UNIT 2
BOOLEAN ALGEBRA

Fall 2021

Boolean Algebra

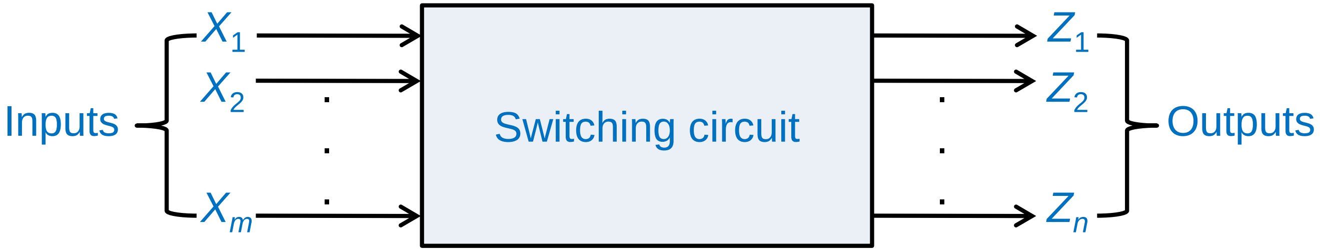
- **Contents**
  - Introduction
  - Basic operations
  - Boolean expressions and truth tables
  - Theorems and laws
    - Basic theorems
    - Commutative, associative, and distributive laws
    - Simplification theorems
    - Multiplying out and factoring
    - DeMorgan’s laws
- **Reading**
  - Unit 2

Introduction

- **Boolean algebra**
  - Is the basic mathematics for logic design of digital systems
  - Differs from ordinary algebra in the values, operations, and laws
- **History**
  - George Boole developed Boolean algebra in 1847 and used it to solve problems in mathematical logic
    - British mathematician and philosopher
  - Claude Shannon first applied Boolean algebra to the design of switching circuits in 1939
    - American electrical engineer and mathematician
    - Master's thesis (21 years-old)
- **In this unit, you will learn how to…**
  - Use a truth table
  - Manipulate basic operations and apply laws of Boolean algebra
  - Relate Boolean expressions to basic logic gates

Switching Circuit & Boolean Variables

- **A switching circuit has one or more inputs and one or more outputs that take on discrete values (two-value in general)**



- **We usually use a Boolean variable, such as *X*, *Y*, *Z*, to represent an input or output of a switching circuit**
  - Usually take on only two different values
  - 1/0 for High/Low or True/False or Yes/No
    - Just symbols, NO numeric values
  - A two-value Boolean variable is also called a switching variable

Boolean algebra differs from ordinary algebra in values, operations, laws

Boolean algebra differs from ordinary algebra in values, operations, laws

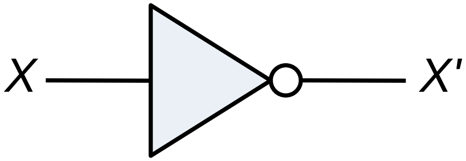
5

Basic Operations

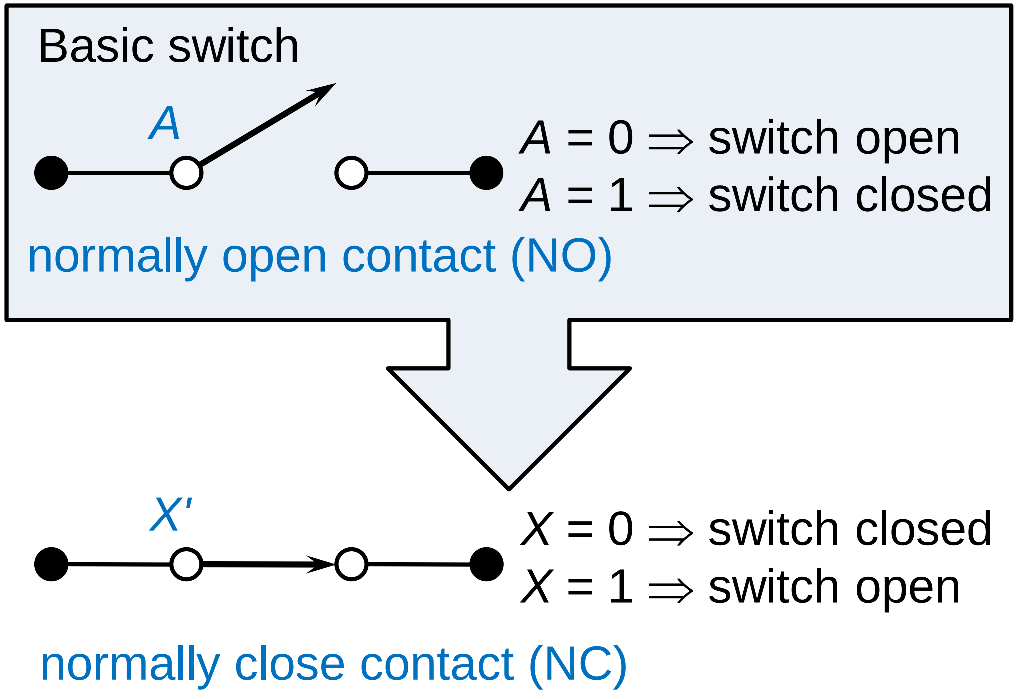
NOT, AND, OR

Operation -- Logic NOT

- **Complement = Inverse = Negate = NOT (' ; ¯; ~; ¬)**
  - 0'=1, 1'=0
  - Symbol



NOT gate
Inverter



$A = 0 \Rightarrow$ switch open
$A = 1 \Rightarrow$ switch closed

normally open contact (NO)

$X = 0 \Rightarrow$ switch closed
$X = 1 \Rightarrow$ switch open

normally close contact (NC)

