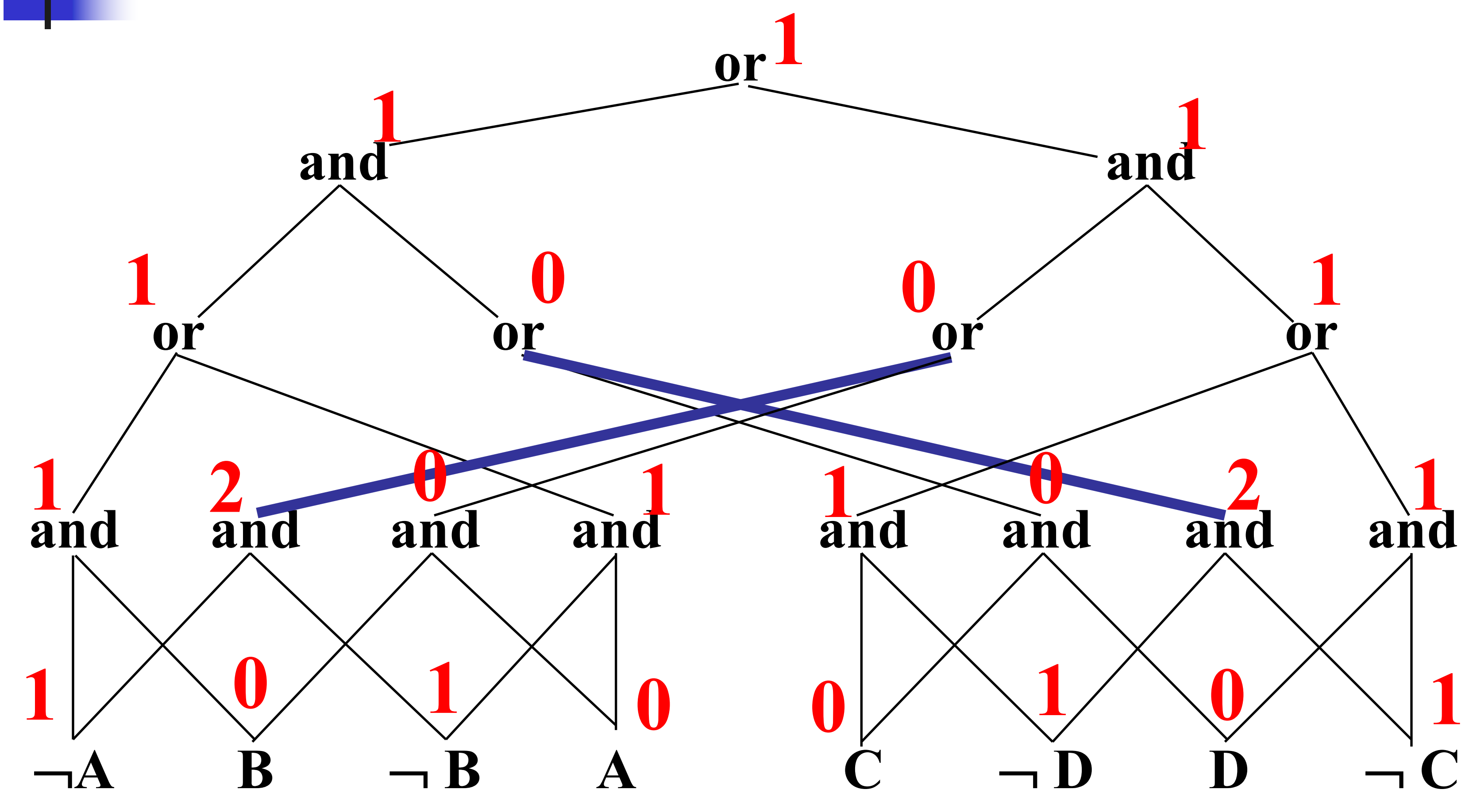
Minimum Cardinality



Minimizing:
Requires Smoothness

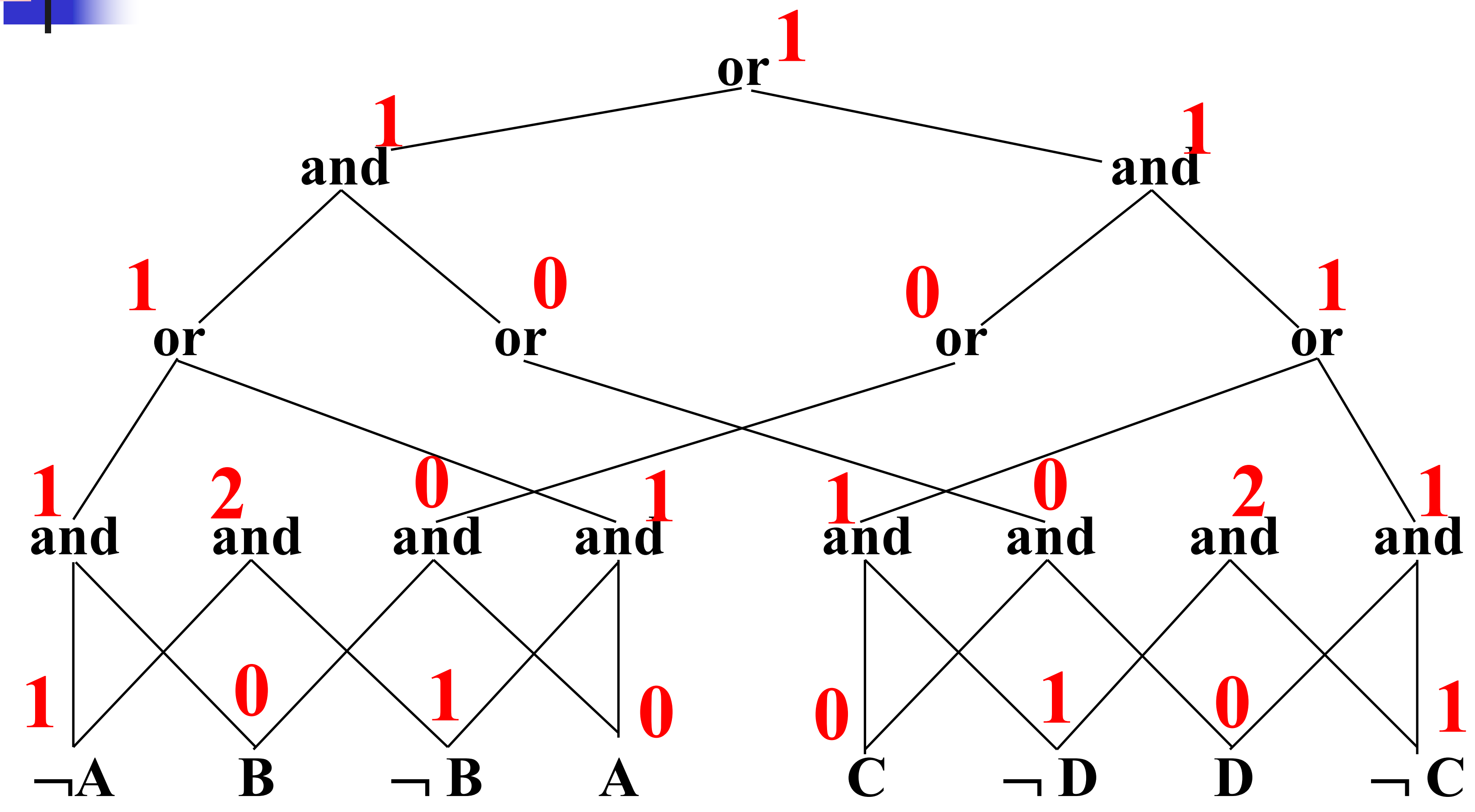


Minimizing

