Fundamentals of Computing

Michael Zakharyaschev

Department of Computer Science and Information Systems

Birkbeck University of London

- room 265, Main Building, Malet Street
- email: `michael@dcs.bbk.ac.uk`
- homepage: `http://www.dcs.bbk.ac.uk/~michael`
- FoC module site: `http://www.dcs.bbk.ac.uk/~michael/foc/foc.html`

(can be accessed via Moodle)

Learning the material

- You are not supposed to understand **everything** during the lectures. The lecture material is prepared to guide you through the chosen topics.

- The **only** way to master maths material is by **doing it yourself**

- In the tutorials, you are expected to work through the provided exercises **yourself**. The exercises are carefully chosen to help you understand the introduced concepts and techniques. It is often helpful to go through the lecture slides once more in your attempt of solving the exercises. The tutor will help you if you have any difficulties, and will also discuss your solutions (but he won't solve the exercises instead of you).

  Model solutions will be published right after the tutorial.

- More exercises are provided on the module site (scroll down the page). If you think you need even more, please consult the recommended textbooks. And finally: please **use the Web** (there is an extensive amount of online material, including videos, on the covered topics).

What is this module for?

- This module aims to (i) provide a common mathematical background for many of your other modules, and to (ii) bring the **mathematical** skills of the class to the same level.

- The module also aims to develop your problem solving skills and your ability to express yourself in a more precise manner.

- If you had Mathematics A-level, you may find that some parts of the module cover material you already know to some extent (and you might be bored:-)

  But watch out: there will be something new any time

- If you did not have Mathematics A-level: **don't panic!**
  You are not expected to know topics that you have never seen before, everything is developed from scratch.

  And much of Maths A-level is not relevant to this module anyway (for example, we won't use differentiation or trigonometry).

What is used?

- **integer numbers**

$$\overbrace{\ldots,\ -5,\ -4,\ -3,\ -2,\ -1}^{\text{negative}},\ \underbrace{0,\ \overbrace{1,\ 2,\ 3,\ 4,\ 5,\ \ \ldots}^{\text{positive}}}_{\text{natural numbers}}$$

  - odd numbers, even numbers, divisibility
  - operations on integers: $+,\ -,\ \times,\ /$
  - exponentiation: $5^6,\ (-2)^0,\ 6^3,\ \ldots$
  - comparing integers using binary relations $=,\ >,\ <,\ \geq,\ \leq$
  - denoting integers with letters $(n, m, x, y, \ldots)$

    $\rightsquigarrow$ understanding statements like "$2^n > n^2$ *for every integer* n".

- a bit of **algebra:** $(a+b)^2 = a^2 + 2ab + b^2,\quad a^2 - b^2 = (a-b)(a+b),\quad \ldots$
- **working with fractions**

Please refresh your knowledge of these !

Syllabus: autumn term

Part I: Computer logic and arithmetic

- How are numbers represented in computers?
- How are negative numbers represented?
- What is the largest number that can be represented in a computer word?
- How does hardware really add, subtract, multiply, or divide numbers?

M.M. Mano and C.R. Kime. *Logic and Computer Design Fundamentals*. 4th Ed. Pearson, 2008

also: S.S. Epp. Discrete Mathematics with Applications, Sections 2.4–2.5

Part II: Models of computation

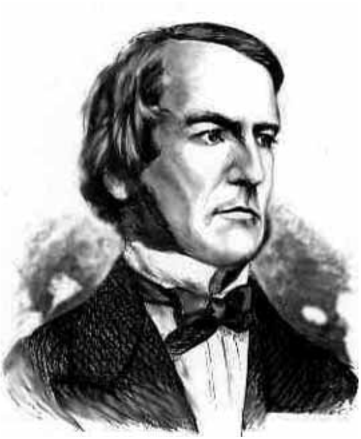
- What is a computation or an algorithm?
- What can and what cannot be computed?
- What can be computed with limited memory?
- What makes some problems computationally hard and others easy?