  - Truth table

Inputs

Outputs

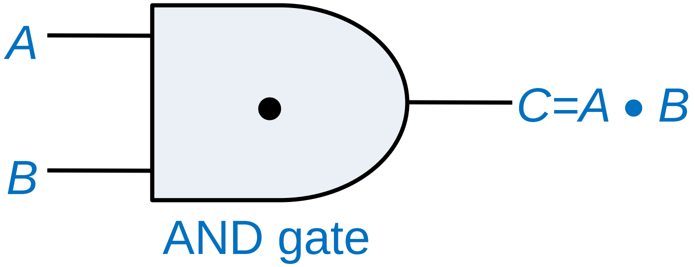
X	X'
0	1
1	0

Input combinations

Output values

Operation -- Logic AND

- **AND (•; ∧)**
  - 0 • 0 = 0, 0 • 1 = 0, 1 • 0 = 0, 1 • 1 = 1
  - Symbol



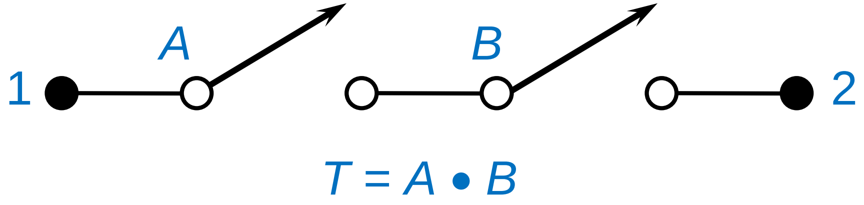
  - Truth table

A	B	C = A B
0	0	0
0	1	0
1	0	0
1	1	1

Omit "•"

If one input is 0
⇒ output is 0

  - Switch (in series)

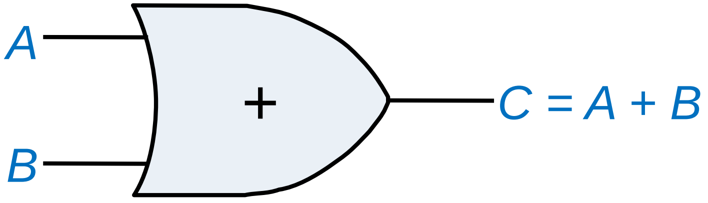


T = 0 ⇒ 1→2 open
T = 1 ⇒ 1→2 closed

Switch *A* closed and switch *B* closed

Operation -- Logic OR

- **OR (+; ∨)**
  - 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1
  - Symbol



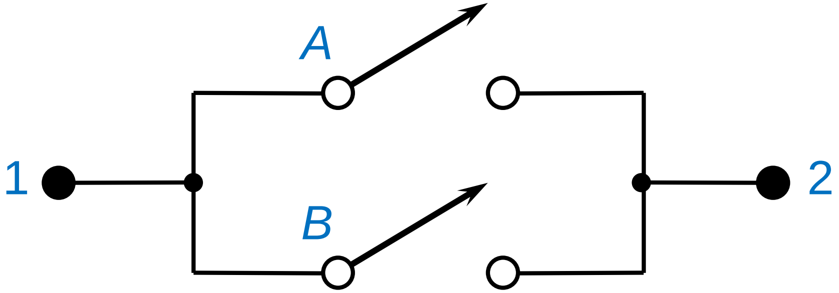
OR gate

  - Truth table

A	B	$C = A+B$
0	0	0
0	1	1
1	0	1
1	1	1

If one input is 1
⇒ output is 1

  - Switch (in parallel)

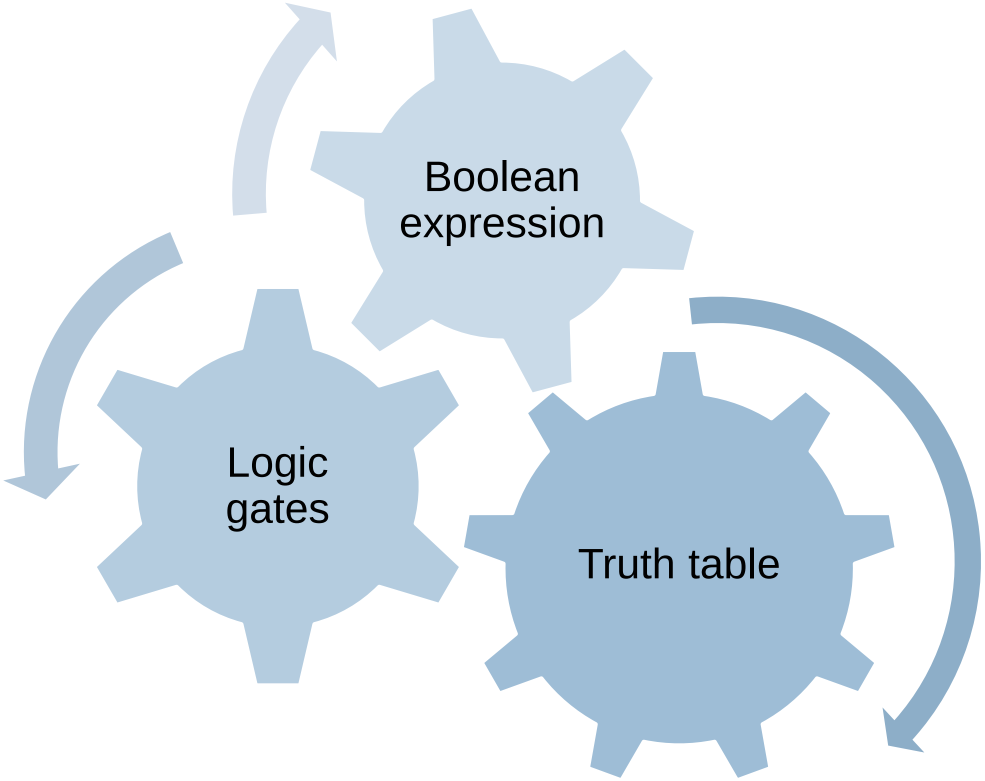


$T = A + B$

$T = 0 \Rightarrow 1 \rightarrow 2$ open
$T = 1 \Rightarrow 1 \rightarrow 2$ closed