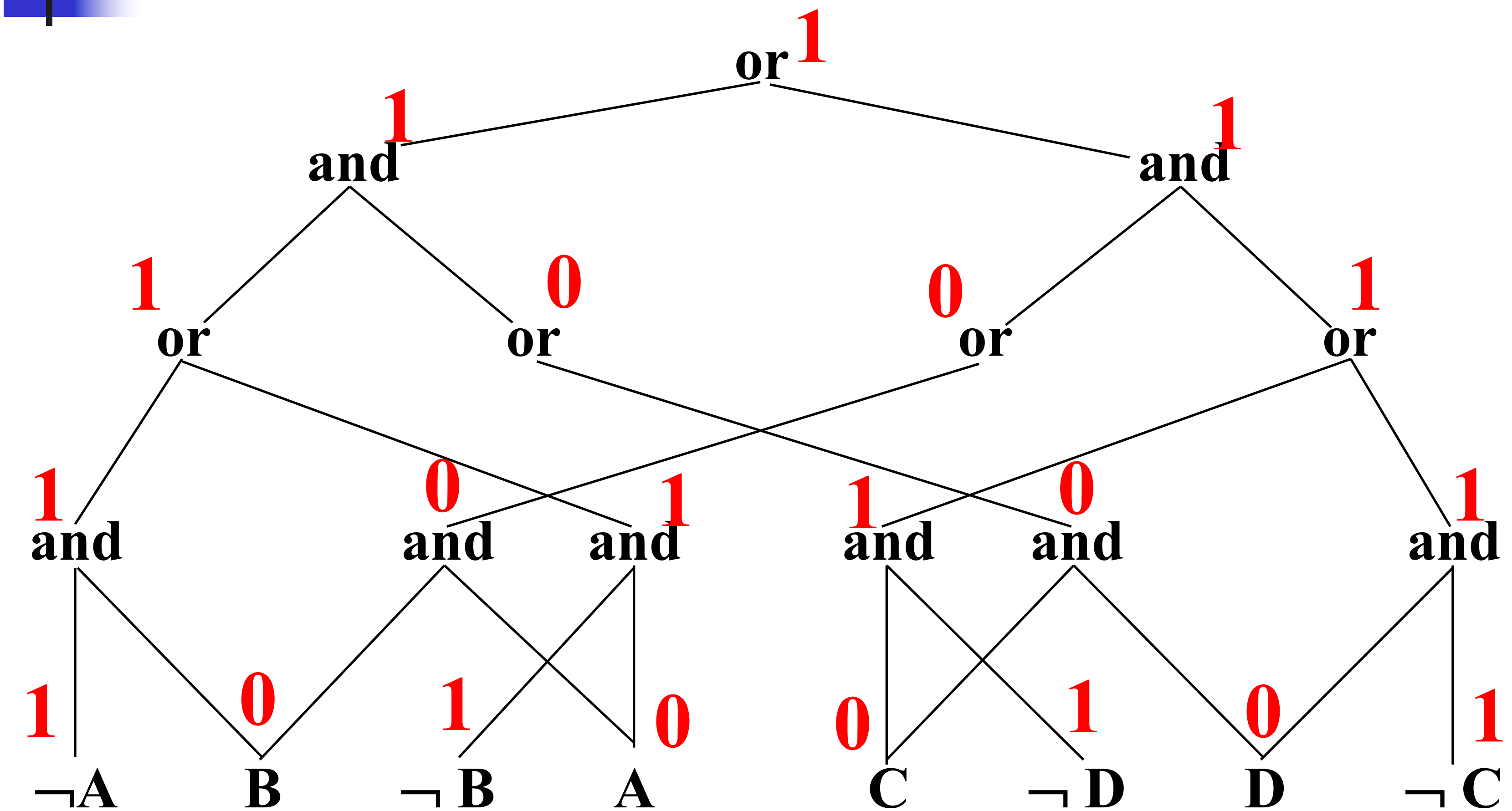


or 1
1 and
and 1
1 or
0 or
0 or
1 or
1 and
2 and
0 and
1 and
1 and
0 and
2 and
1 and
1
0
1
0
0
1
0
1
¬A
B
¬ B
A
C
¬ D
D
¬ C

