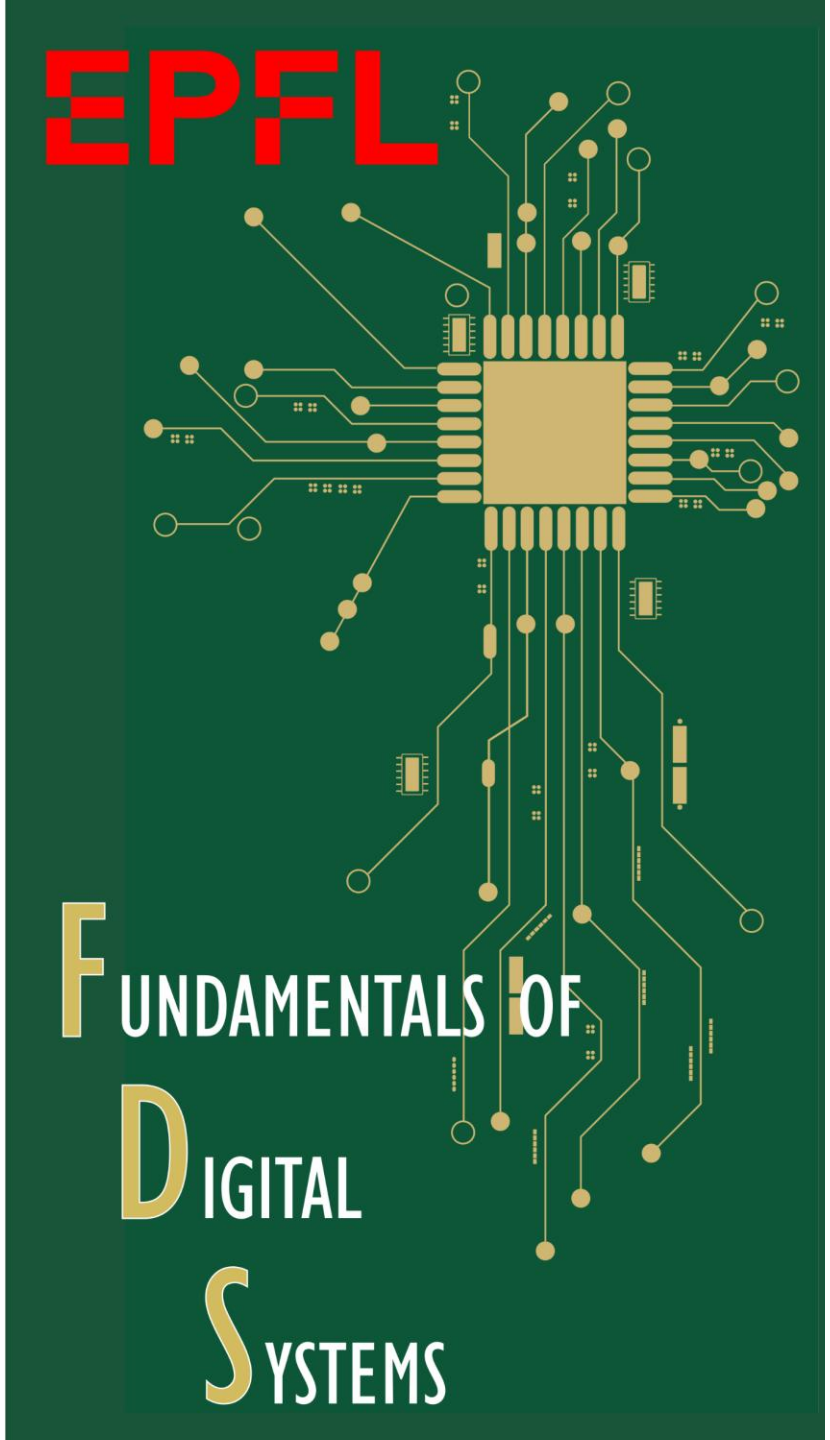


Digital Logic Circuits

Logic Functions, Logic Gates, Boolean Algebra

CS-173 Fundamentals of Digital Systems
Mirjana Stojilović

Spring 2025

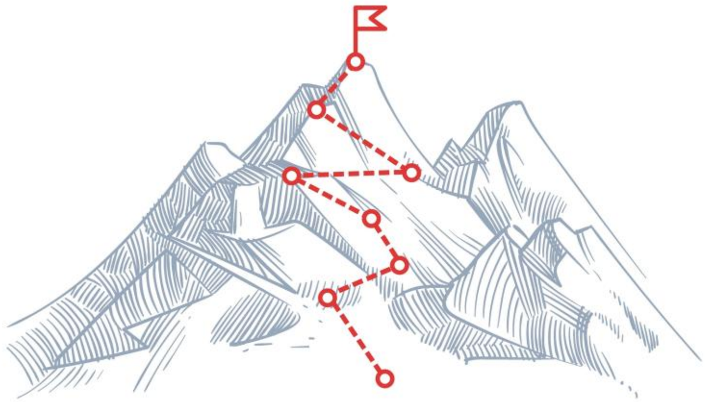
Introduction to Logic Circuits

- Logic circuits: the foundations of digital systems
  - In smartphones; computers; control systems; digital communication devices; … (the list is endless)
- The smallest unit of digital information is one bit, represented as a binary value **0 and 1**
- In a binary logic circuit, the electrical signals are constrained to two discrete values
  - The key to binary circuits dominance is **simplicity**
  - In practice, the two discrete values are implemented as voltage levels (the supply voltage or the ground)

Let’s Talk About…

…Logic circuits, which form
the foundation of digital systems

Learning Outcomes



- Discover basic digital logic gates and use them to build logic networks
- Describe logic circuit operation through
  - Truth tables
  - Timing diagrams
- Learn Boolean algebra axioms/theorems/properties
  - Check logic function equivalence
  - Find more efficient logic circuit implementations

Quick Outline

Variables and Functions

The Simplest Binary Logic Element

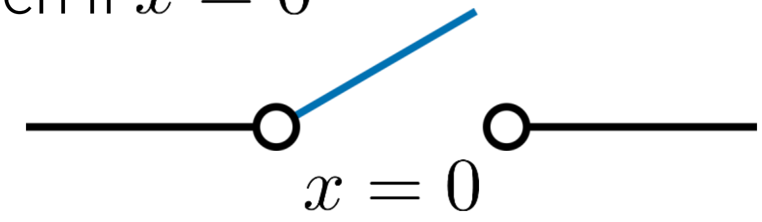
- …a **switch** that has two states
  - Open

  - Closed

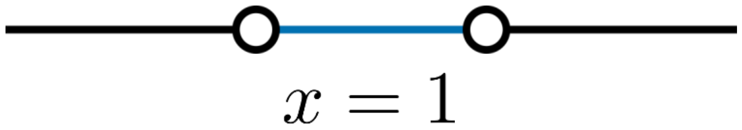
- In practice, implemented as transistors
  - A topic of another lecture

Two States of a Switch

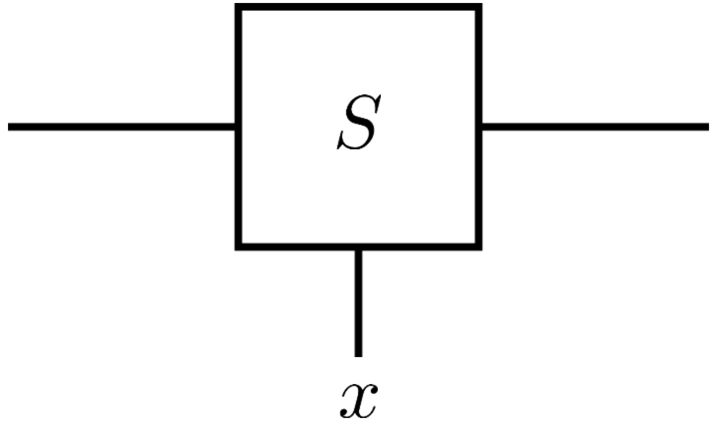
- If controlled by an input variable x, the switch is
  - Open if $x = 0$



  - Closed if $x = 1$

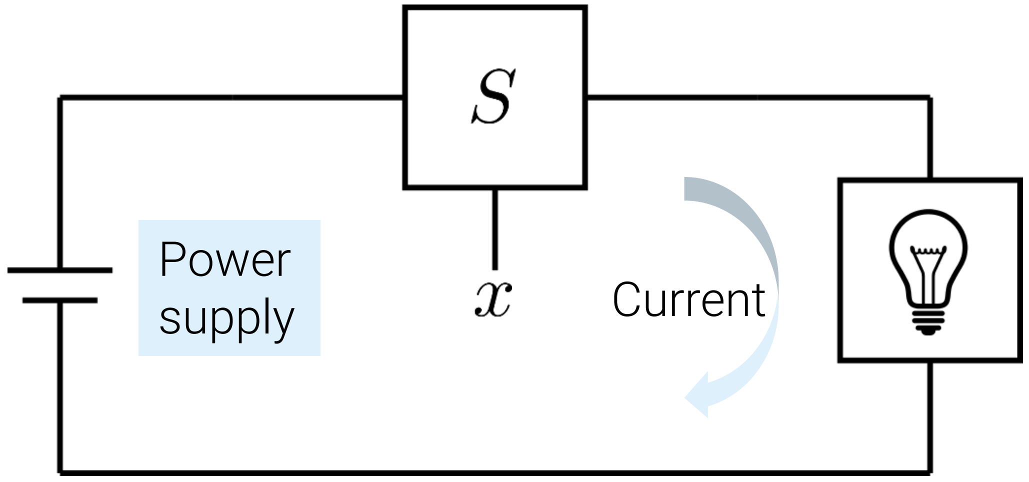


- The symbol for a switch controlled by an input variable

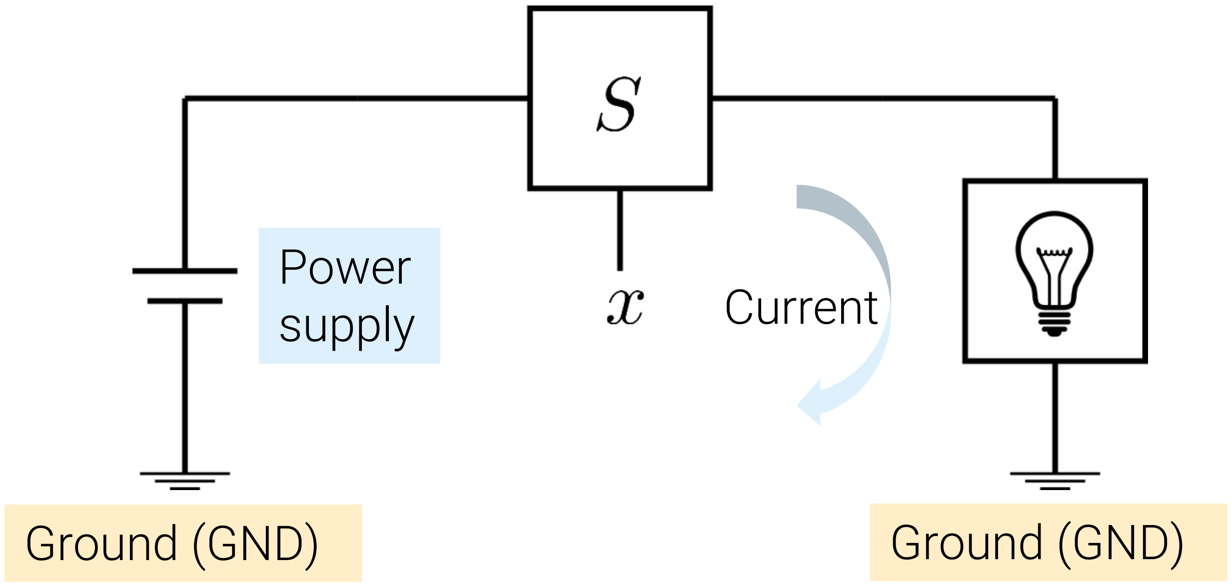


A Light Controlled by a Single Switch

- Simple connection to battery
  - Explicit return path connection

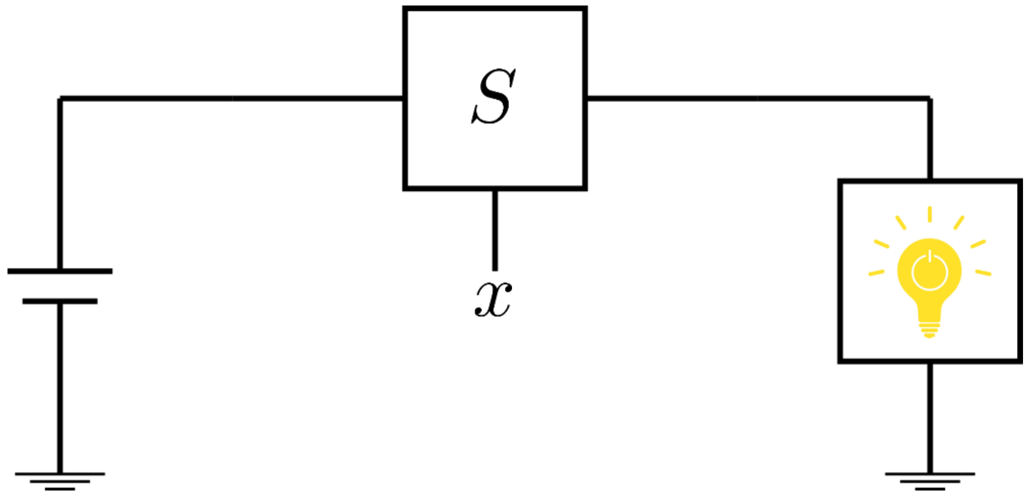


- Simple connection to battery
  - Using a ground connection as the return path (simplified view)

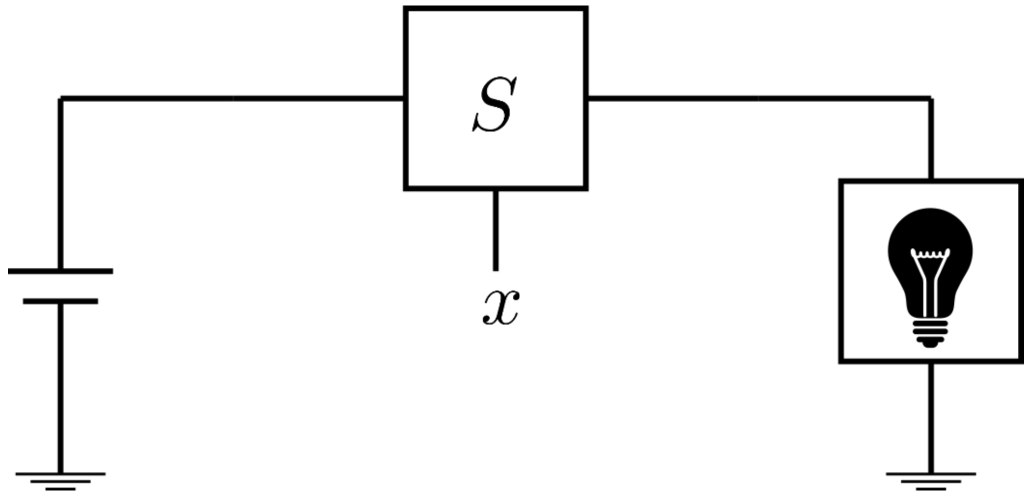


A Light Controlled by a Single Switch
Contd.