E. Kinber and C. Smith. *Theory of Computing. A gentle introduction.* Prentice Hall, 2001

also: S.S. Epp. Discrete Mathematics with Applications, Chapter 12

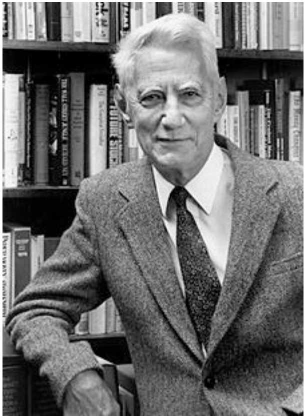
Part I:

Computer logic and arithmetic



George Boole

(1815–1864)



Claude Shannon

(1916–2001)

Why are computers binary?

- Simple and cheap: easy to build.
- Unambiguous signals (hence noise immunity).
- Flawless copies can be made.
- 0 and 1 are enough to encode anything we need.
- Binary arithmetic is **more efficient** than decimal.
- . . . (there also exist ternary computers)

- **George Boole** invented Boolean algebra, an algebra of two values,

'The Laws of Thought' 1854 which is the basis for all modern computer arithmetic

- **Claude Shannon** showed how electrical application of Boolean algebra could construct and resolve any logical, numerical relationship (also founded information theory)

Logic

Logic is the formal systematic study of the principles of **valid inference** and **correct reasoning**

Are the following inferences valid?

- If it is raining then I take an umbrella
- It is raining
- Therefore I take an umbrella

- If it is raining then I take an umbrella
- It is not raining
- Therefore I don't take an umbrella

What does it mean for one sentence to **follow logically** from certain others?

The sentences above are **declarative sentences**, or **propositions**, which one can, in principle, argue as being **true** or **false**

Boolean algebra (or **Boolean logic**) is a logical calculus of truth values, developed by George Boole in the 1840s

Elements of Boolean logic

Basic assumption: every **proposition** is either **true** or **false** (but not both)

Examples:

(A) George W. Bush is the current president of the United States of America.

(B) Donald Trump is the current president of the United States of America.

(C) Extraterrestrial life does not exist.

NB: Questions/exclamations, paradoxical statements like 'this proposition is false' are **not** propositions.

Propositional connectives:

- 'not' (negation) denoted by $\neg$ (! in C++/Java) Is $\neg A$ true?
- 'and' (conjunction) denoted by $\wedge$ (&& in C++/Java) Is $A \wedge B$ true?
- 'or' (disjunction) denoted by $\vee$ (|| in C++/Java) Is $A \vee B$ true?
- 'if ... then ...' (implication) denoted by $\to$ Is $A \to C$ true?

? Are there any other propositional connectives?

Complex propositions (formulas): $(\neg A) \to (B \vee C)$, $((\neg B) \wedge (\neg\neg C)) \to \neg A$, etc.

Semantics: truth-tables

Notation: $\mathbf{1}$ for 'true', $\mathbf{0}$ for 'false'

$A, B, C, A_1, B_1, \dots$ for atomic (in a given context) propositions
a.k.a. **propositional variables**

$A \vee B, \ (\neg A) \to (A_1 \wedge \neg B_2), \dots$ for complex propositions
a.k.a. **propositional** or **Boolean formulas**

Truth-tables for $\wedge$, $\vee$, $\to$ and $\neg$:

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$\neg A$
0	0	0	0	1	1
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	1	0

so the proposition 'if the Moon is made of green cheese, then $2 \times 2 = 7$' is true

Explaining ‘implication’

One possible ‘explanation’ of the truth-table for the ‘implication’ $\to$:

The following statement is true for **every** natural number n:

If n is divisible by 4, then n is divisible by 2.

So the following instances of this general statement must be true as well:

If 8 is divisible by 4, then 8 is divisible by 2	$(1 \to 1) = 1$
If 7 is divisible by 4, then 7 is divisible by 2	$(0 \to 0) = 1$
If 2 is divisible by 4, then 2 is divisible by 2	$(0 \to 1) = 1$

And of course, ‘if 8 is divisible by 4, then 7 is divisible by 2’ is false $\quad (1 \to 0) = 0$

(also: analyse the wrong inference on page 8)

This interpretation of logical connectives is a **mathematical abstraction**. Under such abstractions, meaningless sentences may become sensible, and the other way round.