Switch A closed or switch B closed

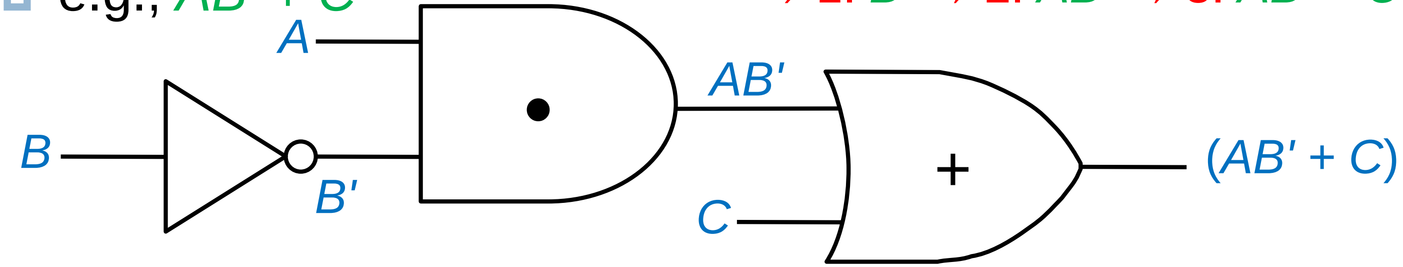
Boolean Expressions and Truth Tables



Boolean Expressions vs. Logic Gates

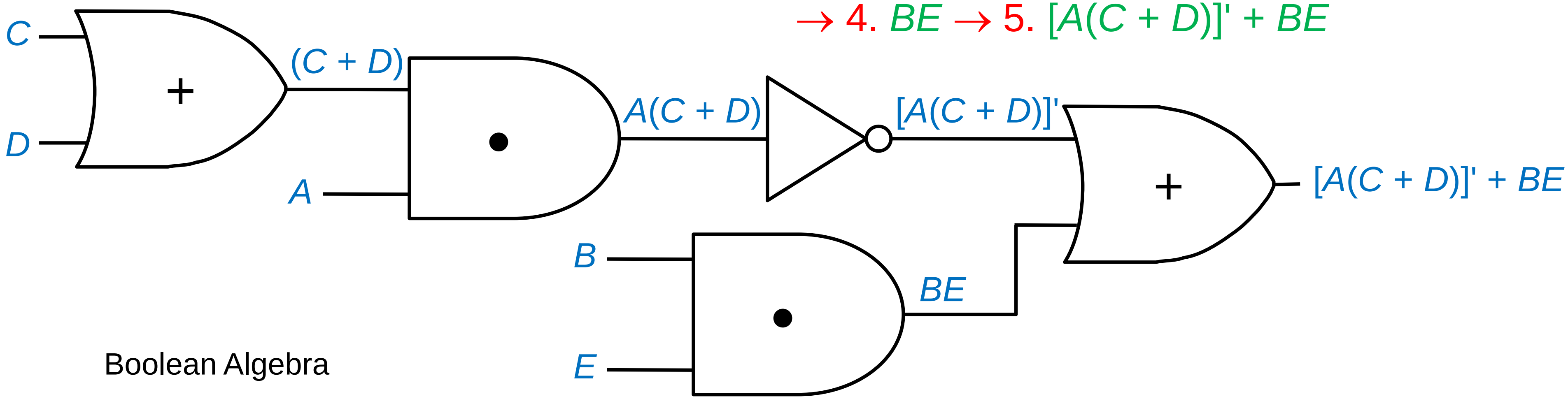
- **A Boolean expression is formed by basic operations on variables or constants, e.g., the simplest one: 0, 1, *X*, *Y*'**
- **Realize a Boolean expression by a circuit of logic gates**
  - Perform operations in order: Parentheses → NOT → AND → OR
  - e.g., $AB' + C$

$\Rightarrow$ 1. B' → 2. AB' → 3. $AB' + C$



  - e.g., $[A(C + D)]' + BE$

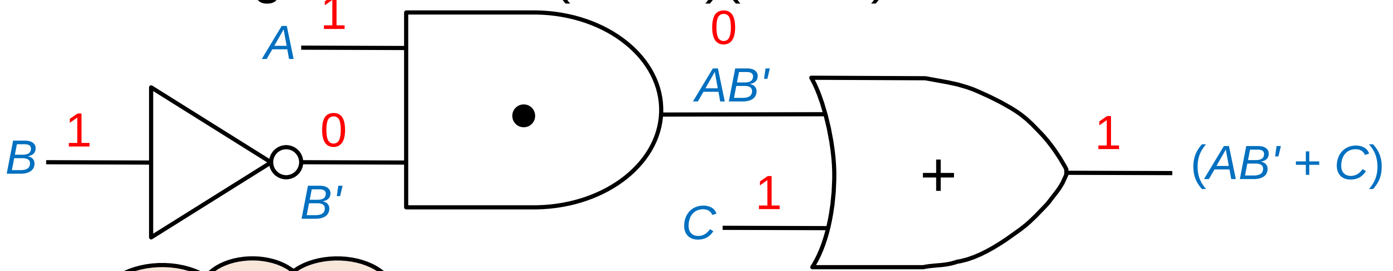
$\Rightarrow$ 1. $C + D$ → 2. $A(C + D)$ → 3. $[A(C + D)]'$
→ 4. BE → 5. $[A(C + D)]' + BE$



Boolean Expressions vs. Truth Tables

n variables $\Rightarrow 2^n$ rows

- A **truth table** specifies the output values of a Boolean expression for **all possible combinations of input values**
- **⇒ Check the equivalence between two expressions**
  - e.g., $AB' + C = (A + C)(B' + C)$