Minimizing

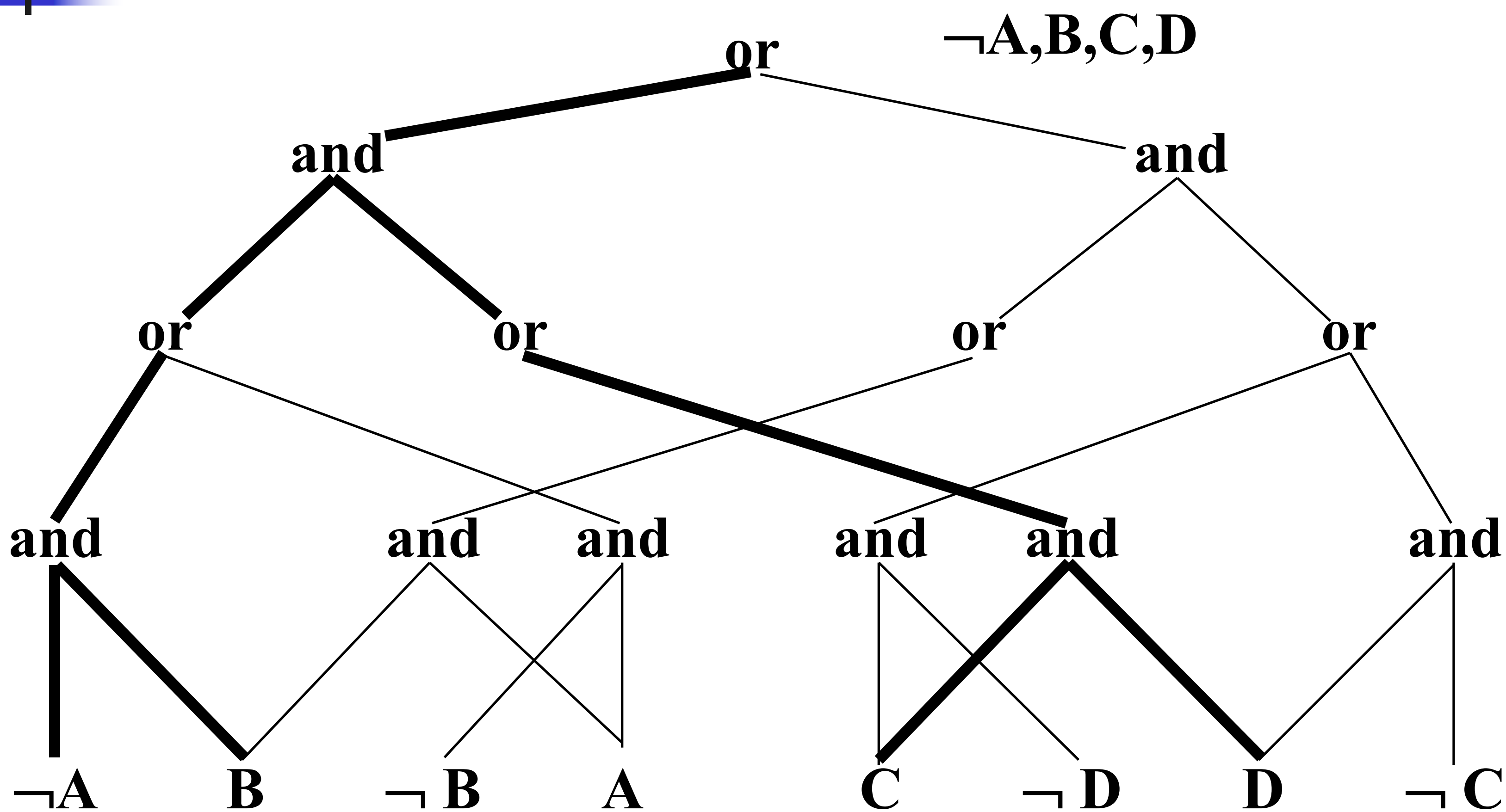

or 1
and 1
and 1
or 1
or 0
or 0
or 1
and 1
and 2
and 0
and 1
and 1
and 0
and 2
and 1
1 ¬A
0 B
1 ¬ B
0 A
0 C
1 ¬ D
0 D
1 ¬ C

Minimizing

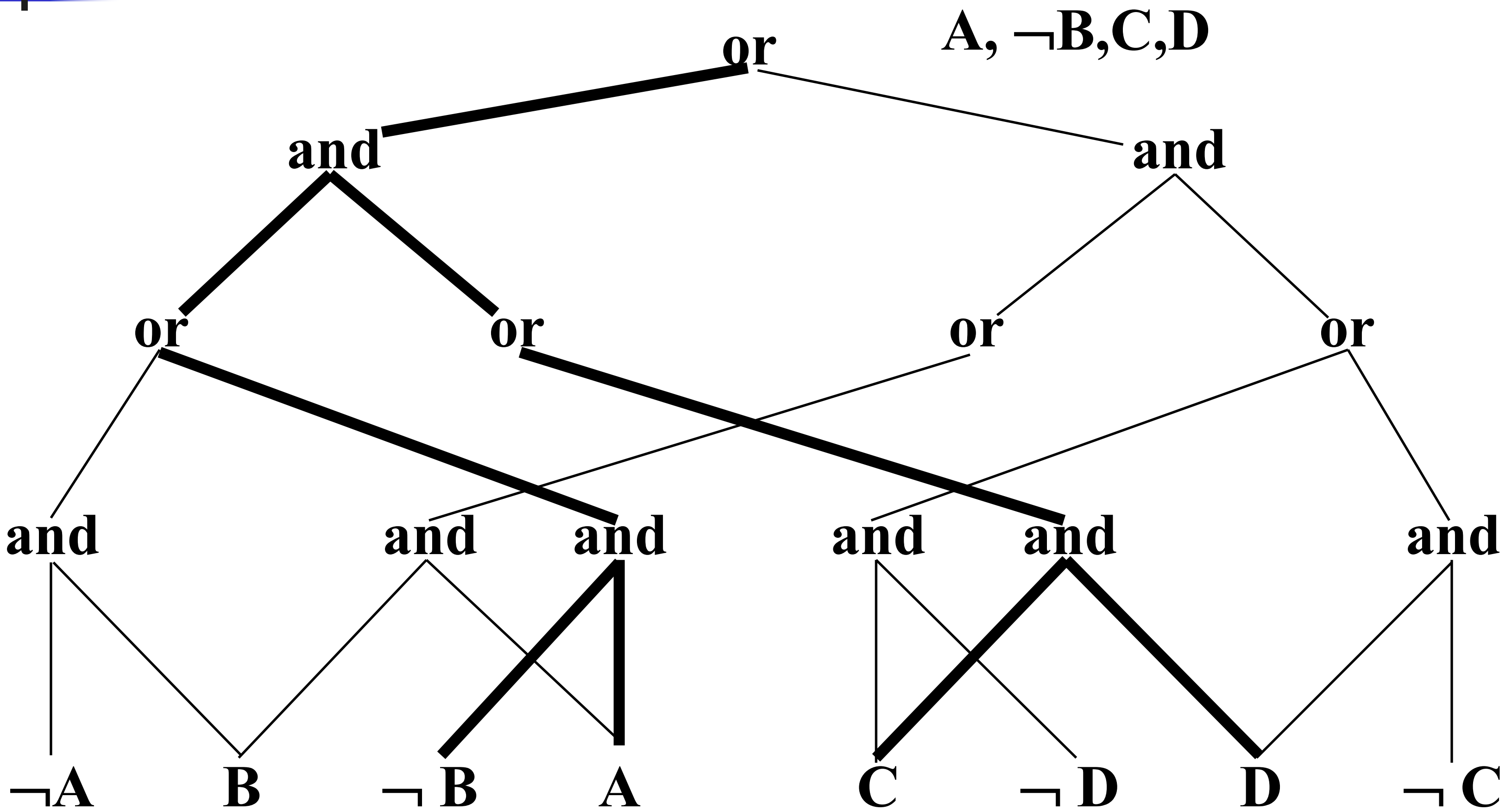


or 1
1 and
and 1
1 or
or 0
0 or
or 1
1 and
0 and
and 1
1 and
0 and
and 1
1
0
1
0
0
1
0
1
¬A
B
¬ B
A
C
¬ D
D
¬ C