- When $x = 1$, the switch is closed, the current flows, the light is ON



- When $x = 0$, the switch is open, the current does not flow, the light is OFF



A Light Controlled by a Single Switch
Logic function

- The **output** is defined as the state (or condition) of the light, L
  - If the light is ON, we will say that $L = 1$
  - Otherwise, $L = 0$

- We can describe the state of the light L with a **logical expression**

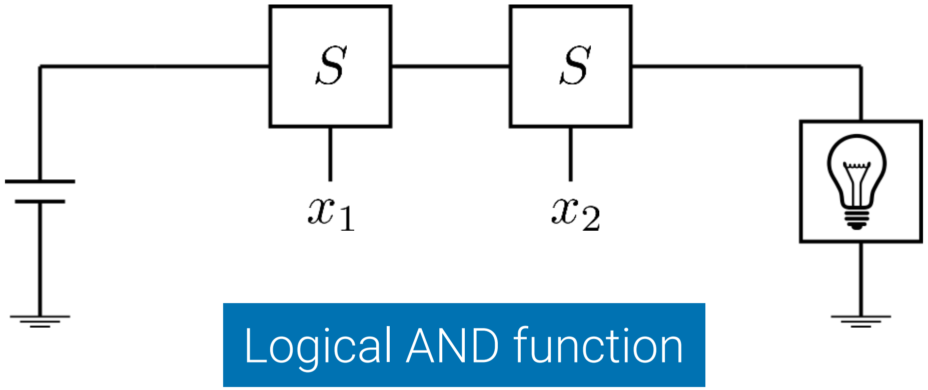
$$L(x) = x$$

- We say that L is a **logic function** of the **input variable** x

Two-Variable Logic Functions

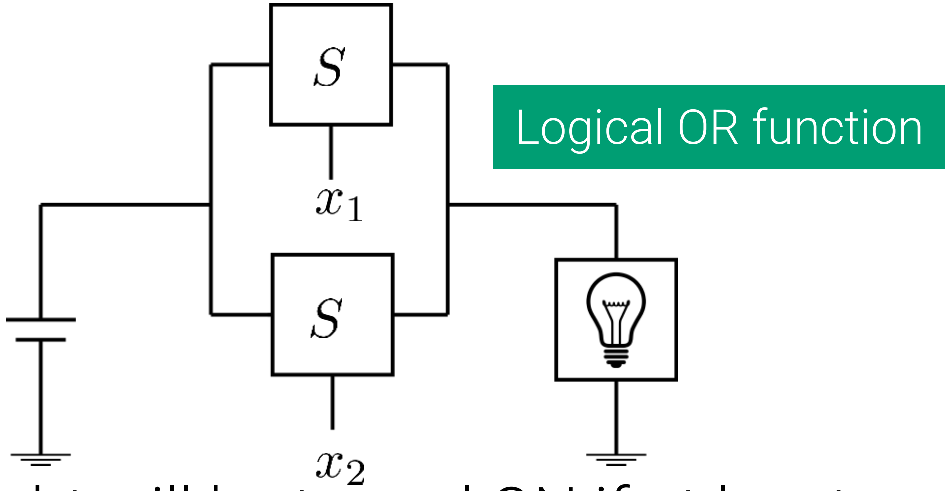
Series and Parallel Connections

- Consider the possibility of using two switches to control the light
  - Series connection:



  - Light will be turned ON only if both (the left **and** the right) switches are closed

  - Parallel connection:



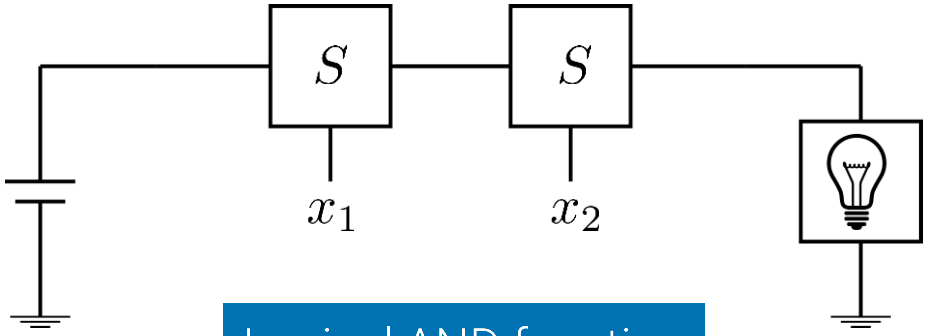
  - Light will be turned ON if at least one (the upper **or** the lower) of the two switches is closed

Logical AND and OR Functions

Series and Parallel Connections, Contd.

- Consider the possibility of using two switches to control the light

- Series connection:



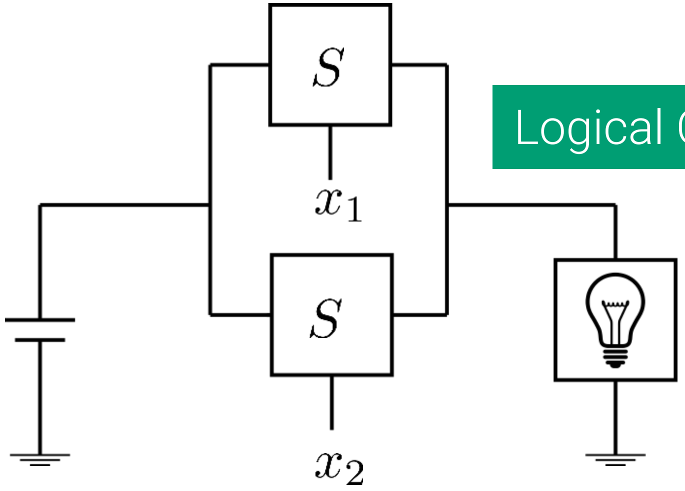
Logical AND function

AND operator

$$L(x_1, x_2) = x_1 \cdot x_2$$

where $L = 1$ if $x_1 = 1$ and $x_2 = 1$,
$L = 0$ otherwise.

- Parallel connection:



Logical OR function

OR operator

$$L(x_1, x_2) = x_1 + x_2$$

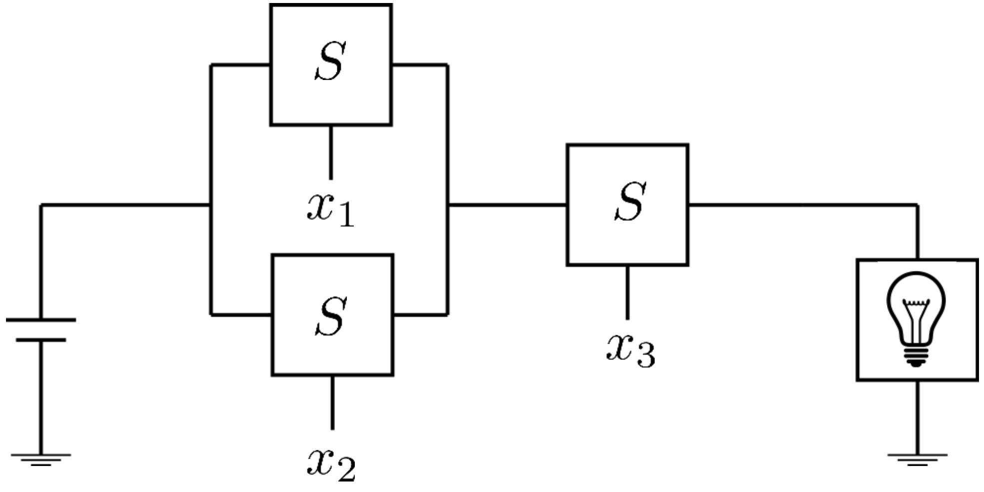
where $L = 1$ if $x_1 = 1$ or $x_2 = 1$, or if $x_1 = x_2 = 1$,
$L = 0$ if $x_1 = x_2 = 0$.

EXAMPLES

Two-Variable Logic Functions

Series-Parallel Connections

- The AND and OR functions are two of the most important logic functions and can be used (together with some other simple functions) as building blocks of all logic circuits
  - Example: A series-parallel connection of three switches



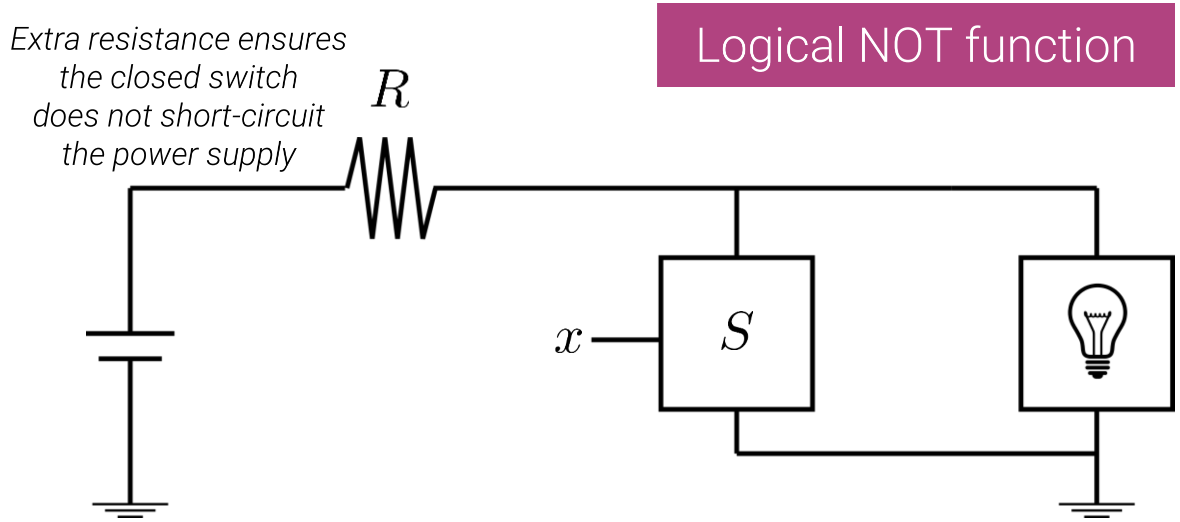
$$L(x_1, x_2, x_3) = (x_1 + x_2) \cdot x_3$$

where $L = 1$ if $x_3 = 1$ and, at the same time, at least one of the x_1 and x_2 is equal to 1.

Logic Complement Operation
Inversion

- We assumed positive action occurs when the switch is closed (x = 1: light ON)
- It is equally relevant to consider the contrary: positive action occurs when the switch is open (x = 0: light ON)
- **Complement (NOT)** is the 3rd most basic logic operation