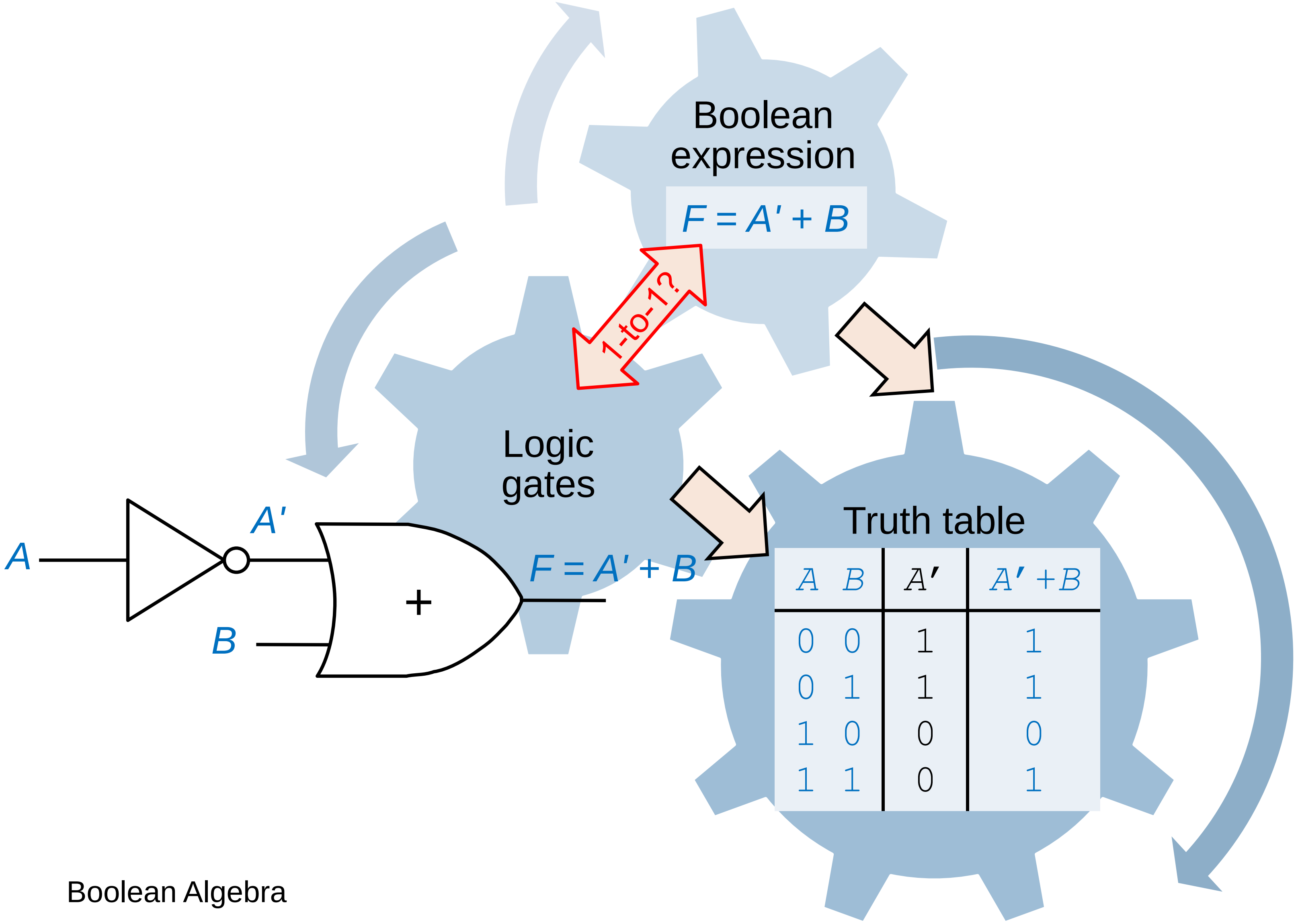


One function has different expressions

Expressions show the condition to make output ==1

A	B	C	B′	AB′	AB′ + C	A+C	B′+C	(A+C)(B′+C)
0	0	0	1	0	0	0	1	0
0	0	1	1	0	1	1	1	1
0	1	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1	1
1	0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	1	0	0
1	**1**	**1**	**0**	**0**	**1**	1	1	1

Example: $F = A' + B$



Boolean expression

$F = A' + B$

1-to-1?

Logic gates

$F = A' + B$

Truth table

A	B	A'	$A' + B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Boolean algebra differs from ordinary algebra in values, operations, laws

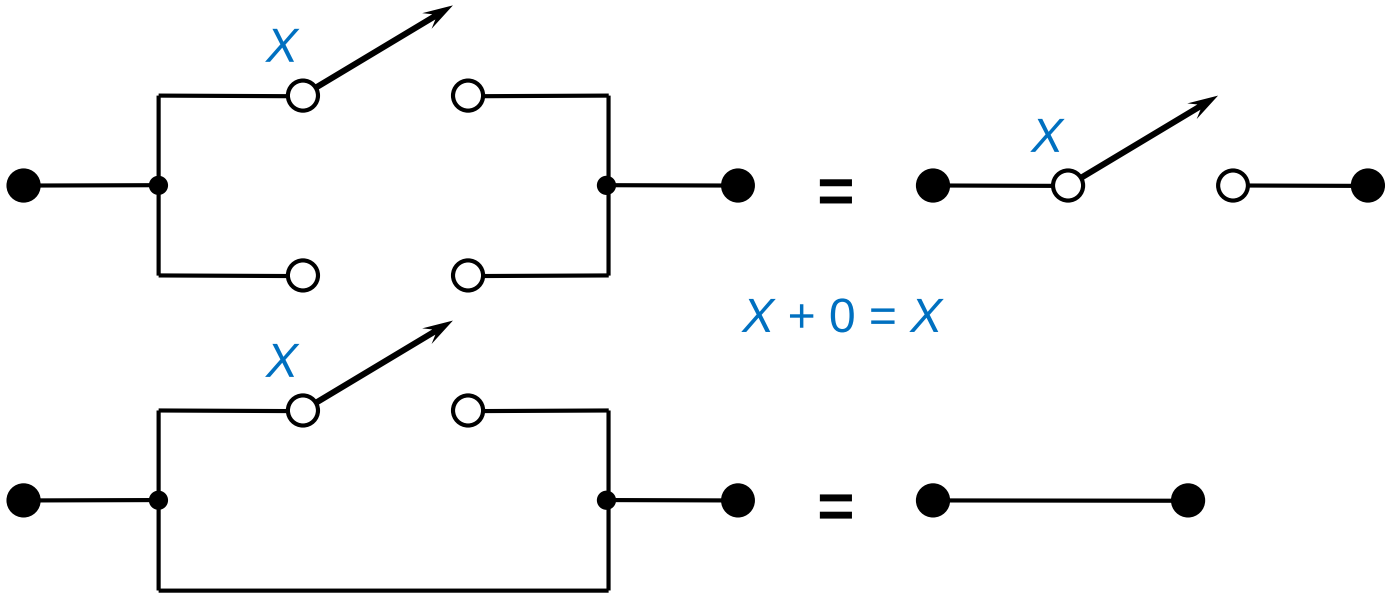
13

Theorems and Laws

Basic Theorems (1/3)

- **Operations with 0 and 1**
  - $X + 0 = X$
  - $X \bullet 1 = X$
  - $X + 1 = 1$
  - $X \bullet 0 = 0$
  - e.g., $(AB' + D)E + 1 = 1$