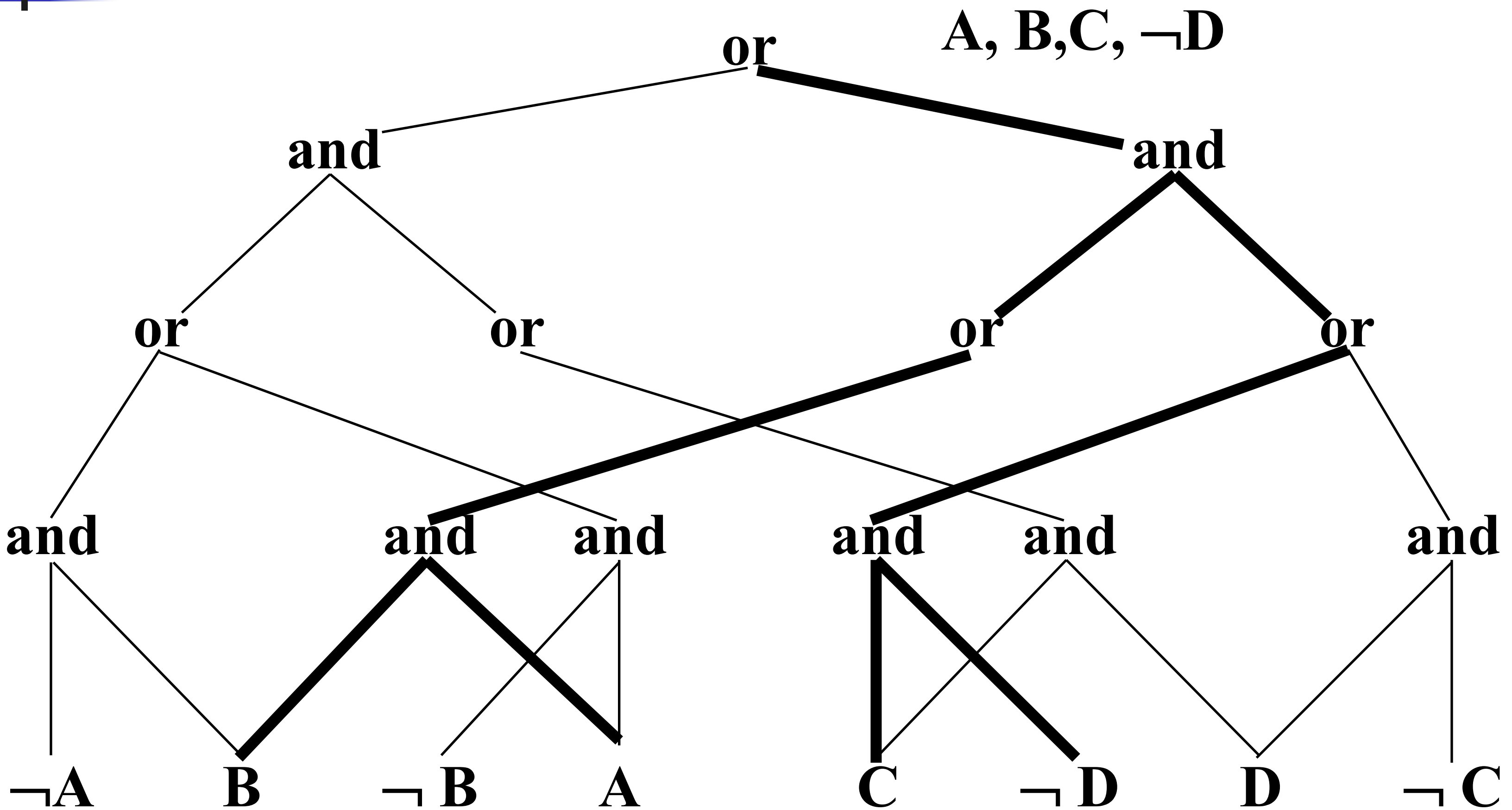
Minimizing



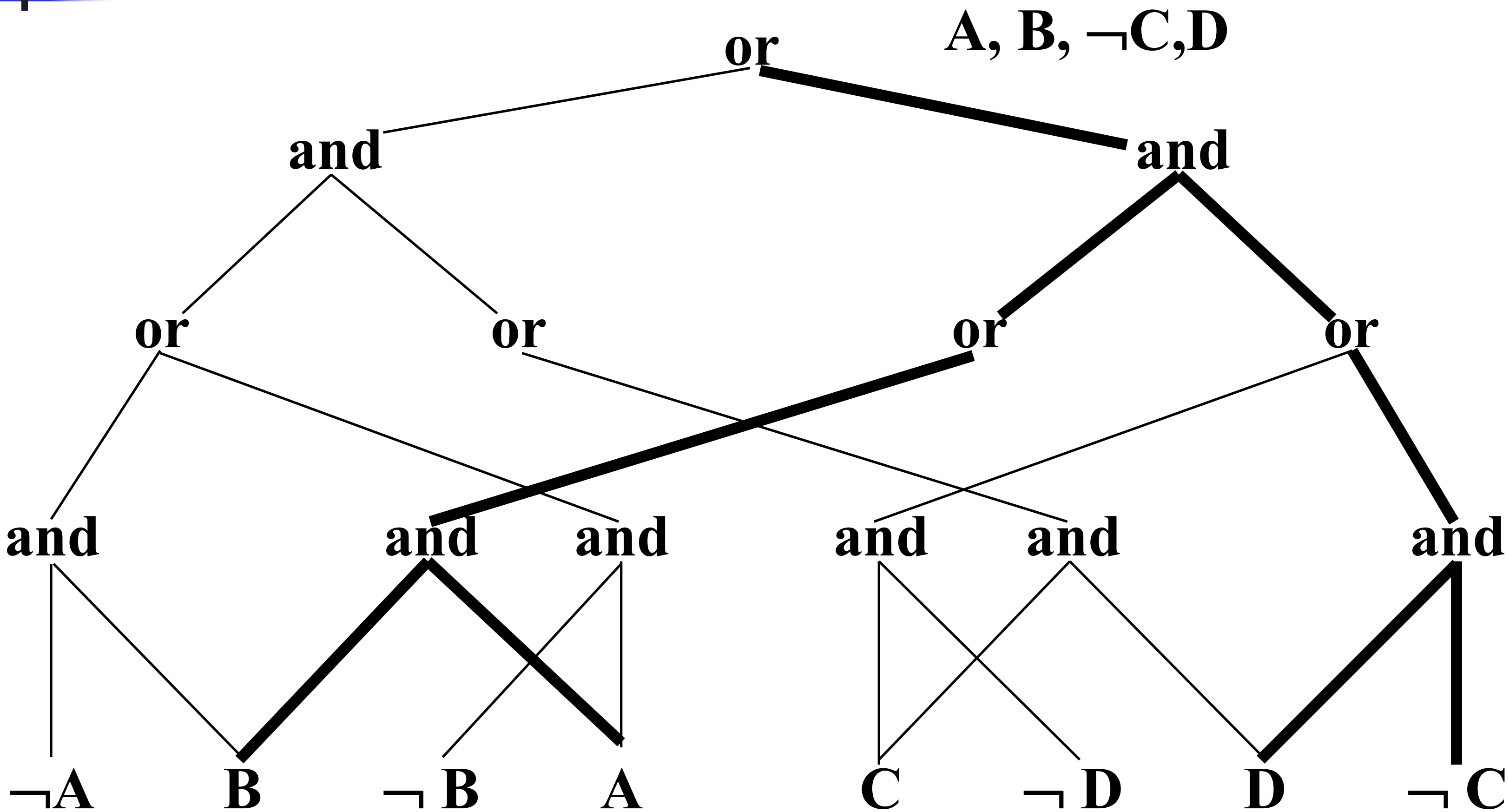
Minimizing



Minimizing

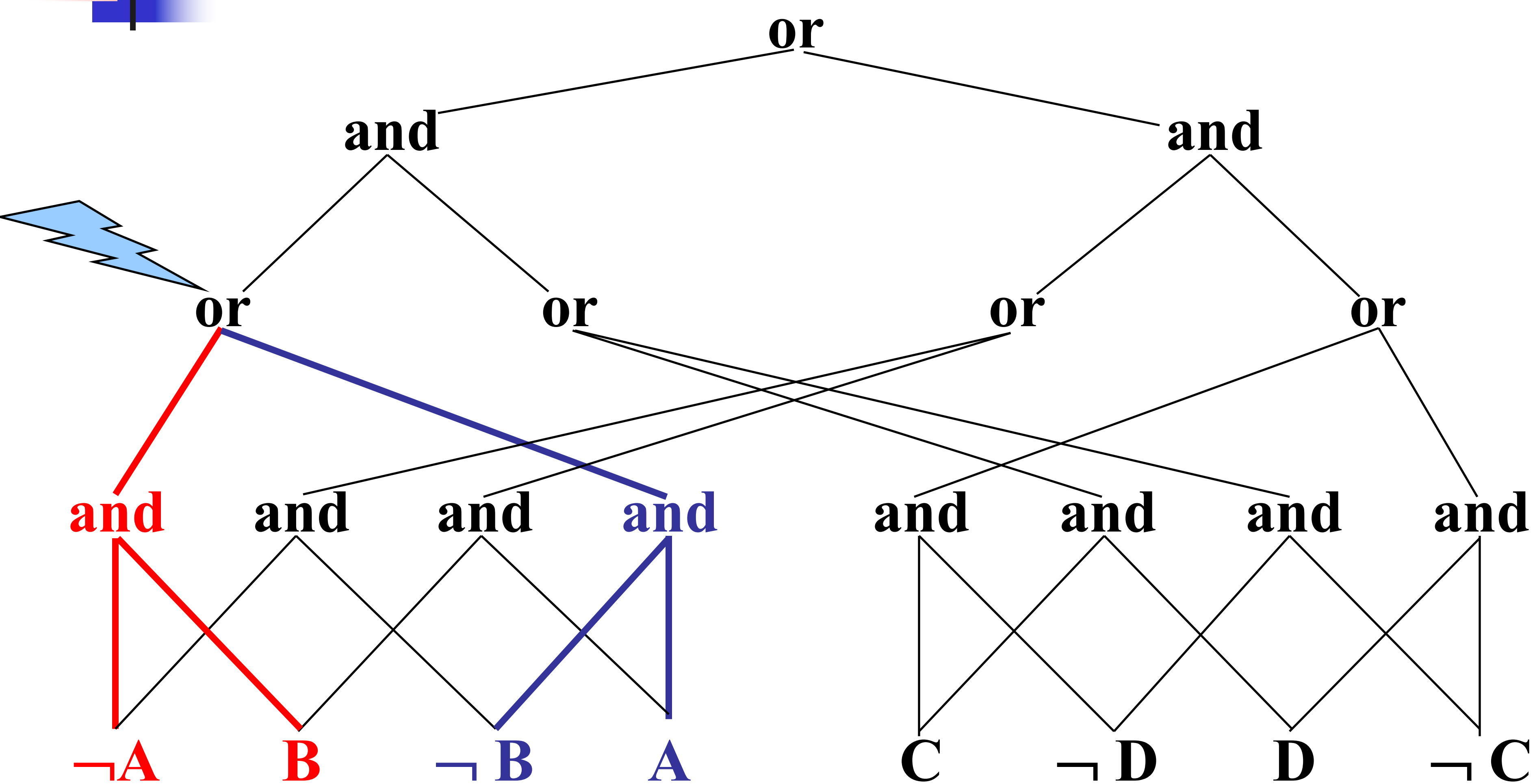


Minimizing

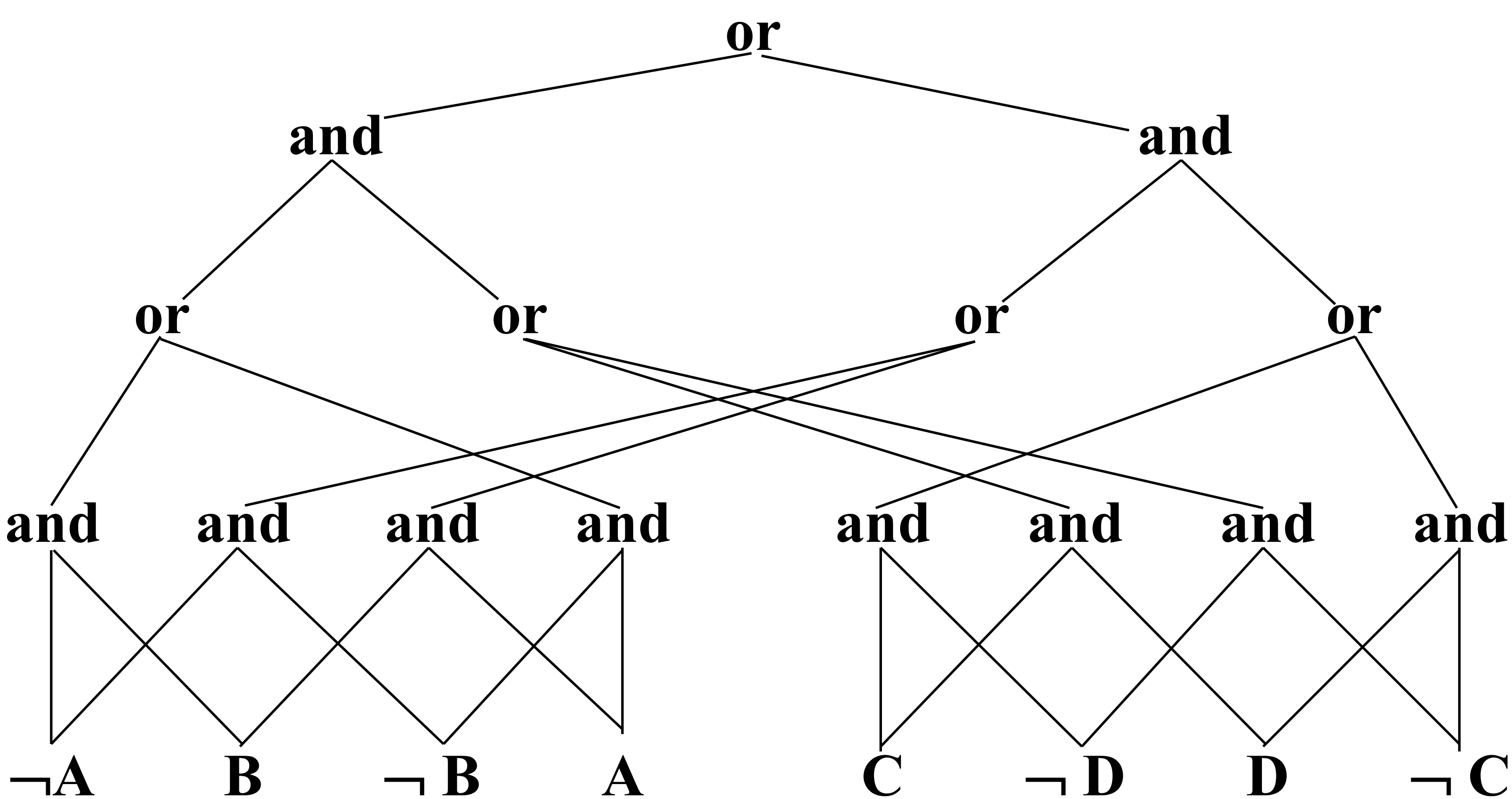


Determinism

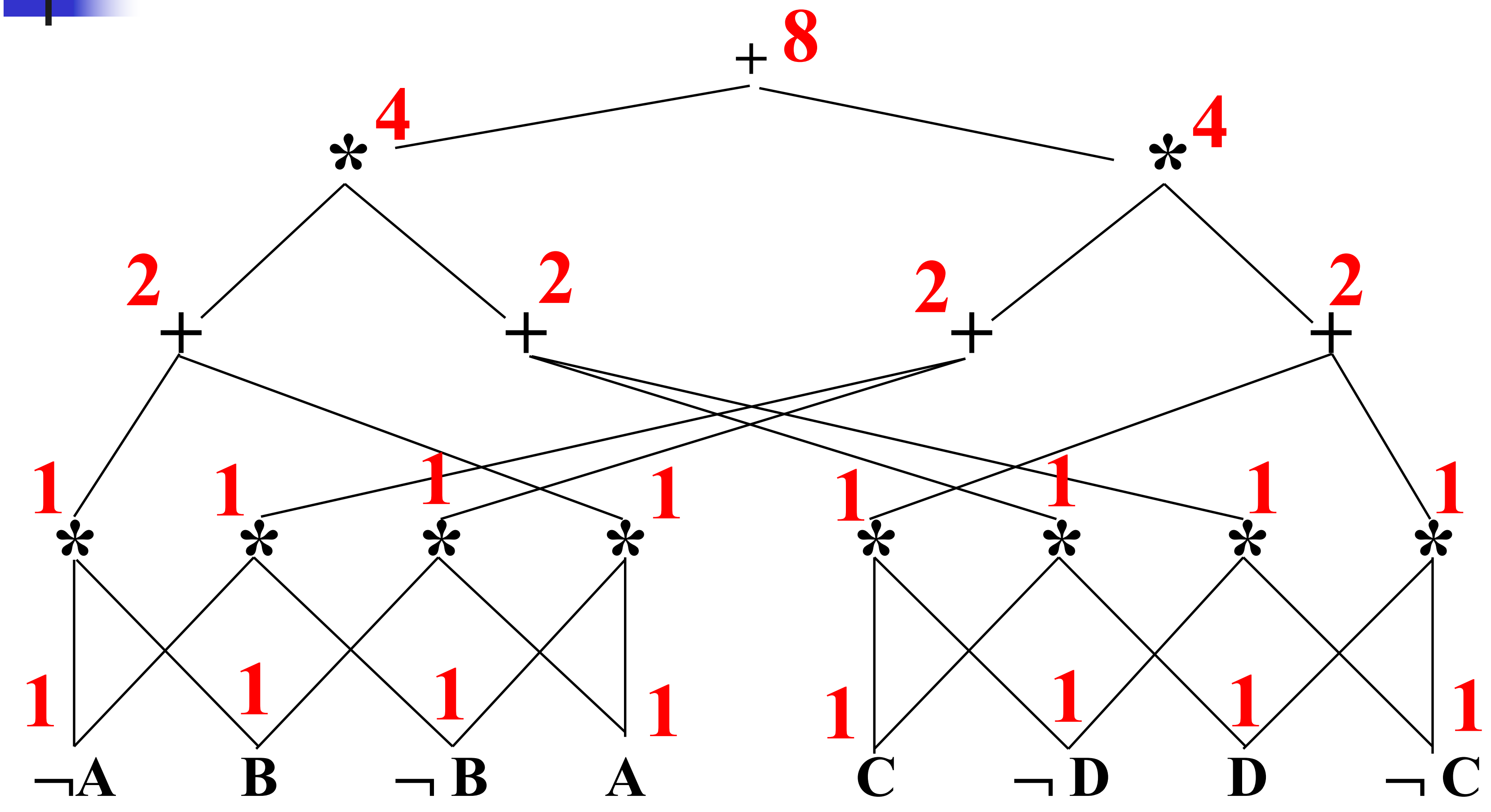
Determinism



Counting Models

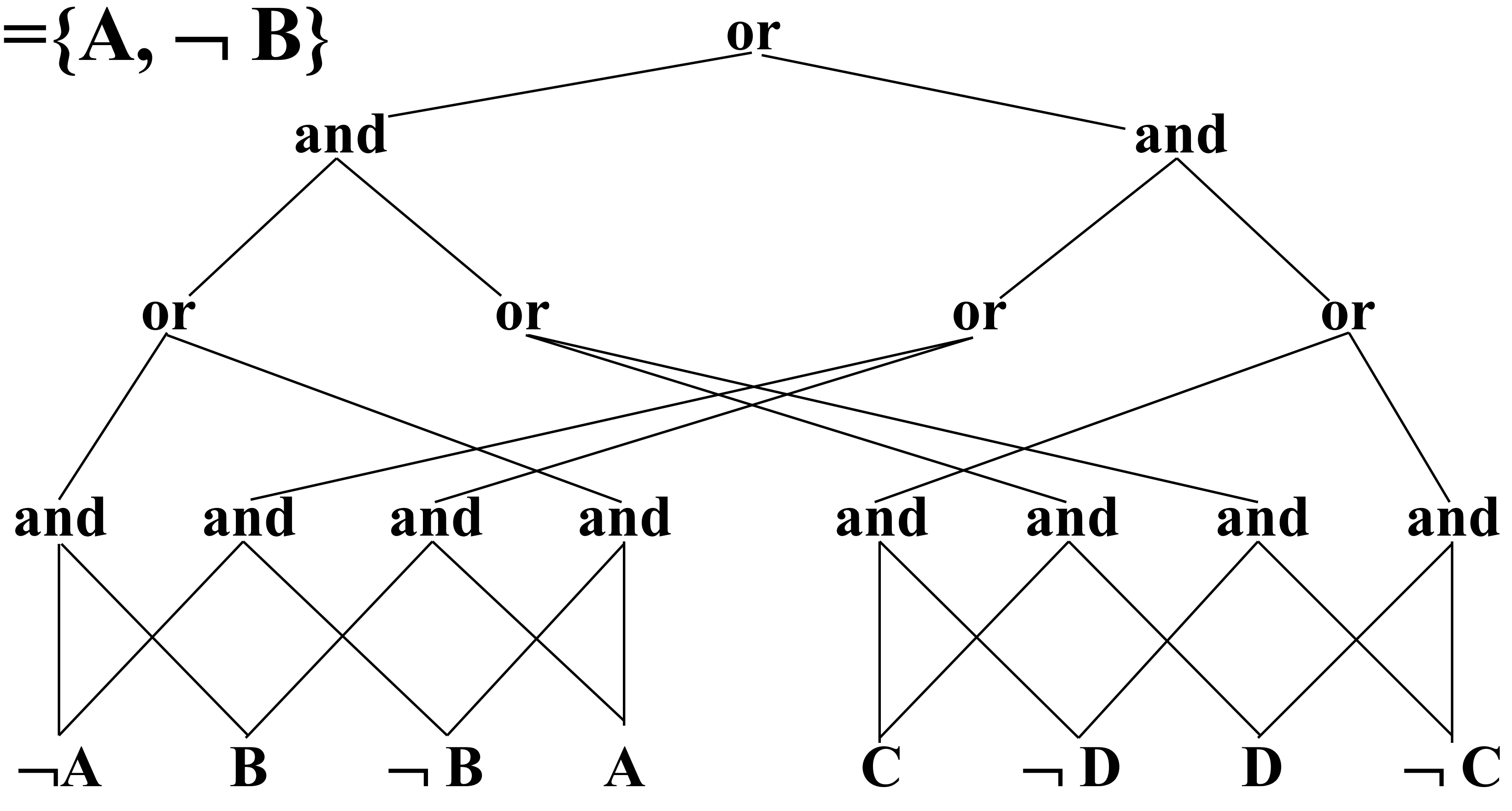


Counting Graph



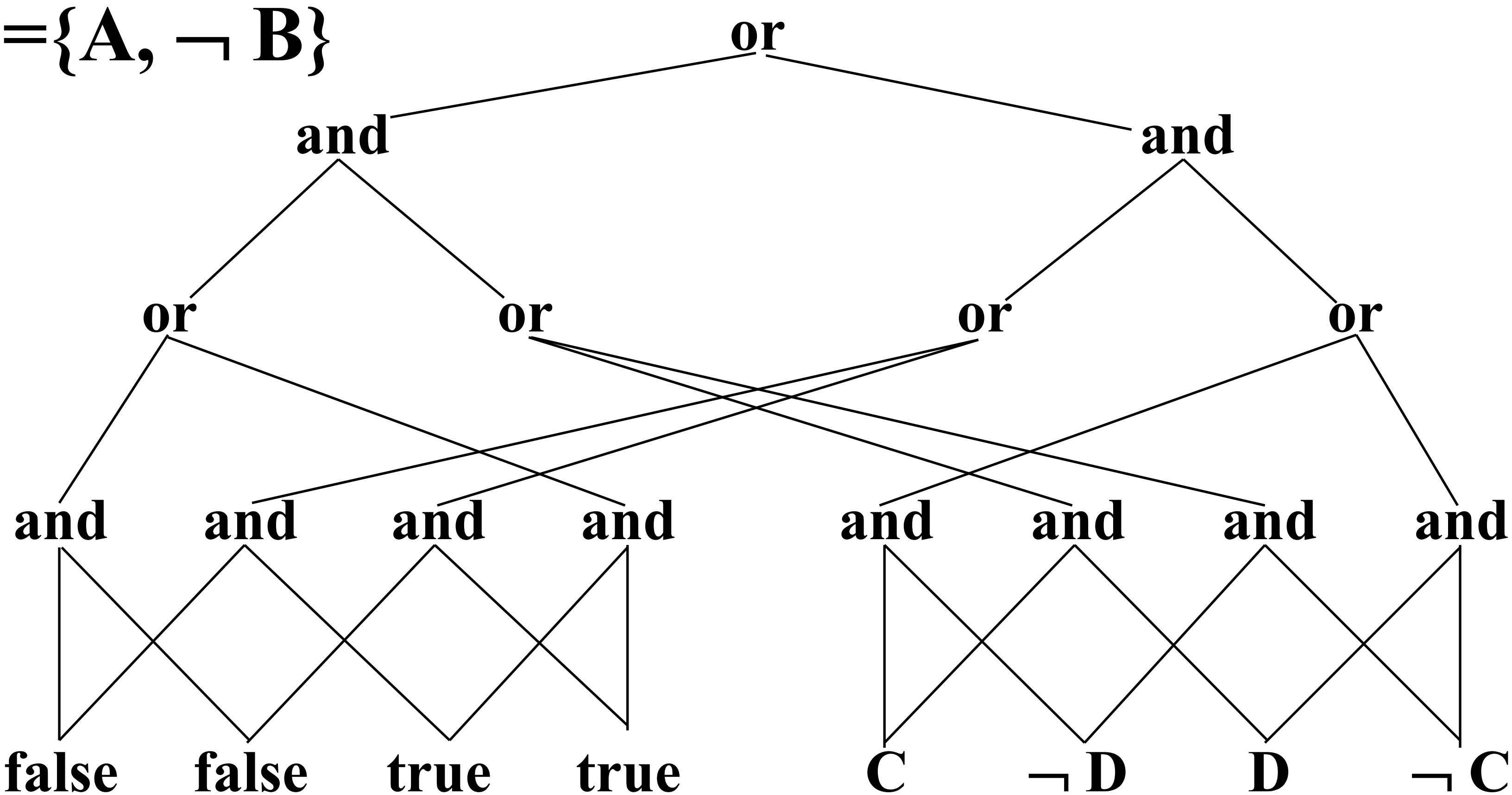
Counting Models

S={A, ¬ B}



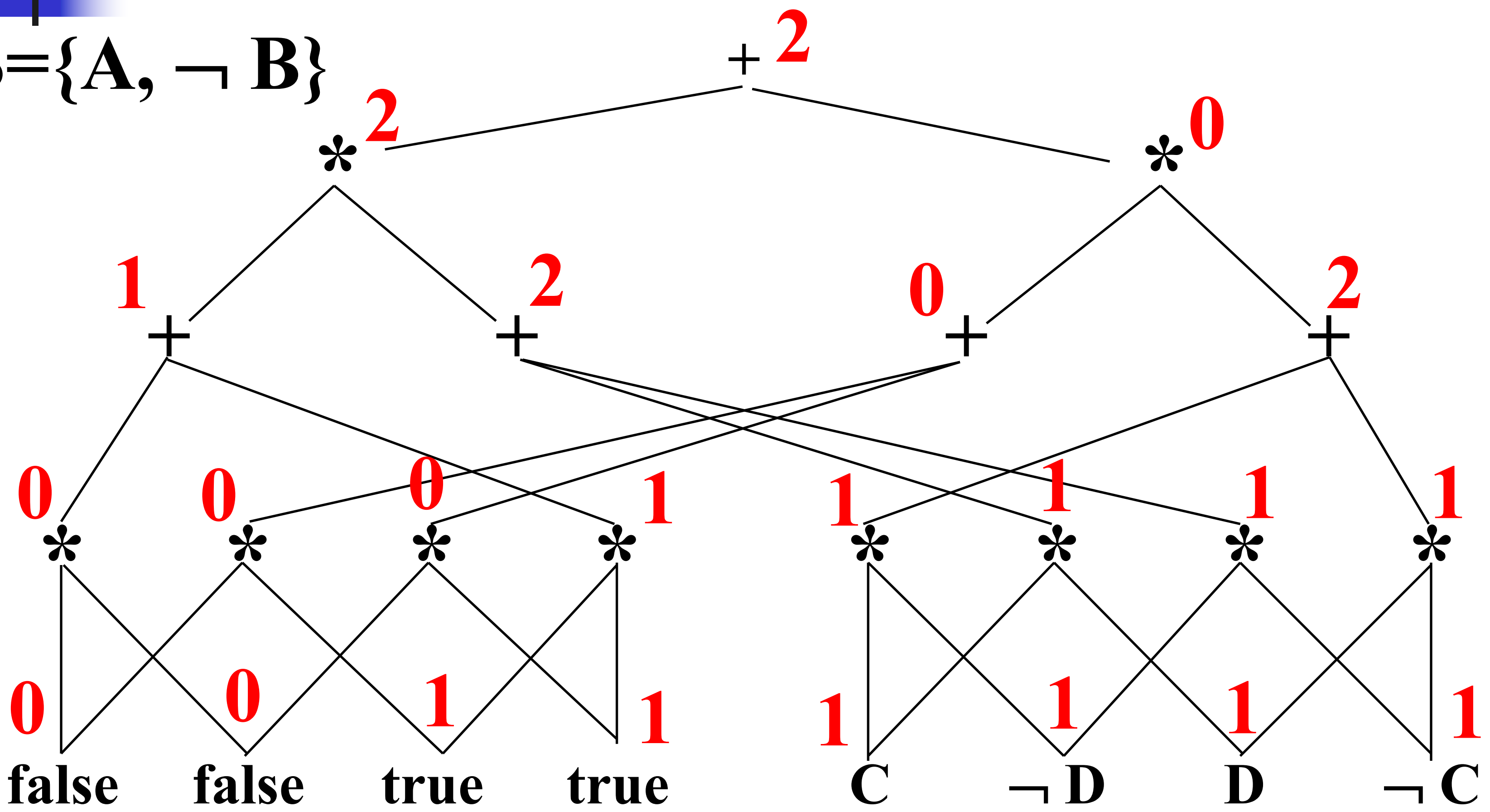
Counting Models