Logical NOT function

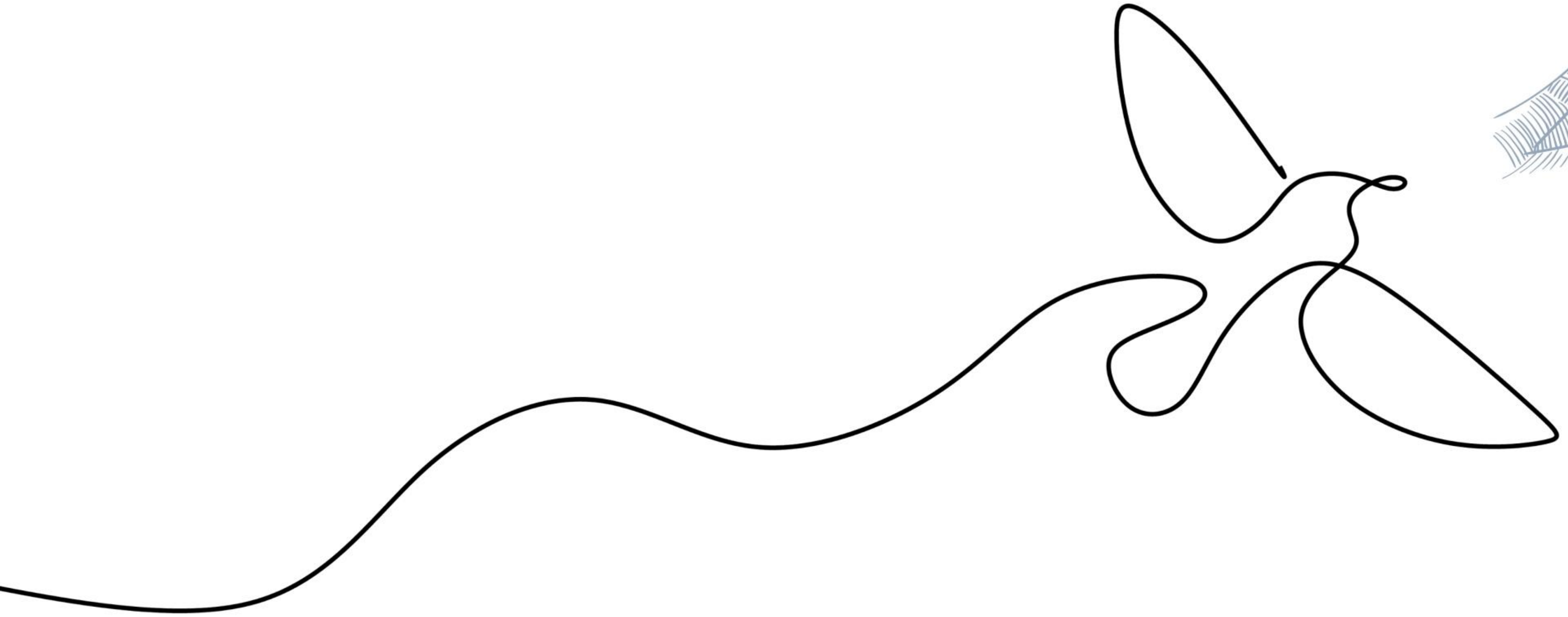


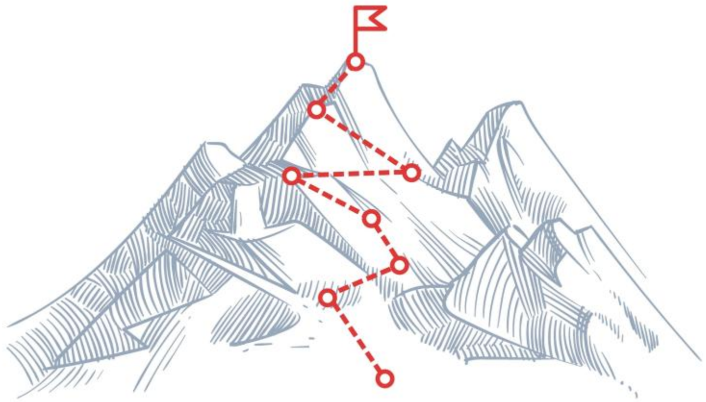
$$L(x) = \bar{x}$$

NOT (complement) operator

where $L = 1$ if $x = 0$

$L = 0$ if $x = 1$.





Truth Tables

Logic AND and OR Operations

Truth Tables

- Logical operations can be defined in the form of a **truth table**
  - **AND**

$$L(x_1, x_2) = x_1 \cdot x_2$$
$$\text{where } L = 1 \text{ if } x_1 = 1 \text{ and } x_2 = 1,$$
$$L = 0 \text{ otherwise.}$$

  - **OR**

$$L(x_1, x_2) = x_1 + x_2$$
$$\text{where } L = 1 \text{ if } x_1 = 1 \text{ or } x_2 = 1, \text{ or if } x_1 = x_2 = 1,$$
$$L = 0 \text{ if } x_1 = x_2 = 0.$$

		AND	OR
x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	**0**
0	1	0	1
1	0	0	1
1	1	**1**	1

Logic AND and OR Operations

Truth Tables

- For n logic variables, there are 2^n rows in the **truth table**
  - **AND**

$$L(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3$$
$$\text{where } L = 1 \text{ if } x_1 = x_2 = x_3 = 1,$$
$$L = 0 \text{ otherwise.}$$

  - **OR**

$$L(x_1, x_2, x_3) = x_1 + x_2 + x_3$$
$$\text{where } L = 0 \text{ if } x_1 = x_2 = x_3 = 0,$$
$$L = 1 \text{ otherwise.}$$

			AND	OR
x_1	x_2	x_3	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$
0	0	0	0	**0**
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	**1**	1

Logic Complement Operation

Truth Tables

- Logical operations can be defined in the form of a **truth table**
  - NOT

$$L(x) = \bar{x}$$
$$\text{where } L = 1 \text{ if } x = 0$$
$$L = 0 \text{ if } x = 1.$$

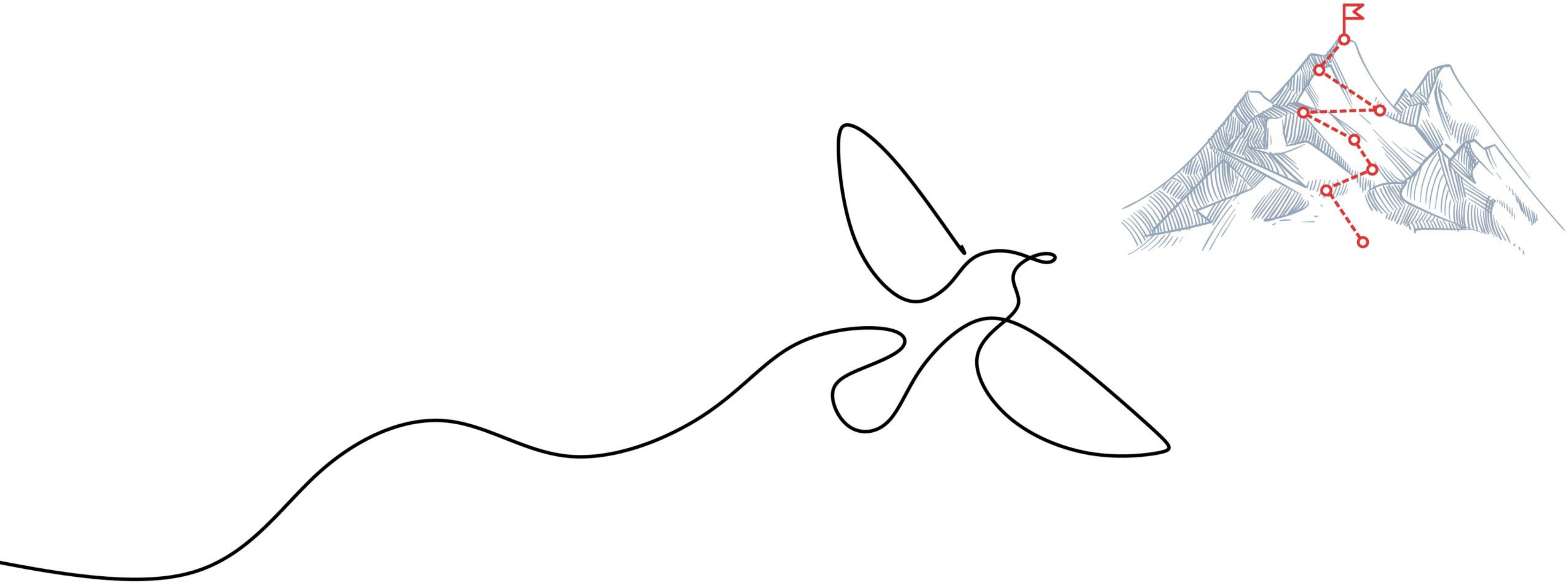
	NOT
x	$\bar{x}$
0	1
1	0

Precedence of Operations

- Parentheses can be used to indicate the order of operations
- Alternatively, to help the **readability** of logic expressions by reducing the number of parentheses, a convention states:
  - In the absence of parentheses, operations in a logic expression must be performed in the order **first NOT, then AND, and then OR**
  - Example:

$$x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2} \equiv ((x_1 \cdot x_2) + ((\overline{x_1}) \cdot (\overline{x_2}))) \equiv x_1 x_2 + \overline{x_1}\ \overline{x_2}$$

    - First, complements $\overline{x_1}, \overline{x_2}$ are generated
    - Then, the product terms $x_1 \cdot x_2$ and $\overline{x_1} \cdot \overline{x_2}$ are formed
    - Lastly, the sum of the two product terms is generated
    - Note: We can omit the multiplication symbol

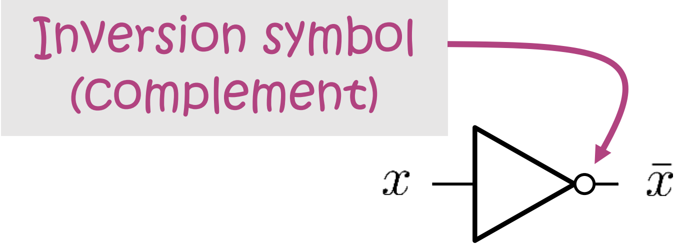
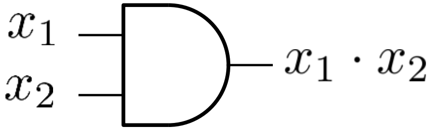
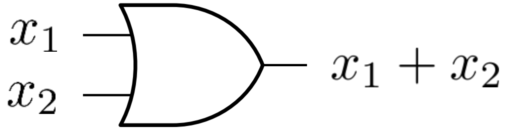
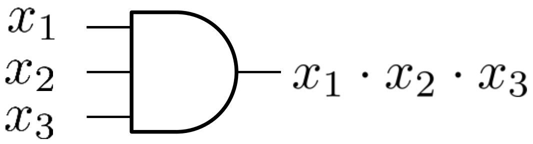
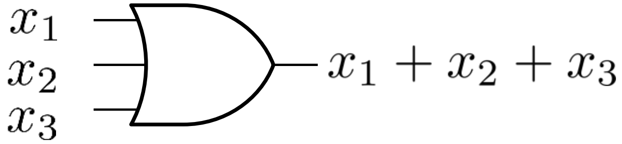
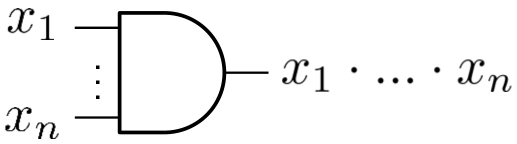
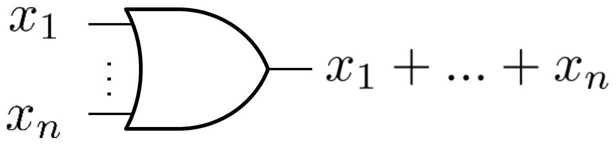


Logic Gates

Logic Gates and Networks

- AND/OR/NOT can implement logic functions of any complexity
- Electronically, the operations are implemented with transistors, resulting in a circuit called a **logic gate**
- A logic gate has
  - One or more **inputs**
  - An **output**, which is a function of the inputs
- To visualize **logic circuits** (i.e., networks of logic gates), we draw schematics composed of logic gates

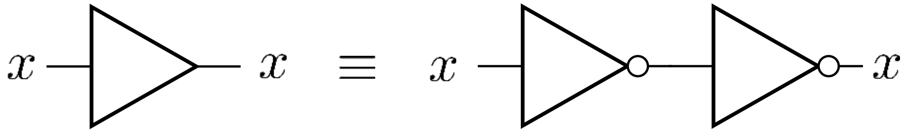
AND/OR/NOT Graphical Symbols

NOT gate	AND gate	OR gate
 $f(x) = \bar{x}$	 $f(x_1, x_2) = x_1 \cdot x_2$	 $f(x_1, x_2) = x_1 + x_2$
	 $f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3$	 $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$
	 $f(x_1, ..., x_n) = x_1 \cdot ... \cdot x_n$	 $f(x_1, ..., x_n) = x_1 + ... + x_n$

Variants of Single-Input Gates

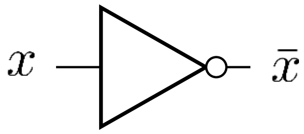
Inverter, Buffer

- **Buffer**, passes the input to the output unchanged



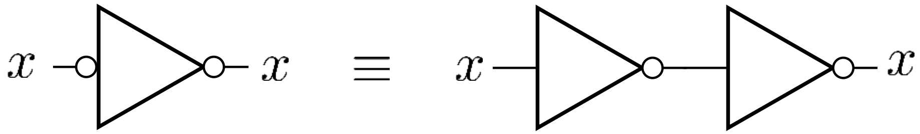
$$f(x) = x$$

- Inverter, passes the input to the output after inverting its polarity



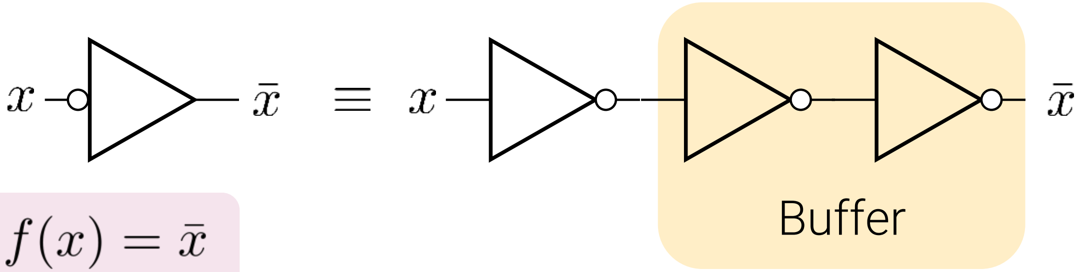
$$f(x) = \bar{x}$$

- Buffer, with in/out inversion



$$f(x) = x$$

- Buffer, with input inversion

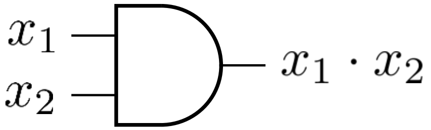


$$f(x) = \bar{x}$$

Variants of AND Gates

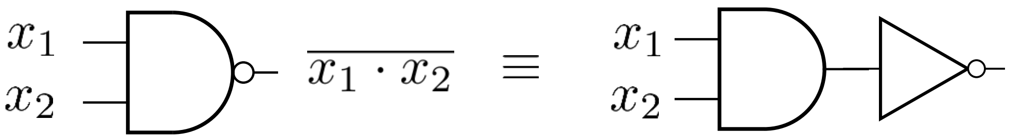
NAND gate

- AND gate, basic



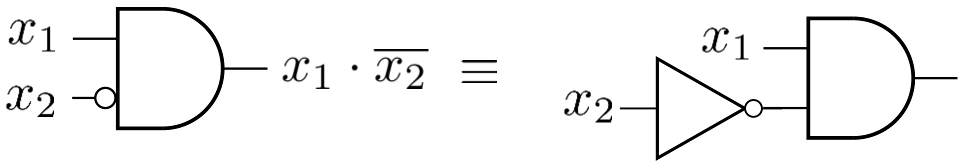
$$f(x_1, x_2) = x_1 \cdot x_2$$

- **NAND gate**, equiv. to AND gate with the output inverted



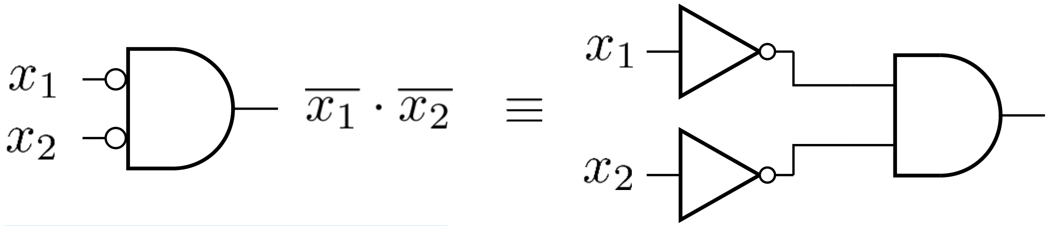
$$f(x_1, x_2) = \overline{x_1 \cdot x_2}$$