The variable *X* can be substituted by any expression

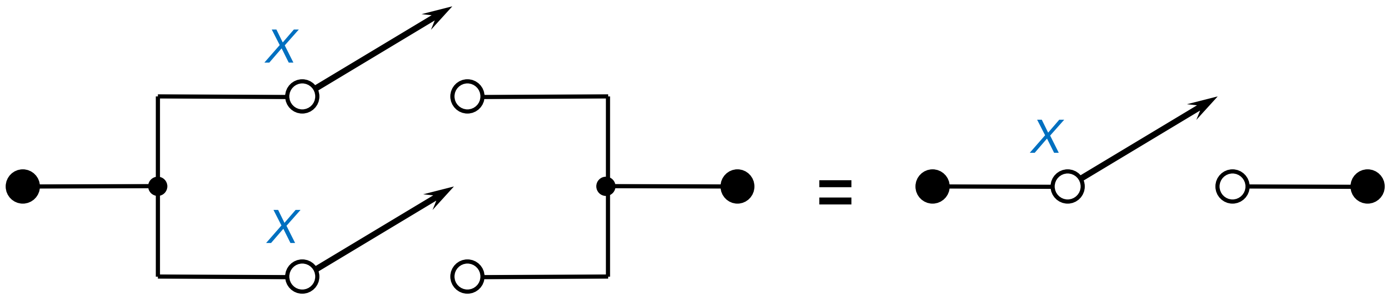


$X + 0 = X$

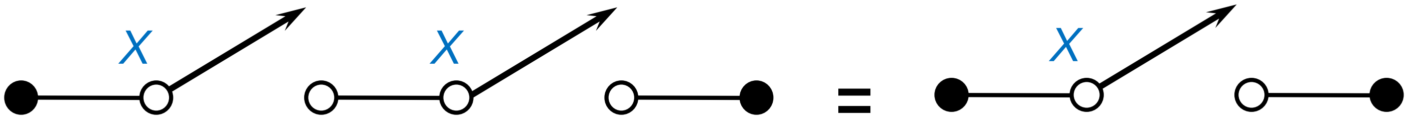
$X + 1 = 1$

Basic Theorems (2/3)

- **Idempotent laws**
  - $X + X = X$
  - $X \bullet X = X$



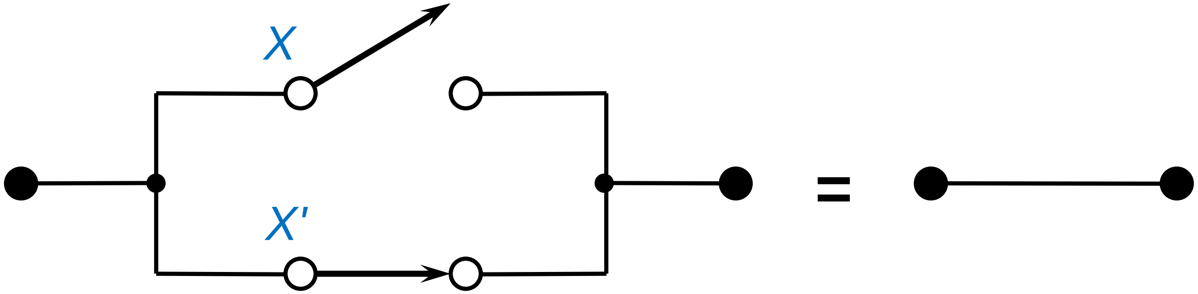
$X + X = X$



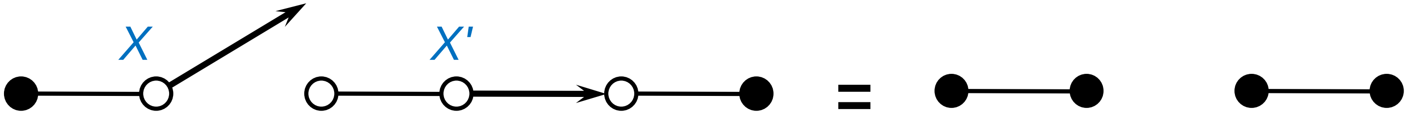
$X \bullet X = X$

Basic Theorems (3/3)

- **Involution law**
  - $(X')' = X$
- **Laws of complementarity**
  - $X + X' = 1$
  - $X \bullet X' = 0$
  - e.g., $(AB' + D)(AB' + D)' = 0$