$S=\{A, \neg B\}$



Counting Graph

S={A, ¬ B}



Determinism

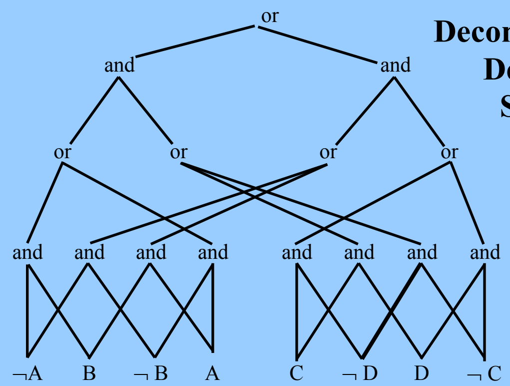
Query	d-DNNF
CO: Consistency	Yes
VA: Validity	Yes
CE: Clausal entailment	Yes
SE: Sentential entailment	
IP: Implicant testing	Yes
EQ: Equivalence testing	?
MC: Model Counting	Yes
ME: Model enumeration	Yes

OBDDS and SDDs

Query	OBDD
CO: Consistency	Yes
VA: Validity	Yes
CE: Clausal entailment	Yes
SE: Sentential entailment	Yes
IP: Implicant testing	Yes
EQ: Equivalence testing	Yes
MC: Model Counting	Yes