- AND gate, one input inverted



$$f(x_1, x_2) = x_1 \cdot \overline{x_2}$$

- AND gate, both inputs inverted

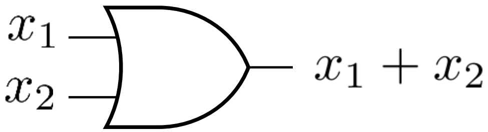


$$f(x_1, x_2) = \overline{x_1} \cdot \overline{x_2}$$

Variants of OR Gates

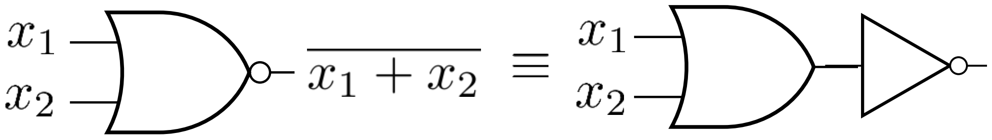
NOR gate

- OR gate, basic



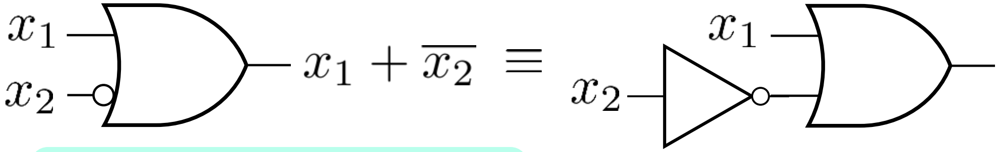
$$f(x_1, x_2) = x_1 + x_2$$

- **NOR gate**, equiv. to OR gate with the output inverted



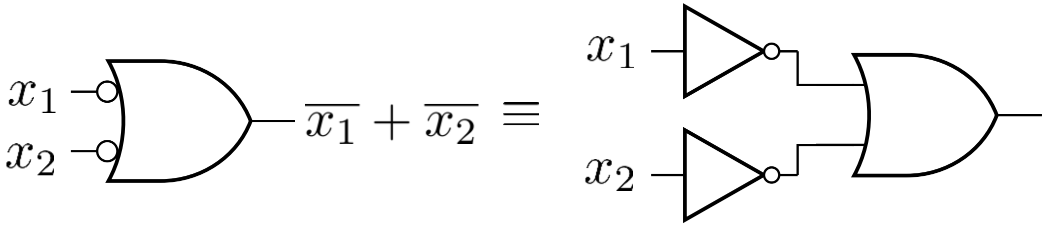
$$f(x_1, x_2) = \overline{x_1 + x_2}$$

- OR gate, one input inverted



$$f(x_1, x_2) = x_1 + \overline{x_2}$$

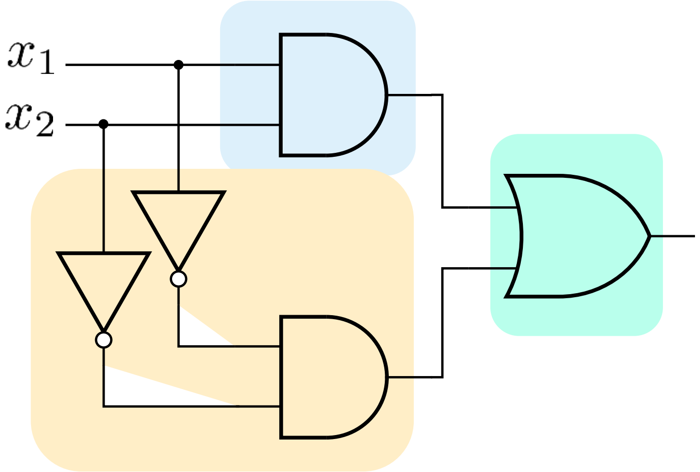
- OR gate, both inputs inverted



$$f(x_1, x_2) = \overline{x_1} + \overline{x_2}$$

Example Logic Network

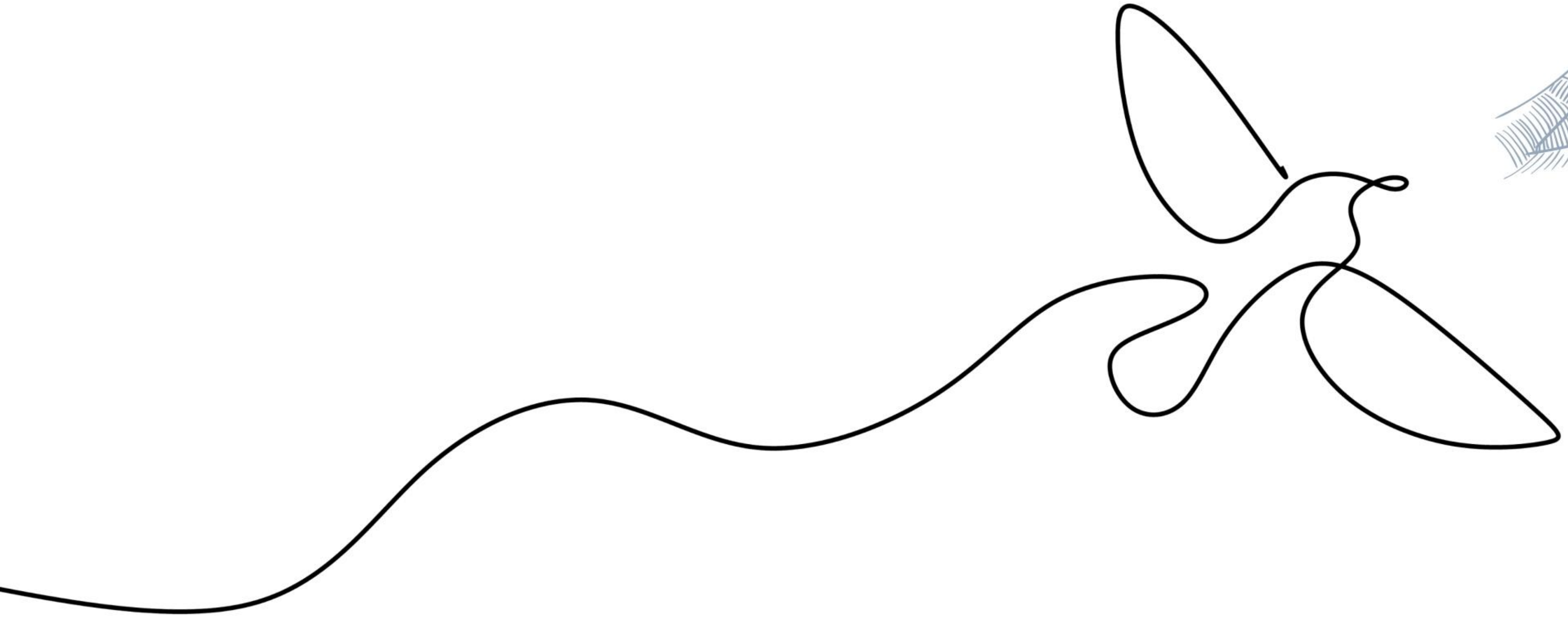
- AND/OR/NOT can implement logic functions of any complexity

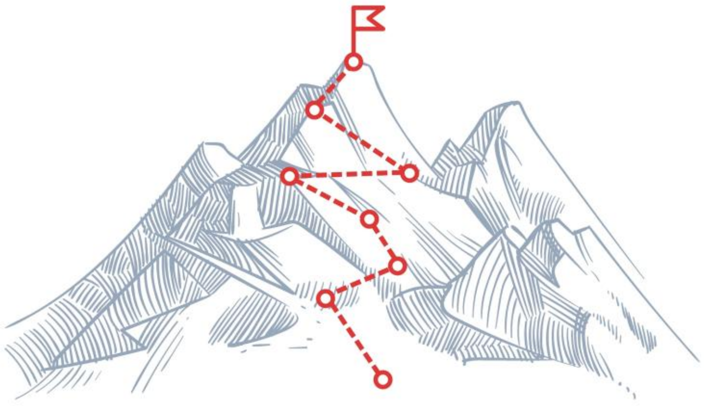


$$f(x_1, x_2) = x_1 x_2 + \overline{x_1}\ \overline{x_2}$$

$$f(0,0) = 0 \cdot 0 + \overline{0} \cdot \overline{0} = 1$$
$$f(0,1) = 0 \cdot 1 + \overline{0} \cdot \overline{1} = 0$$
$$f(1,0) = 1 \cdot 0 + \overline{1} \cdot \overline{0} = 0$$
$$f(1,1) = 1 \cdot 1 + \overline{1} \cdot \overline{1} = 1$$

x_1	x_2	$f(x_1, x_2)$
0	0	**1**
0	1	0
1	0	0
1	1	**1**

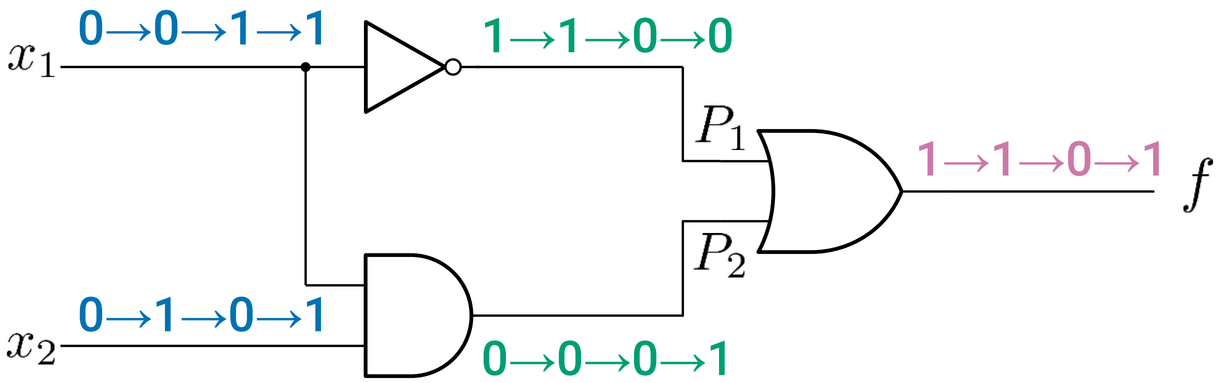




Analysis of a Logic Network

Analysis of a Logic Network

- Example logic network
  - The sequence of input values in the truth table visualized in the network
  - Any sequence can be visualized in a **timing diagram**

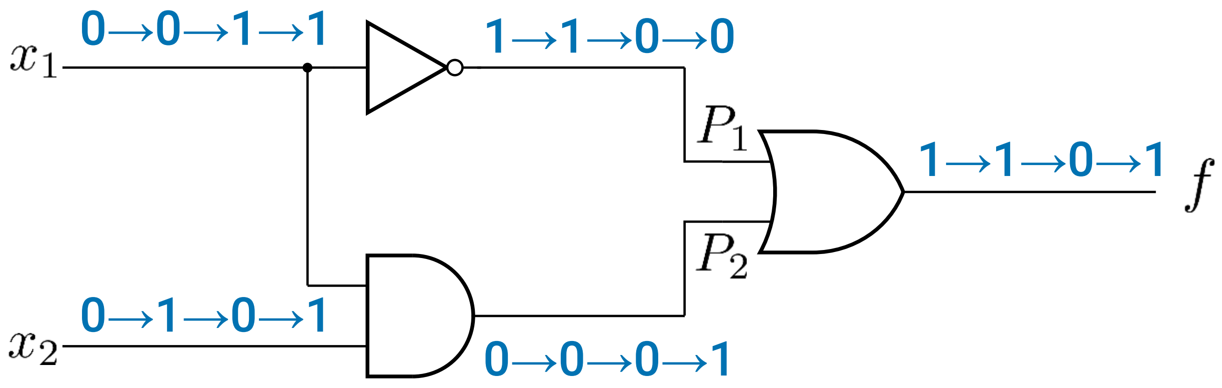


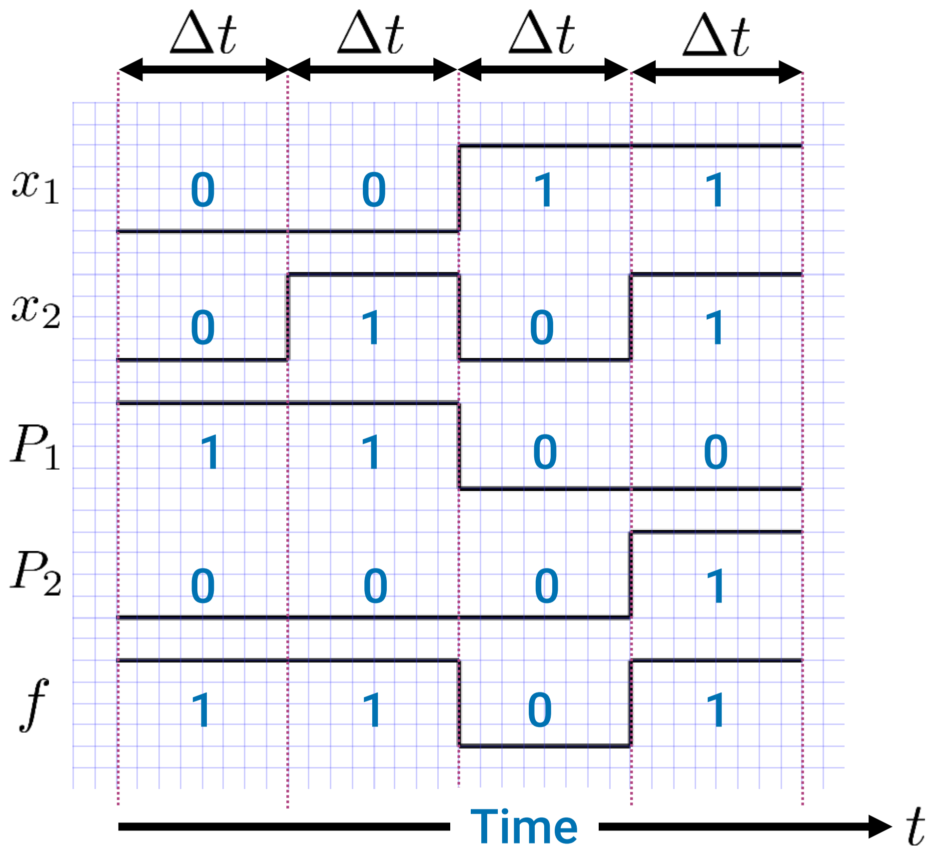
x_1	x_2	$f(x_1, x_2)$	P_1	P_2
0	0	1	1	0
0	1	1	1	0
1	0	0	0	0
1	1	1	0	1