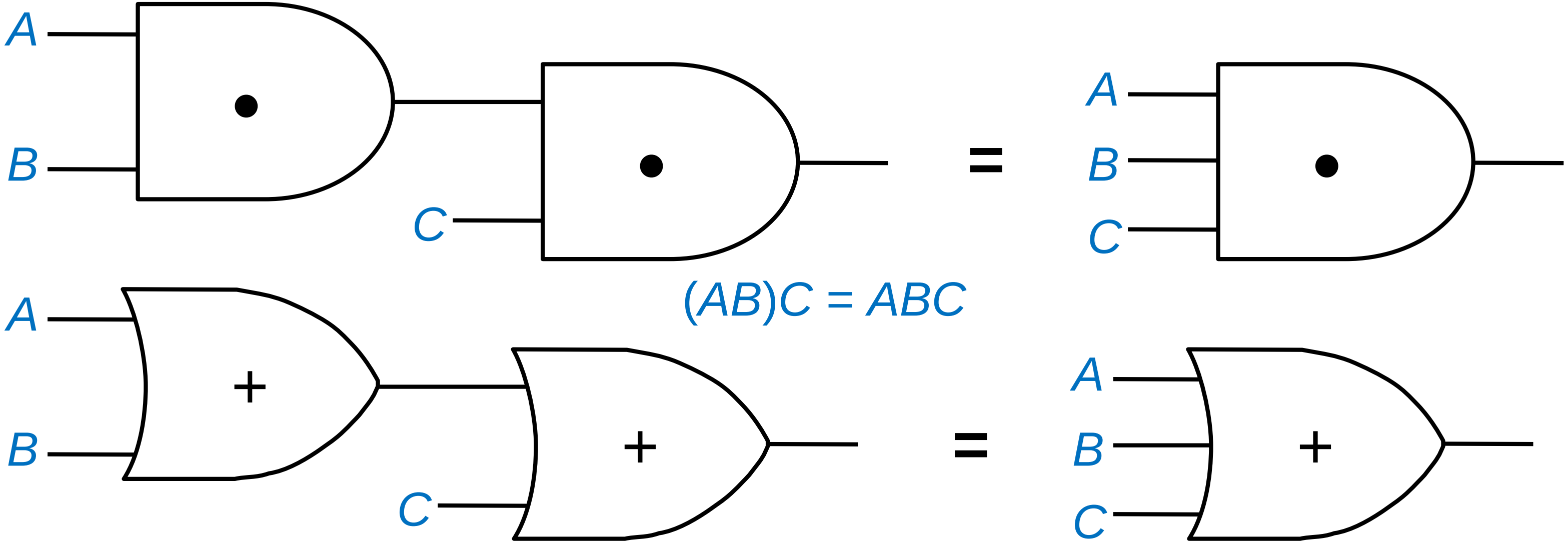
$X + X' = 1$



$X \bullet X' = 0$

Commutative/Associative Laws

- **Commutative laws for AND and OR**
  - $XY = YX$
  - $X + Y = Y + X$
- **Associative laws for AND and OR**
  - $(XY)Z = X(YZ) = XYZ$
  - $(X + Y) + Z = X + (Y + Z) = X + Y + Z$
  - e.g., 2-input gates ⇒ multiple-input gates

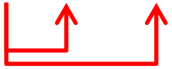


$(AB)C = ABC$

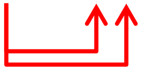
$(A + B) + C = A + B + C$

Distributive Laws (1/2)

- **Ordinary distributive law**
  - $X(Y + Z) = XY + XZ$



- **Second distributive law (Important !)**
  - $X + YZ = (X + Y)(X + Z)$



Only valid for Boolean algebra

  - Proof?

Distributive Laws (2/2)

- **Prove a Boolean theorem/law by:**

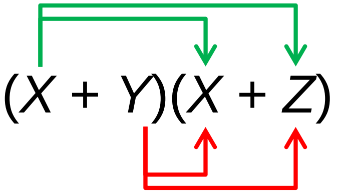
1. Truth table

$X + YZ = (X + Y)(X + Z)$

=?

X	Y	Z	YZ	$X+YZ$	$X+Y$	$X+Z$	$(X+Y)(X+Z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

2. Basic theorems



$(X + Y)(X + Z)$

$= X(X + Z) + Y(X + Z)$

$= XX + XZ + YX + YZ$

$= X + XZ + XY + YZ$

$= X\bullet 1 + XZ + XY + YZ$

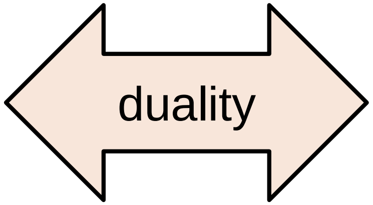
$= X(1+Z+Y) + YZ$

$= X\bullet 1 + YZ$

$= X + YZ$

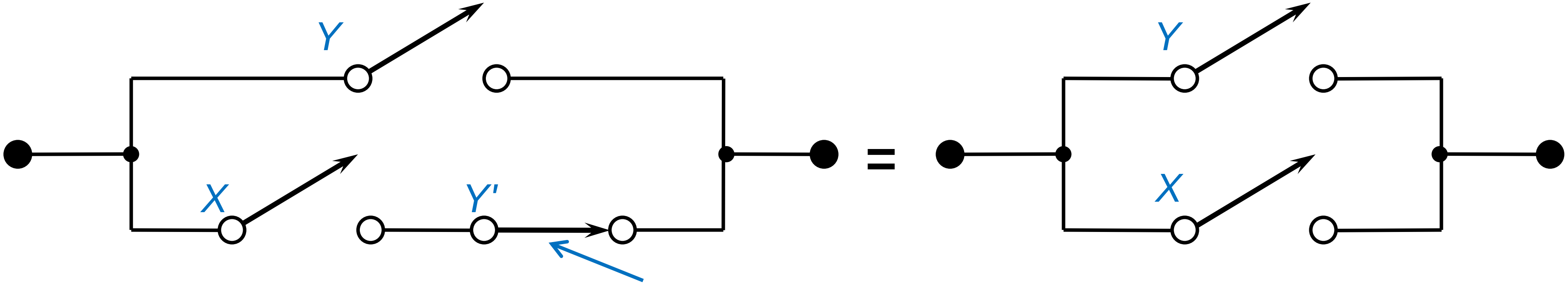
Simplification Theorems

- **Useful simplification theorems**
  - $XY + XY' = X$
  - $X + XY = X$
  - $(X+Y')\,Y = XY$



duality

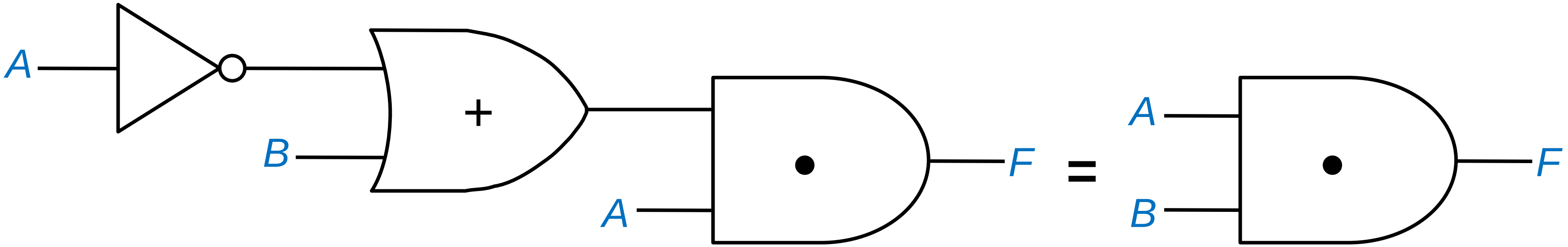
  - $(X+Y)(X+Y') = X$
  - $X(X+Y) = X$
  - $XY' + Y = X + Y$
- **Proof:**
  - $X + XY = X\bullet 1 + XY = X(1+Y) = X\bullet 1 = X$
  - $X(X+Y) = XX + XY = X + XY = X$
  - $XY' + Y = Y + XY' = (Y+X)(Y+Y') = (Y+X) \bullet 1 = Y + X$
    - Use switches