ME: Model enumeration	Yes

A Knowledge Compilation MAP

Negation Normal Form



Decomposability
Determinism
Smoothness
Flatness
Decision
Ordering

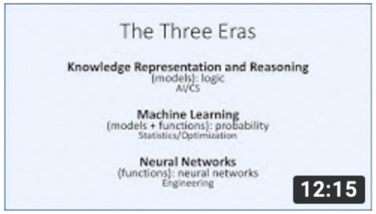
Succinctness

Polytime Operations

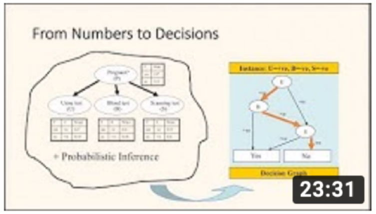
Consistency (CO)
Validity (VA)
Clausal entailment (CE)
Sentential entailment (SE)
Implicant testing (IP)
Equivalence testing (EQ)
Model Counting (CT)
Model enumeration (ME)

Projection (exist. quantification)
Conditioning
Conjoin, Disjoin, Negate

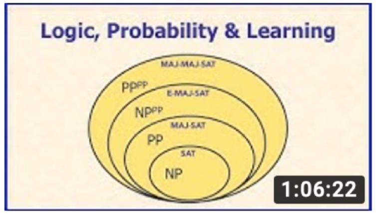
CACM Oct. 2018 - Human-Level Intelligence or Animal...
Association for Computing Ma...
2.7K views • 4 months ago



Adnan Darwiche – On AI Education
UCLA Automated Reasoning G...
1.4K views • 5 months ago



Adnan Darwiche – Explaining and Verifying AI Systems
UCLA Automated Reasoning G...
464 views • 4 months ago



Adnan Darwiche – On the Role of Logic in Probabilisti...
UCLA Automated Reasoning G...
2K views • 1 year ago

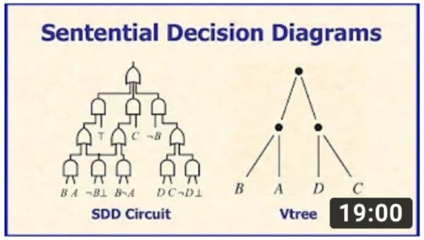
Adnan Darwiche – On Model-Based versus Model-Blind...
UCR School of Public Policy
3.3K views • 1 year ago

Learning and Reasoning with Bayesian Networks ▶ PLAY ALL

Lectures by Adnan Darwiche for his UCLA course on Bayesian Networks.

1a. Course Overview with a Historical Perspective on AI
UCLA Automated Reasoning Group • 2.2K views • 1 year ago
Adnan Darwiche's UCLA course: Learning and Reasoning with Bayesian Networks.

Subscribe!



Adnan Darwiche – SDD: A New Canonical...