$$f(x_1, x_2) = \overline{x_1} + x_1 x_2$$

Timing Diagram

- The **timing diagram** shows the changes in waveforms of the internal signals of a logic network and its outputs resulting from the inputs changing their values over time





- Note: In reality, transitions between logical levels take some time and gates may have different delays

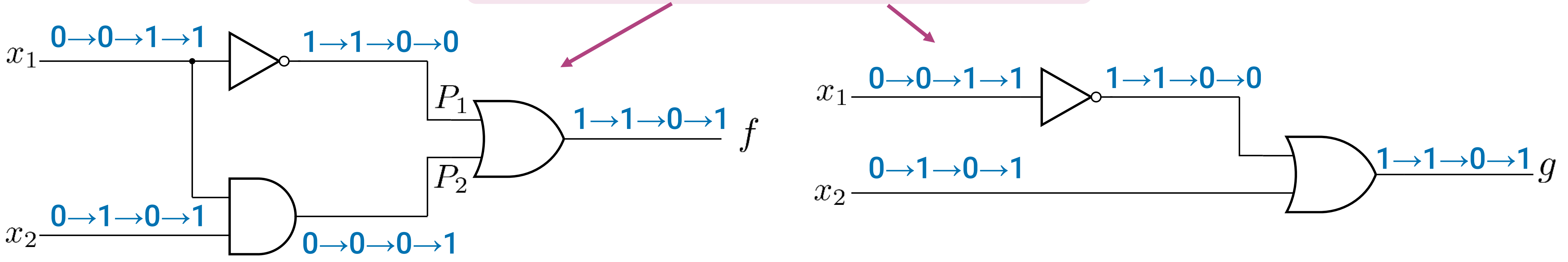
Cost of a Logic Circuit

- The total cost of a logic circuit is typically defined as the total **number of gates** **plus** the total **number of gate inputs**
  - Each logic gate (AND, OR, NOT, etc.) contributes to the cost
  - More inputs to gates often mean larger, more costly gates
  - In simplified cost models, only the number of gates might be considered
  - In detailed cost models, weights may be assigned to different types of gates, depending on their complexity or physical implementation

Functionally Equivalent Networks

- A logic function can be implemented with a variety of different logic networks of different cost

$$f(x) = \overline{x_1} + x_1x_2 = \overline{x_1} + x_2 = g(x)$$



- The above two networks are functionally **equivalent**
  - For the same input sequence, they produce the same output sequence

How To Check for Equivalence?

$$f(x_1, ..., x_n) = g(x_1, ..., x_n), \forall x_1, ..., x_n$$

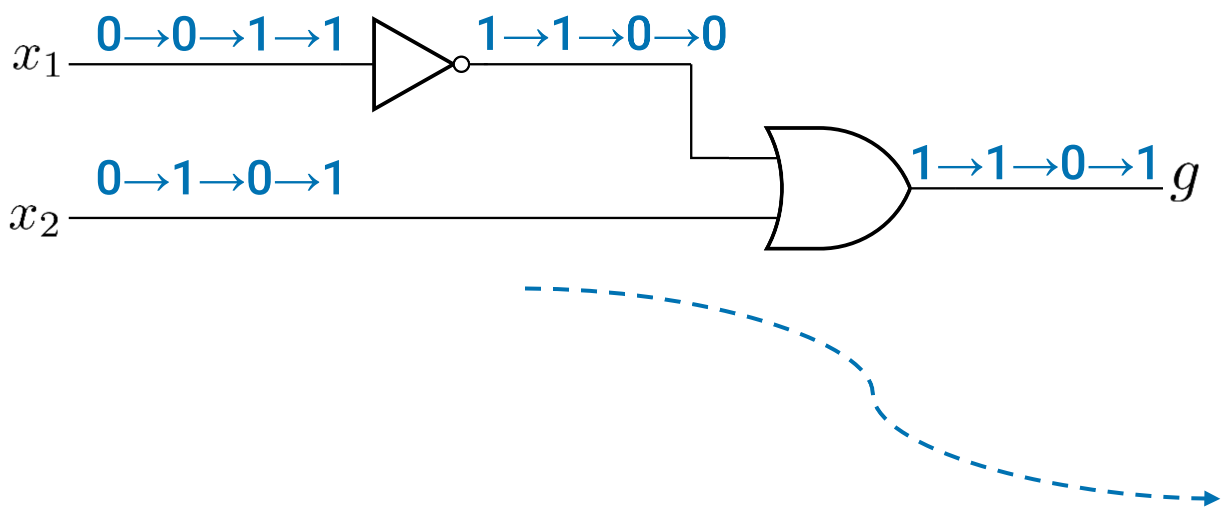
- Two logic networks are equivalent if
  - Their **truth tables** are the same (**perfect induction**)
  - There exists a sequence of algebraic manipulations to transform one logic expression to the other
    - These algebraic manipulations are defined as **Boolean algebra**
  - Their **Venn diagrams** are the same

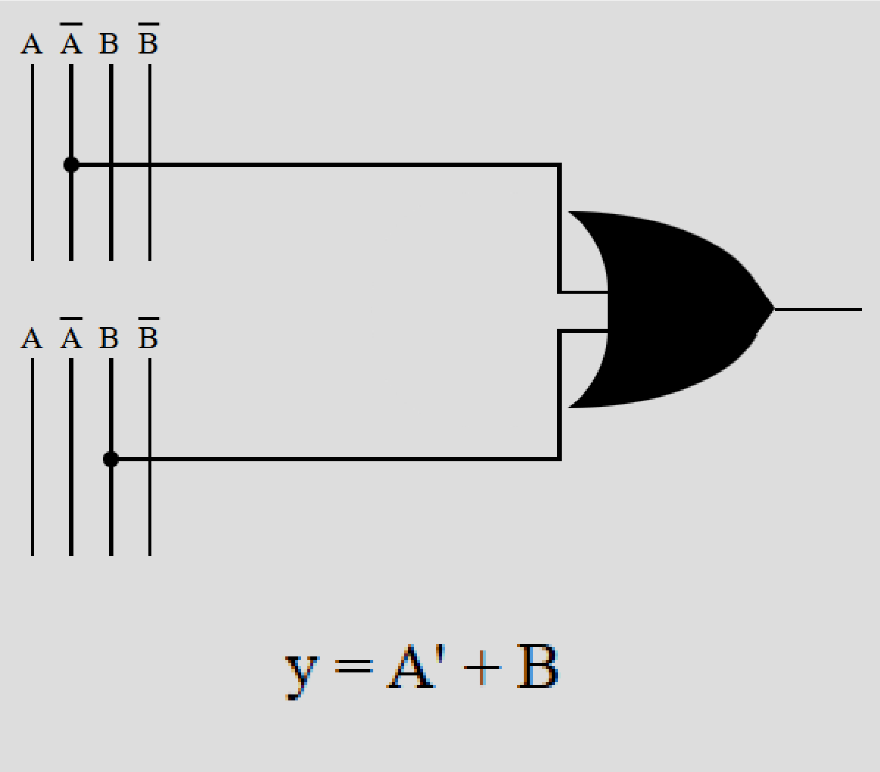
How To Find The Best Equivalent Network?

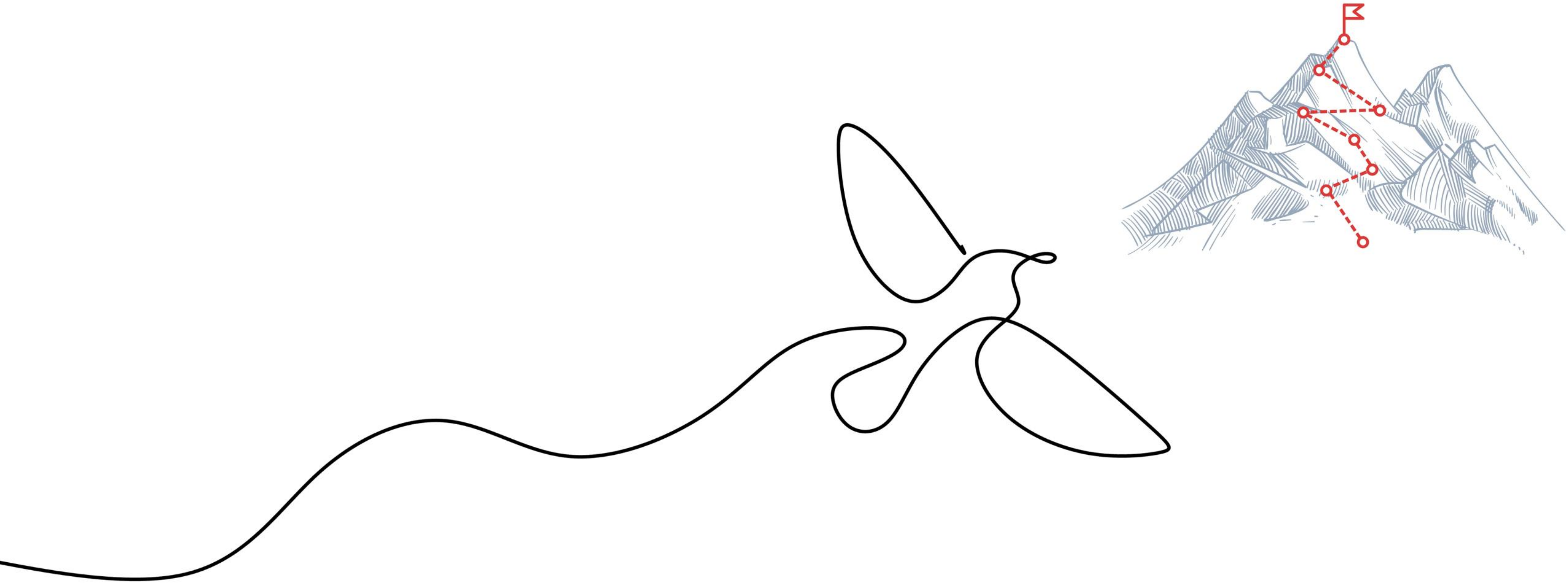
- Logic function can be implemented with a variety of different networks. How does one find the best (simplest, least costly)?
- The process of finding the best equivalent logical expression describing a logic network is called **minimization**
  - **Approach 1**: Apply a sequence of algebraic transformation
    - Not always obvious when to apply which transformation, tedious, impractical
  - **Approach 2**: Use Karnaugh maps (an alternative to the truth table)
    - Simpler, but quickly becomes unmanageable by hand (up to 4 inputs acceptable)
  - 🏆 **Approach 3**: Automated techniques in synthesis software tools

Logic Circuit Simplification

- Online resource on logic circuit simplification using Karnaugh maps for enthusiasts: http://www.32x8.com/index.html

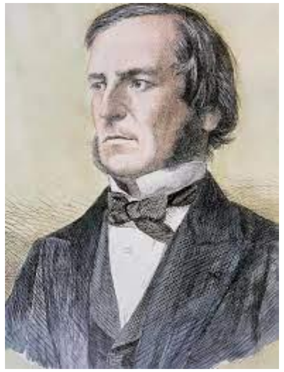






Boolean Algebra

A Bit of History



- In 1849, George Boole published a scheme for the algebraic description of processes involved in logical thought and reasoning
- This scheme and its refinements became known as Boolean algebra

- In the late 1930s, Claude Shannon showed that Boolean algebra provides an effective means of describing circuits built with switches
  - Therefore, Boolean algebra can be used to describe logic circuits
- Boolean algebra is a powerful technique for designing and analyzing logic circuits; it is the foundation for our modern digital technology

Axioms

Boolean Algebra

- Like any algebra, Boolean algebra is based on a set of rules derived from a small number of basic assumptions (i.e., **axioms**)
- Let us assume that Boolean algebra involves elements that take on one of the two binary values. Assume the following axioms are true:

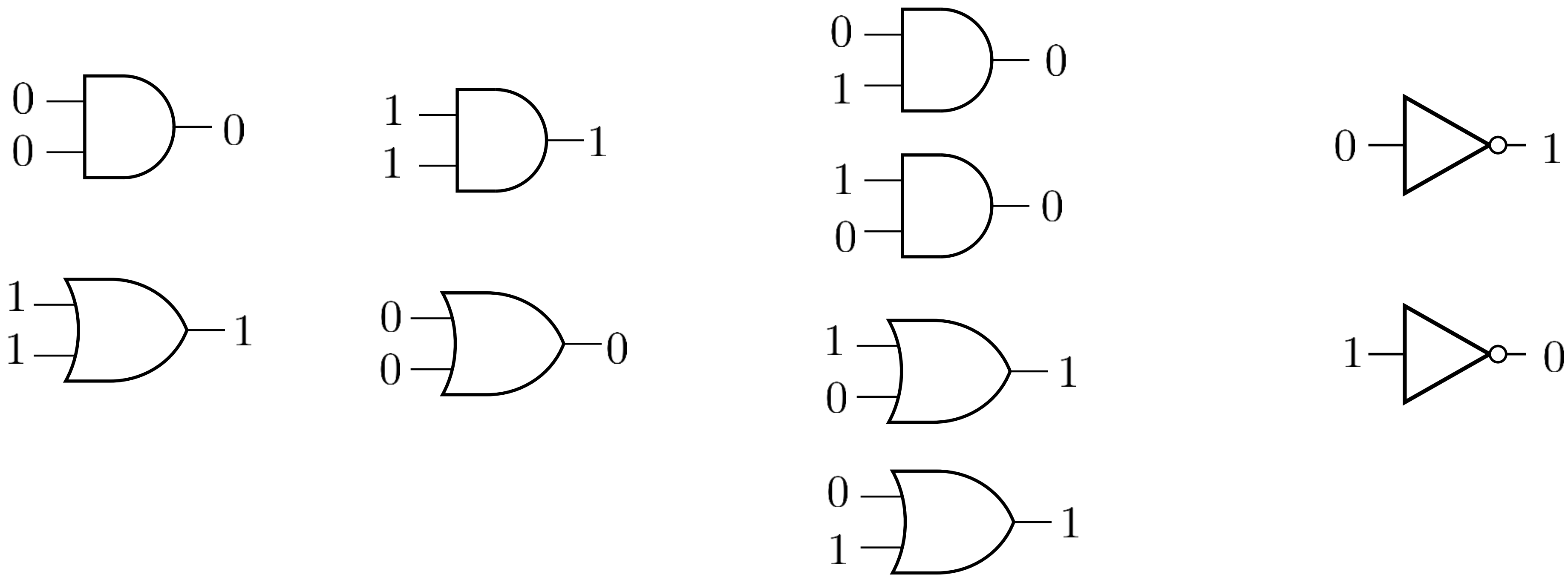
$1a.\ \ 0 \cdot 0 = 0$	$2a.\ \ 1 \cdot 1 = 1$	$3a.\ \ 0 \cdot 1 = 1 \cdot 0 = 0$	$4a.\ \ \text{If } x = 0,\ \text{then } \bar{x} = 1$
$1b.\ \ 1 + 1 = 1$	$2b.\ \ 0 + 0 = 0$	$3b.\ \ 1 + 0 = 0 + 1 = 1$	$4b.\ \ \text{If } x = 1,\ \text{then } \bar{x} = 0$

- From the axioms, we can define some rules (i.e., **theorems**) for dealing with single Boolean variables

Axioms

Analogy with Logic Gates

1	2	3	4
$1a.\quad 0 \cdot 0 = 0$	$2a.\quad 1 \cdot 1 = 1$	$3a.\quad 0 \cdot 1 = 1 \cdot 0 = 0$	$4a.\quad \text{If } x = 0, \text{ then } \bar{x} = 1$
$1b.\quad 1 + 1 = 1$	$2b.\quad 0 + 0 = 0$	$3b.\quad 1 + 0 = 0 + 1 = 1$	$4b.\quad \text{If } x = 1, \text{ then } \bar{x} = 0$



Single-Variable Theorems

Boolean Algebra

- If x is a Boolean variable, then the following theorems hold:

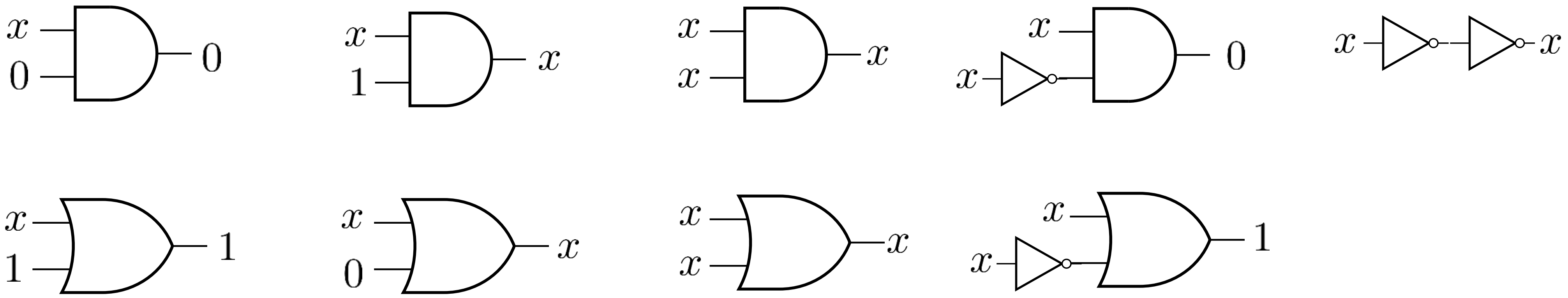
$5a.\quad x \cdot 0 = 0$	$6a.\quad x \cdot 1 = x$	$7a.\quad x \cdot x = x$	$8a.\quad x \cdot \bar{x} = 0$	$9.\quad \bar{\bar{x}} = x$
$5b.\quad x + 1 = 1$	$6b.\quad x + 0 = x$	$7b.\quad x + x = x$	$8b.\quad x + \bar{x} = 1$	

- Theorems grouped in pairs, emphasizing **the principle of duality**
  - **Dual form** is obtained by replacing all + operators with · operators, and vice versa; and by replacing all 0s with 1s, and vice versa
- To prove the theorems, apply **perfect induction** (i.e., substitute the variable with 1 or 0) and use the axioms

Single-Variable Theorems

Analogy with Logic Gates

$5a.\quad x \cdot 0 = 0$ $5b.\quad x + 1 = 1$	$6a.\quad x \cdot 1 = x$ $6b.\quad x + 0 = x$	$7a.\quad x \cdot x = x$ $7b.\quad x + x = x$	$8a.\quad x \cdot \bar{x} = 0$ $8b.\quad x + \bar{x} = 1$	$9.\quad \bar{\bar{x}} = x$



Single-Variable Theorem Proof

Using Perfect induction

EXAMPLES

- Let us prove the validity of theorem $5a.\quad x \cdot 0 = 0$
- Perfect induction:
  - $x = 0$: the theorem states $0 \cdot 0 = 0$
    This is true according to axiom 1a.

    $1a.\quad 0 \cdot 0 = 0$
    $1b.\quad 1 + 1 = 1$

  - $x = 1$: the theorem states $1 \cdot 0 = 0$
    This is true according to axiom 3a.

    $3a.\quad 0 \cdot 1 = 1 \cdot 0 = 0$
    $3b.\quad 1 + 0 = 0 + 1 = 1$

Recall the truth table

AND

	x_1	x_2	$x_1 \cdot x_2$
$x_1 \cdot 0$	0	0	0
	0	1	0
$x_1 \cdot 0$	1	0	0
	1	1	**1**

Two- and Three-Variable Properties
Boolean Algebra

- Given three Boolean variables, the following properties hold

Commutative

$10a.\quad x \cdot y = y \cdot x$

$10b.\quad x + y = y + x$

Associative

$11a.\quad x \cdot (y \cdot z) = (x \cdot y) \cdot z$

$11b.\quad x + (y + z) = (x + y) + z$

Distributive

$12a.\quad x \cdot (y + z) = x \cdot y + x \cdot z$

$12b.\quad x + y \cdot z = (x + y) \cdot (x + z)$

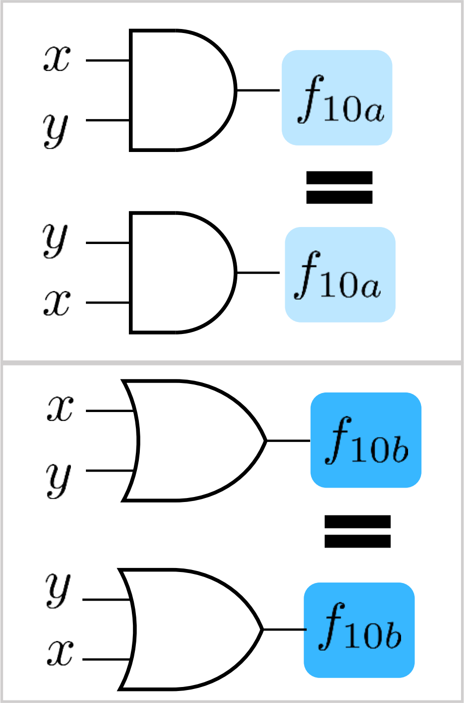
Two- and Three-Variable Properties

Analogy with Logic Gates

Commutative

10a. $x \cdot y = y \cdot x$

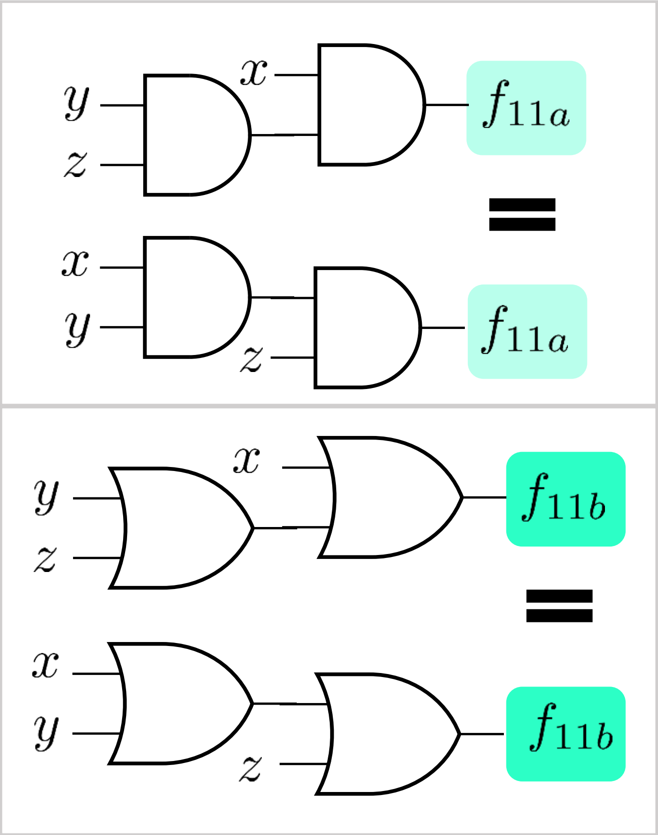
10b. $x + y = y + x$



Associative

11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

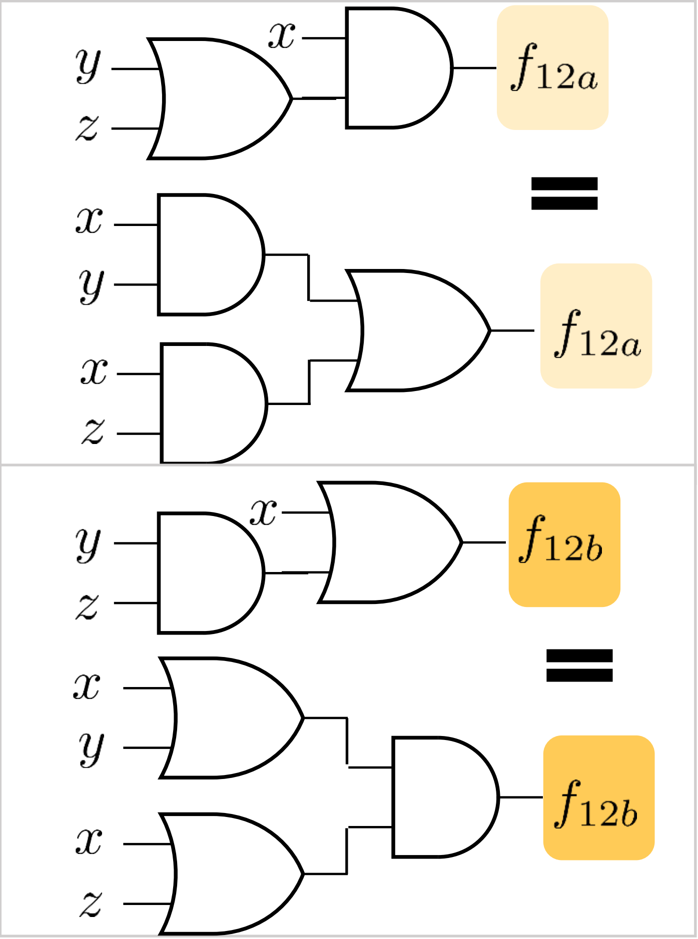
11b. $x + (y + z) = (x + y) + z$



Distributive

12a. $x \cdot (y + z) = x \cdot y + x \cdot z$

12b. $x + y \cdot z = (x + y) \cdot (x + z)$



Checking the Validity of a Logic Equation

Using Boolean Algebra

- Let us prove the validity of the following logic equation

$$(x_1 + x_3)(\overline{x_1} + \overline{x_3}) = x_1\overline{x_3} + \overline{x_1}x_3$$

Distributive

$$12a.\ x \cdot (y + z) = x \cdot y + x \cdot z$$
$$12b.\ x + y \cdot z = (x + y) \cdot (x + z)$$

$$8a.\quad x \cdot \bar{x} = 0$$
$$8b.\quad x + \bar{x} = 1$$

$$6a.\quad x \cdot 1 = x$$
$$6b.\quad x + 0 = x$$

Commutative

$$10a.\quad x \cdot y = y \cdot x$$
$$10b.\quad x + y = y + x$$

EXAMPLES

Checking the Validity of a Logic Equation

Without truth tables and Venn diagrams

- Let us prove the validity of the following logic equation

$$(x_1 + x_3)(\overline{x_1} + \overline{x_3}) = x_1\overline{x_3} + \overline{x_1}x_3$$

- Let us manipulate the left-hand side (LHS)

$$\begin{aligned}
\text{LHS} &= (x_1 + x_3)(\overline{x_1} + \overline{x_3}) \\
\textbf{(12a)} \quad &= (x_1 + x_3)\overline{x_1} + (x_1 + x_3)\overline{x_3} \\
\textbf{(12a)} \quad &= x_1\overline{x_1} + x_3\overline{x_1} + x_1\overline{x_3} + x_3\overline{x_3} \\
\textbf{(8a)} \quad &= 0 + x_3\overline{x_1} + x_1\overline{x_3} + 0 \\
\textbf{(6b)} \quad &= x_3\overline{x_1} + x_1\overline{x_3} \\
\textbf{(10a, 10b)} \quad &= x_1\overline{x_3} + \overline{x_1}x_3
\end{aligned}$$

- The result is exactly the right-hand side (RHS)

Distributive

$12a.\ x \cdot (y + z) = x \cdot y + x \cdot z$

$12b.\ x + y \cdot z = (x + y) \cdot (x + z)$

$8a.\ \ x \cdot \bar{x} = 0$

$8b.\ \ x + \bar{x} = 1$

$6a.\ \ x \cdot 1 = x$

$6b.\ \ x + 0 = x$

Commutative

$10a.\ \ x \cdot y = y \cdot x$

$10b.\ \ x + y = y + x$

What's the Point…?