If switch Y open ⇒ switch Y' closed

Simplification Examples

1. $A(A'+B) = AB$



2. $Z = [A + B'C + D + EF][A + B'C + (D + EF)']$

$= [\underline{A + B'C} + \underline{D + EF}][\underline{A + B'C} + (\underline{D + EF})']$

$= [\ X + Y\ ][\ X + Y'\ ]$

$= X = A + B'C$

3. $Z = (AB + C)(B'D + C'E')+(AB + C)'$

$= (\underline{AB + C})(B'D + C'E')+(\underline{AB + C})'$

$= X\ Y + X' = XY + X' \bullet 1 = XY + X'(1+Y)$

$= XY + X' + X'Y = (X+X')Y + X'$

$= Y + X' = B'D + C'E'+(AB + C)'$

All roads lead to Rome

Multiplying Out

SOP: Sum of products of only single variables

- **Use the ordinary distributive law**

  $$X(Y + Z) = XY + XZ$$

  to multiply out an expression to obtain a sum-of-products form

  - e.g.,

    $AB' + CD'E + AC'E'$ (V)

    $A + B' + C + D'E$ (V)

    $(A + B)CD + EF$ (X)

Example: Multiplying Out $(A+BC)(A+D+E)$

1. **Multiply out completely and then eliminate redundant terms**

$$(A+BC)(A+D+E) = A+AD+AE+ABC+BCD+BCE$$
$$= A(1+D+E+BC)+BCD+BCE$$
$$= A+BCD+BCE$$

2. **Or, apply 2nd distributive law first: $(X+Y)(X+Z)=X+YZ$**

$$(A+BC)(A+D+E) = A+BC(D+E)$$
$$= A+BCD+BCE$$

2nd distributed law saves your time!

Factoring

POS: Products of sums of only single variables

- **Use the second distributive law**

  $$X + YZ = (X + Y)(X + Z)$$

  to factor an expression to obtain a product-of-sums form

  - e.g.,

    $(A + B')(C + D + E)(A + C' + E')$ (V)

    $(A + B)(C + D + E)F$ (V)

    $AB'C(D' + E)$ (V)

    $(A + B)(C + D) + EF$ (X)

Example: Factoring

1. **Factor *A + B'CD***

$$A + B'CD = (A + B')(A + CD) = (A + B')(A + C)(A + D)$$

2. **Factor *AB'+ C'D***

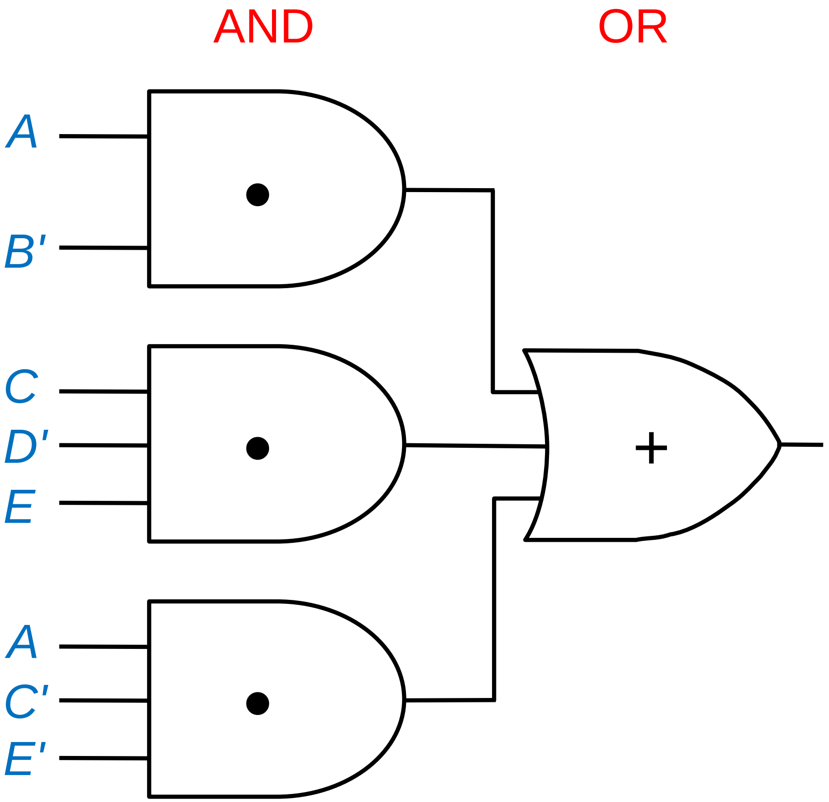
$$AB' + C'D = (AB' + C')(AB' + D) = (A + C')(B' + C')(A + D)(B' + D)$$