… of Axioms, Theorems, Properties

- A: The purpose of the axioms, theorems, and properties in Boolean Algebra is to perform algebraic transformations to do
  - **Check for equivalence**
    - Find if two logical expressions (i.e., logical circuits made of gates) are equivalent (i.e., perform the same functionality) without evaluating all input possibilities
  - **Design efficient circuits**
    - Simplify the logical expression to find a potentially more efficient equivalent variant (i.e., design a circuit of the same desired functionality but with fewer gates)

Two- and Three-Variable Properties, Contd.

Boolean Algebra

- Given three Boolean variables, the following properties hold

Absorption (covering)

$$13a.\quad x + x \cdot y = x$$
$$13b.\quad x \cdot (x + y) = x$$

Combining

$$14a.\quad x \cdot y + x \cdot \bar{y} = x$$
$$14b.\quad (x + y) \cdot (x + \bar{y}) = x$$

DeMorgan's theorem

$$15a.\quad \overline{x \cdot y} = \overline{x} + \overline{y}$$
$$15b.\quad \overline{x + y} = \overline{x} \cdot \overline{y}$$

Redundancy

$$16a.\quad x + \bar{x} \cdot y = x + y$$
$$16b.\quad x \cdot (\bar{x} + y) = x \cdot y$$

Consensus

$$17a.\quad x \cdot y + y \cdot z + \bar{x} \cdot z = x \cdot y + \bar{x} \cdot z$$
$$17b.\quad (x + y) \cdot (y + z) \cdot (\bar{x} + z) = (x + y) \cdot (\bar{x} + z)$$

Checking the Validity of a Logic Equation

Without truth tables and Venn diagrams

- Prove the validity of the following logic equation

$$x_1\overline{x_3} + \overline{x_2}\,\overline{x_3} + x_1x_3 + \overline{x_2}x_3 = \overline{x_1}\,\overline{x_2} + x_1x_2 + x_1\overline{x_2}$$

- The **left-hand** side manipulation

$$\begin{aligned}
\text{LHS} &= x_1\overline{x_3} + \overline{x_2}\,\overline{x_3} + x_1x_3 + \overline{x_2}x_3 \\
\text{(10b)} &= x_1\overline{x_3} + x_1x_3 + \overline{x_2}\,\overline{x_3} + \overline{x_2}x_3 \\
\text{(12a)} &= x_1(\overline{x_3} + x_3) + \overline{x_2}(\overline{x_3} + x_3) \\
\text{(8b)} &= x_1 \cdot 1 + \overline{x_2} \cdot 1 \\
\text{(6a)} &= x_1 + \overline{x_2}
\end{aligned}$$

Commutative

$10a.\quad x \cdot y = y \cdot x$

$10b.\quad x + y = y + x$

Distributive

$12a.\quad x \cdot (y + z) = x \cdot y + x \cdot z$

$12b.\quad x + y \cdot z = (x + y) \cdot (x + z)$

$8a.\quad x \cdot \bar{x} = 0$

$8b.\quad x + \bar{x} = 1$

$6a.\quad x \cdot 1 = x$

$6b.\quad x + 0 = x$

EXAMPLES

Checking the Validity of a Logic Equation

Without truth tables and Venn diagrams

- Prove the validity of the following logic equation

$$x_1\overline{x_3} + \overline{x_2}\,\overline{x_3} + x_1x_3 + \overline{x_2}x_3 = \overline{x_1}\,\overline{x_2} + x_1x_2 + x_1\overline{x_2}$$

- The **right-hand** side manipulation

$$\begin{aligned} \text{RHS} &= \overline{x_1}\,\overline{x_2} + x_1x_2 + x_1\overline{x_2} \\ \text{(12a)} &= \overline{x_1}\,\overline{x_2} + x_1(x_2 + \overline{x_2}) \\ \text{(8b)} &= \overline{x_1}\,\overline{x_2} + x_1 \cdot 1 \\ \text{(6a)} &= \overline{x_1}\,\overline{x_2} + x_1 \\ \text{(10b)} &= x_1 + \overline{x_1}\,\overline{x_2} \\ \text{(16a)} &= x_1 + \overline{x_2} \end{aligned}$$

Commutative

$10a.\quad x \cdot y = y \cdot x$

$10b.\quad x + y = y + x$

Distributive

$12a.\quad x \cdot (y + z) = x \cdot y + x \cdot z$

$12b.\quad x + y \cdot z = (x + y) \cdot (x + z)$

$8a.\quad x \cdot \bar{x} = 0$

$8b.\quad x + \bar{x} = 1$

Redundancy

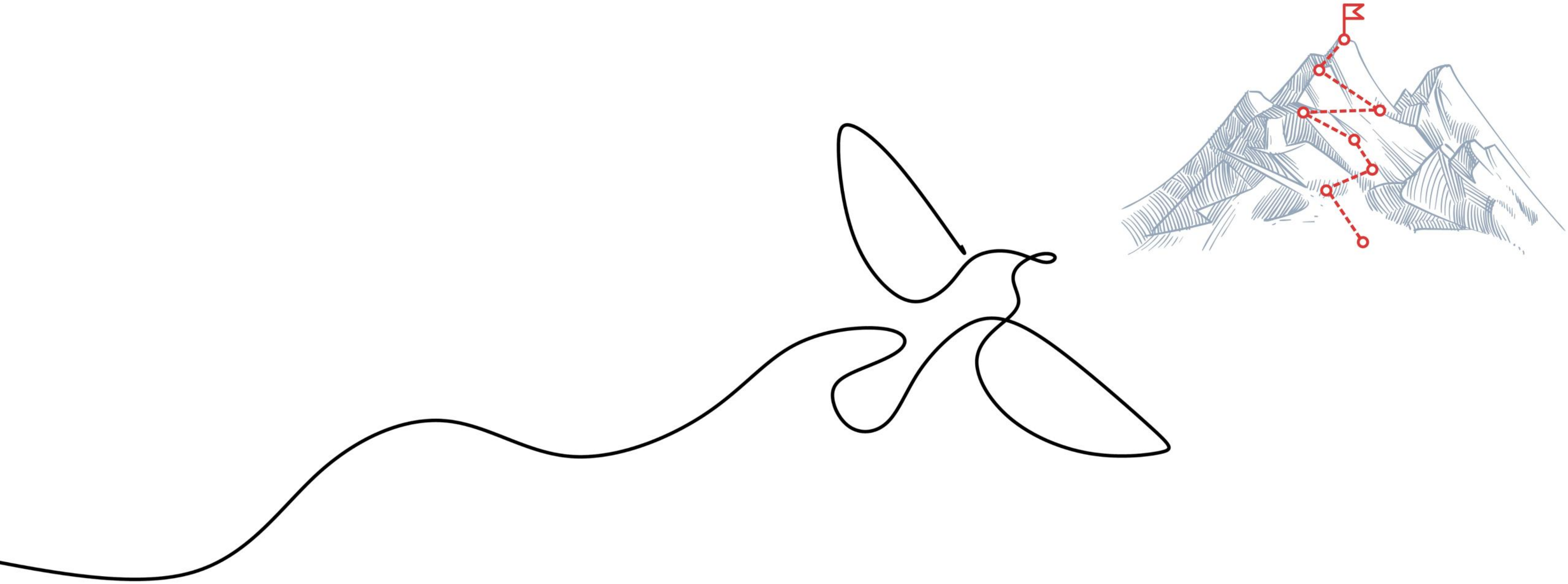
$16a.\quad x + \bar{x} \cdot y = x + y$

$16b.\quad x \cdot (\bar{x} + y) = x \cdot y$

$6a.\quad x \cdot 1 = x$

$6b.\quad x + 0 = x$

EXAMPLES

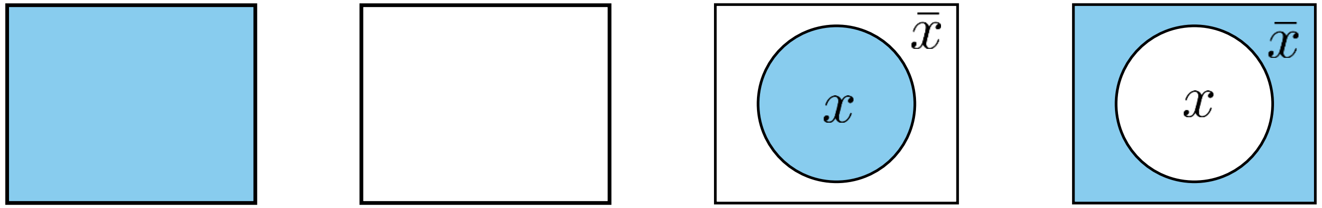


The Venn Diagram

Two networks are equivalent if their Venn diagrams are the same

The Venn Diagram

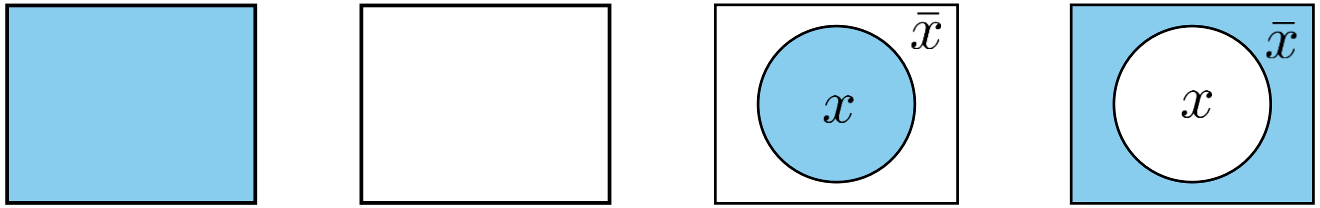
- Provides a graphical illustration of various operations and relations in the algebra of sets
- Popularized by John Venn (1834–1923) in the 1880s



The Venn Diagram

Shades and Contours

- In the diagram, the elements of a set are represented by the area enclosed by a **contour of a circle**
  - Shaded area where the **logical function** value = binary 1
  - The area within the contour: **variable** value = binary 1
  - The area outside the contour: **variable** value = binary 0



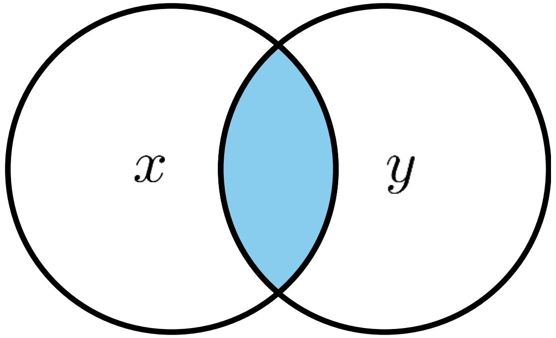
Constant 1 | Constant 0 | $f(x) = x$ | $f(x) = \bar{x}$

- The **union** of the shaded areas corresponds to the logical expression

The Venn Diagram

Simple Intersection

- *Reminder:* The union of the shaded areas corresponds to the logical expression (shaded when the expression is binary 1)
- Q: What is the corresponding logical expression?



xy

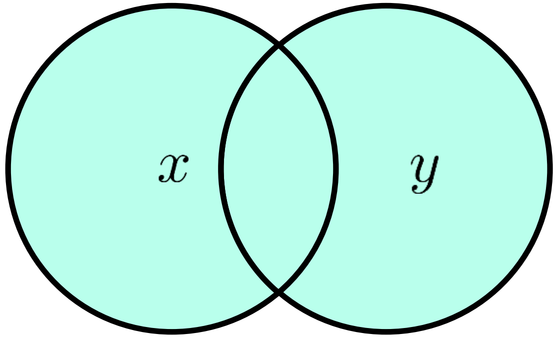
- A: **AND** (intersection; both variables = 1)

EXAMPLES

The Venn Diagram

Simple Union

- *Reminder:* The union of the shaded areas corresponds to the logical expression (shaded when the expression is binary 1)
- Q: What is the corresponding logical expression?



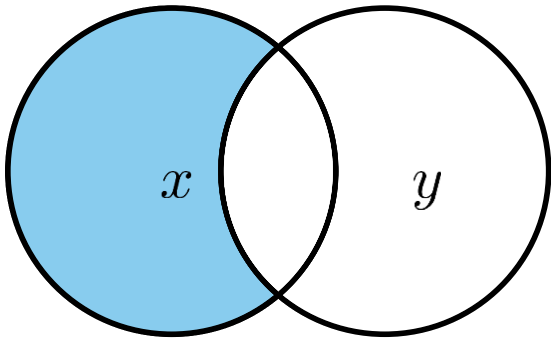
$x + y$

- A: **OR** (union; either variable = 1)

The Venn Diagram

Flipped Task: Draw It

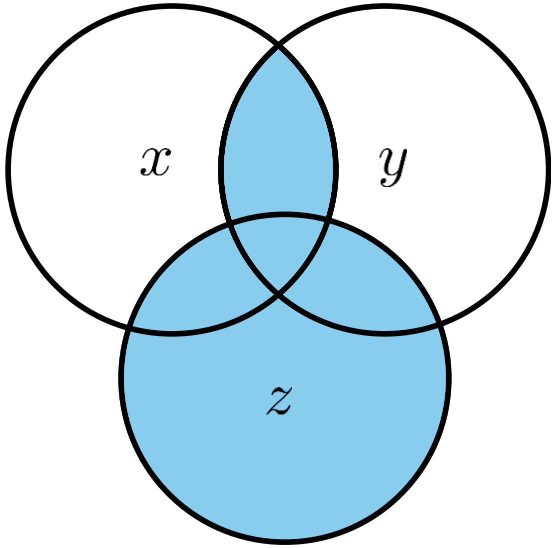
- *Reminder:* The union of the shaded areas corresponds to the logical expression (shaded when the expression is binary 1)
- Q: Show $x\bar{y}$, i.e., the intersection of
  - The region x = 1 and
  - The region y = 0
- A:



$x\bar{y}$

The Venn Diagram, Contd.

- *Reminder:* The union of the shaded areas corresponds to the logical expression (shaded when the expression is binary 1)
- Q: Show $xy + z$, i.e., the union of
  - Intersection x = 1, y = 1 and
  - The region z = 1
- A:

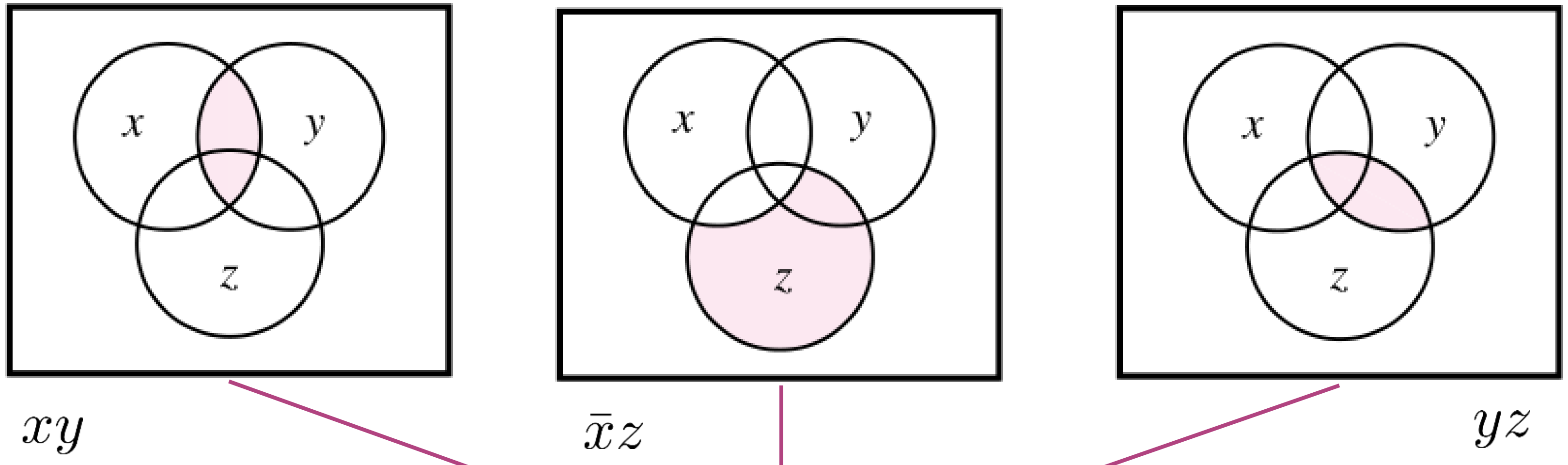


$xy + z$

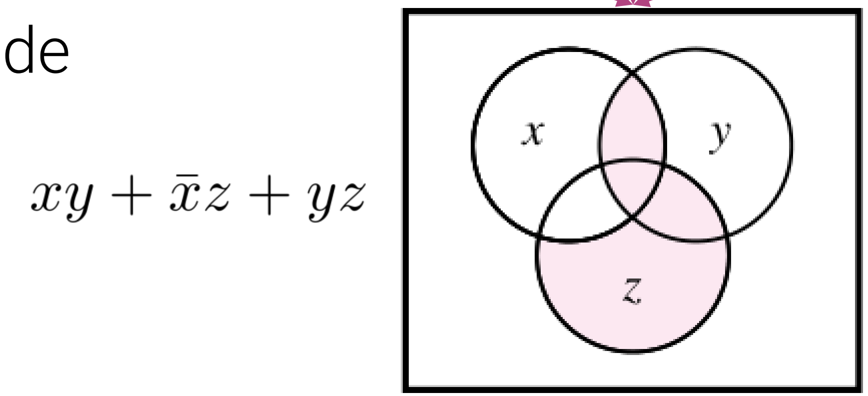
Network Equivalence Verification

Venn Diagram Approach

$$xy + \bar{x}z + yz \stackrel{?}{=} xy + \bar{x}z$$



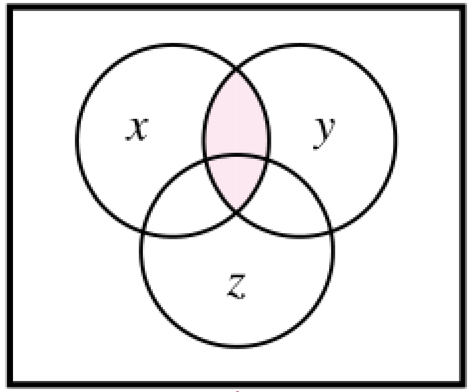
- Left-hand side



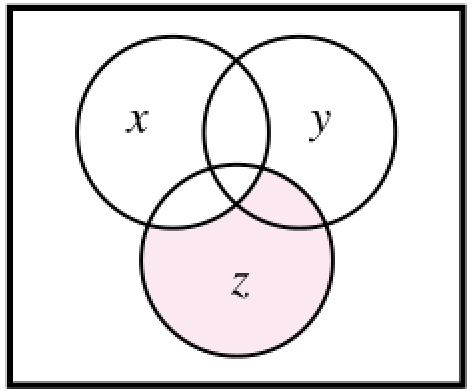
Network Equivalence Verification

Venn Diagram Approach

$xy + \bar{x}z + yz \overset{?}{=} xy + \bar{x}z$



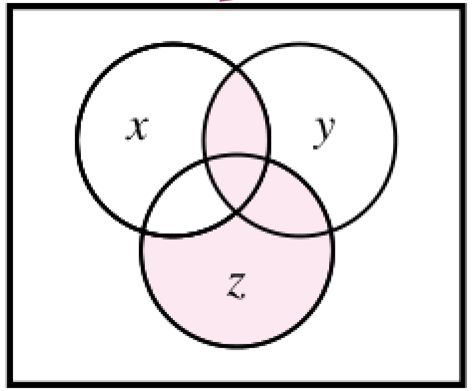
xy



$\bar{x}z$

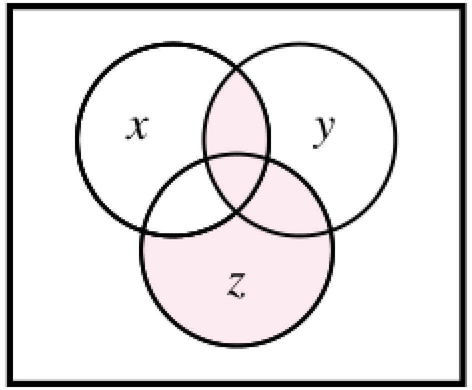
- Right-hand side

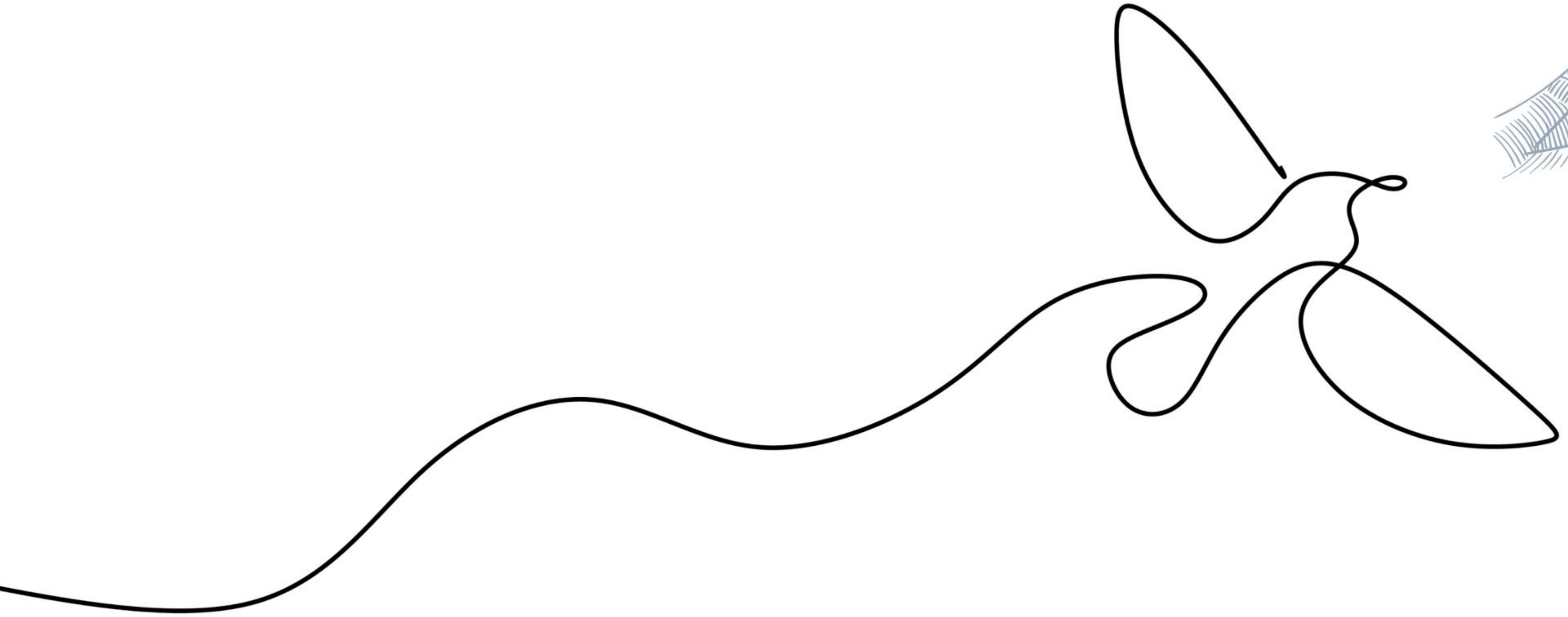
$xy + \bar{x}z$

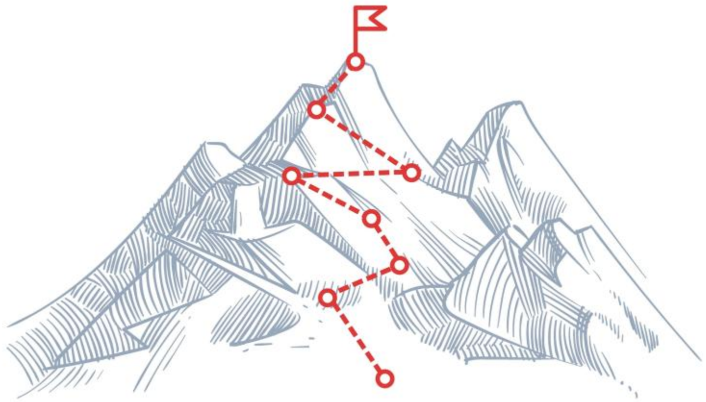


$\equiv$

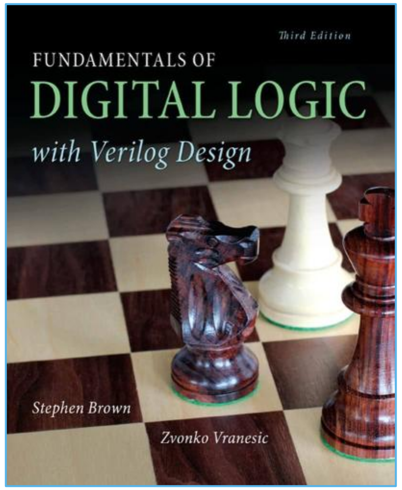
$xy + \bar{x}z + yz$



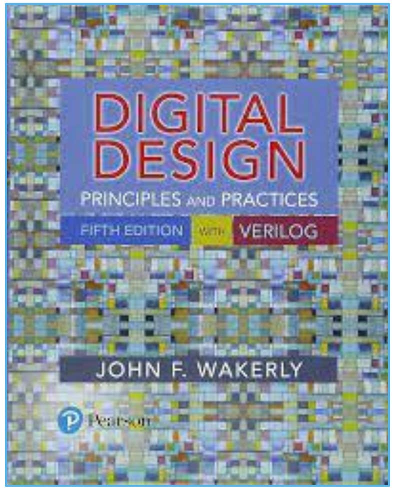




Literature



- Chapter 2: Introduction to Logic Circuits
  - 2.1-2.5



- Chapter 1: Introduction
  - 1.5
- Chapter 3: Switching Algebra and Combinational Logic
  - 3.1.1-3.1.3