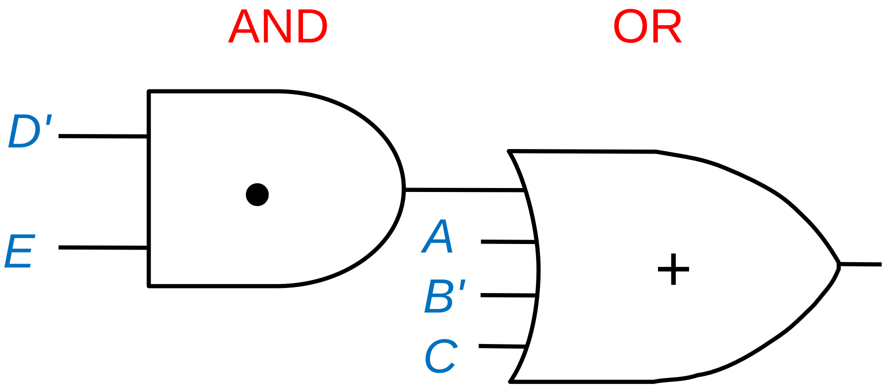
Iteratively apply 2nd distributed law

3. **DIY: Factor *C'D + C'E' + G'H***

SOP vs. Logic Gates

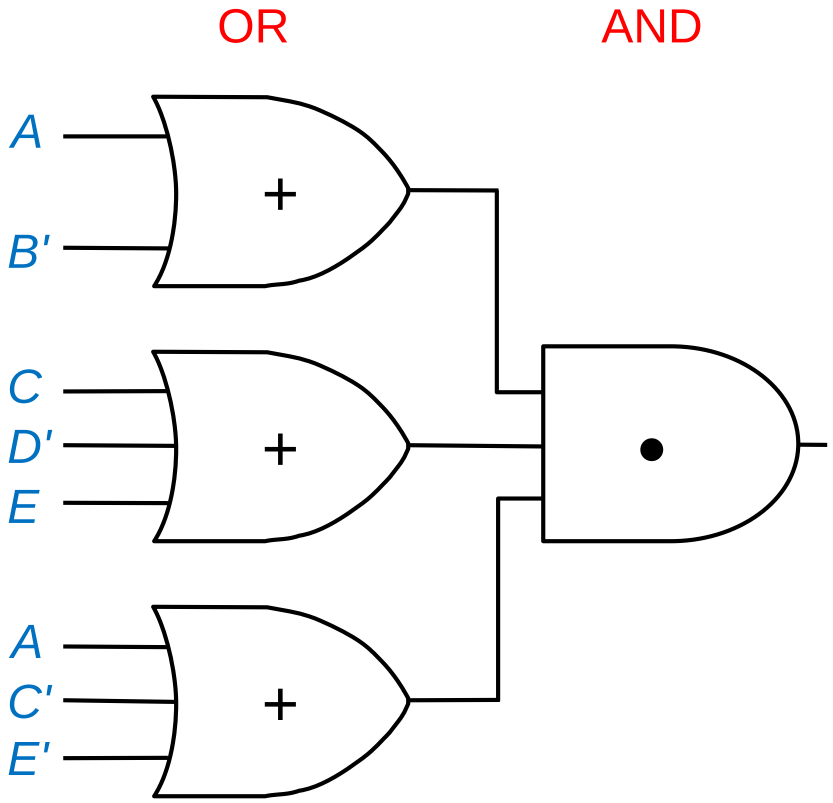
- **Realize SOPs by two-level circuits (AND-OR)**
  - $AB' + CD'E + AC'E'$
  - $A + B' + C + D'E$

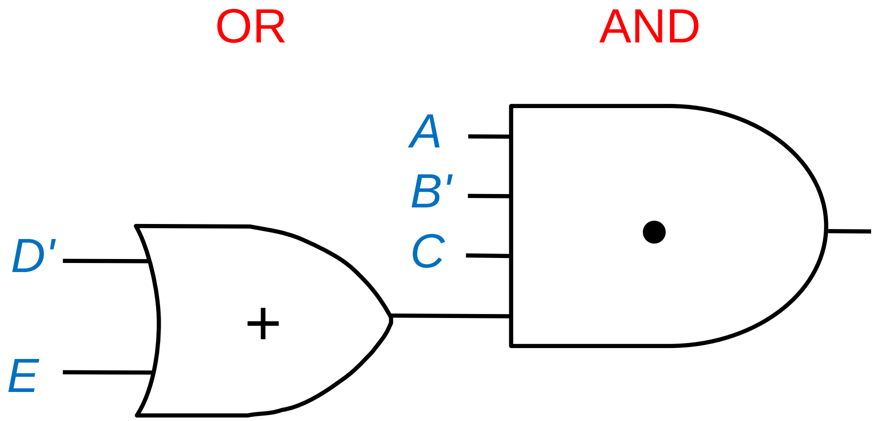




POS vs. Logic Gates

- **Realize POSs by two-level circuits (OR-AND)**
  - $(A + B')(C + D + E)(A + C' + E')$
  - $AB'C(D' + E)$





DeMorgan's Laws

- **Complement a Boolean expression by DeMorgan's laws**
  - $(X + Y)' = X'Y'$
  - $(XY)' = X' + Y'$
  - Proof: By truth table

X	Y	X'	Y'	$X+Y$	$(X+Y)'$	$X'Y'$	XY	$(XY)'$	$X'+Y'$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

- **Generalize to *n* variables:**
  - $(X_1+X_2+\ldots+X_n)' = X_1'X_2'\ldots X_n'$
  - $(X_1X_2\ldots X_n)' = X_1'+X_2'+\ldots+ X_n'$
- **One-step rule:**
  - $[f(x_1,x_2,\ldots,x_n,0,1,+,\bullet)]' = f(x_1', x_2',\ldots,x_n',1,0,\bullet,+)$
    - $x \leftrightarrow x'$; $+ \leftrightarrow \bullet$; $0 \leftrightarrow 1$

Example: Complementing $(AB' + C)D' + E$

1. **Iteratively apply DeMorgan’s laws:**

$$
\begin{aligned}
[(AB' + C)D' + E]' &= [(AB' + C)D'\,]'E' \\
&= [(AB' + C)'+D]E' \\
&= [(AB')'C'+D]E' \\
&= [(A'+B)C'+D]E'
\end{aligned}
$$

NOT is applied only to single variables

2. **Or, use one-step rule:**

$$
\begin{aligned}
[(AB' + C)D' + E]' &= [(((A \bullet B')+C) \bullet D')+ E]' \\
&= (((A'+B) \bullet C') +D\,) \bullet E'
\end{aligned}
$$

Duality

- **Dual:**
  - $[f(x_1,x_2,\ldots,x_n,0,1,+,\bullet)]^D = f(x_1,x_2,\ldots,x_n,1,0,\bullet,+)$
    - $+ \leftrightarrow \bullet$; $0 \leftrightarrow 1$
- **Cf. DeMorgan's laws:**
  - $[f(x_1,x_2,\ldots,x_n,0,1,+,\bullet)]' = f(x_1',x_2',\ldots,x_n',1,0,\bullet,+)$
    - $x \leftrightarrow x'$; $+ \leftrightarrow \bullet$; $0 \leftrightarrow 1$
- **⇒ Find the dual of an expression:**
  - Complement the entire expression
  - Complement each individual variable
- **e.g., $(XYZ\ldots)^D = X + Y + Z + \ldots$**
- **e.g., $(AB' + C)^D = ?$**

  $(AB' + C)' = (A'+B)C' \Rightarrow (AB' + C)^D = (A+B')C$
- **Application: $F = G \Leftrightarrow F^D = G^D$**