There are different interpretations of logic connectives, e.g., with three or more truth-values.

Truth-tables for Boolean formulas

A	B	$(\neg$	$A)$	$\vee$	B
0	0	1	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	1	0	1	1	1

Note that this truth-table (the column under the main connective $\vee$) is the same as the truth-table for $A \to B$

So we can say that $(\neg A) \vee B$ is **equivalent to** $A \to B$

Exercise Brown, Jones and Smith are suspected of income tax evasion. They testify under oath as follows:

- Brown: Jones is guilty and Smith is innocent.
- Jones: If Brown is guilty, then so is Smith.
- Smith: I'm innocent, but at least one of the others is guilty.

Assuming everyone's testimony is true, who is innocent and who is guilty?

Solution

BG stands for 'Brown is guilty', JG for 'Jones is guilty' and SG for 'Smith is guilty'

- Brown says: $JG \wedge \neg SG$ Jones says: $BG \to SG$ Smith says: $\neg SG \wedge (BG \vee JG)$

BG	JG	SG	$JG \wedge \neg SG$	$BG \to SG$	$\neg SG \wedge (BG \vee JG)$
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	**1**	**1**	**1**
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	1	0

The only case where all the statements are true: $BG = 0$, $JG = 1$ and $SG = 0$

This problem can be solved in a more direct way. From the first statement we can infer that Jones is guilty and Smith is innocent. From the second statement, it follows that if Smith is not guilty then Brown is not guilty. Therefore we can infer that Brown is innocent.

Formalising English sentences

Exercise: Translate the following English sentences to propositional logic:

(1) *If I am not playing tennis, I am watching tennis.*

(2) *If I am not watching tennis, I am reading about tennis.*

(3) *I cannot do more than one of these activities at the same time.*

Solution: First we choose our propositional variables.

(NB: they must denote propositions! say $\boldsymbol{T}$: `*playing tennis*' is no good)

$\boldsymbol{T}$: `*I am playing tennis*'

$\boldsymbol{W}$: `*I am watching tennis*'

$\boldsymbol{R}$: `*I am reading about tennis*'

Then we can formalise the above English sentences as follows:

(1) $\neg \boldsymbol{T} \to \boldsymbol{W}$

(2) $\neg \boldsymbol{W} \to \boldsymbol{R}$

(3) $\neg(\boldsymbol{T} \wedge \boldsymbol{W}) \wedge \neg(\boldsymbol{T} \wedge \boldsymbol{R}) \wedge \neg(\boldsymbol{W} \wedge \boldsymbol{R})$

or $(\boldsymbol{T} \wedge \neg \boldsymbol{W} \wedge \neg \boldsymbol{R}) \vee (\neg \boldsymbol{T} \wedge \boldsymbol{W} \wedge \neg \boldsymbol{R}) \vee (\neg \boldsymbol{T} \wedge \neg \boldsymbol{W} \wedge \boldsymbol{R}) \vee (\neg \boldsymbol{T} \wedge \neg \boldsymbol{W} \wedge \neg \boldsymbol{R})$

Logically correct arguments in propositional logic

An **argument** is a sequence of propositions:

$$\left.\begin{array}{l} p_1 \\ p_2 \\ \vdots \\ p_n \end{array}\right\} \; n \text{ premises (aka assumptions)}$$

$$\text{Therefore } \frac{\qquad}{q} \quad \text{one conclusion}$$

An argument is **logically correct** if

in every 'situation' that makes all the premises true, the conclusion is true as well

a **situation** = a row in the truth table for $p_1, \dots, p_n, q$

= a possible assignment for the propositional variables in $p_1, \dots, p_n, q$

$\boxed{\begin{array}{l} p_1 \\ p_2 \\ \dots \\ p_n \\ \hline q \end{array}}$ is logically correct iff the formula $(p_1 \wedge p_2 \wedge \dots \wedge p_n) \to q$ is always true

in which case it is called a **tautology**

Logically correct argument: an example

If I am not playing tennis, I am watching tennis	$\neg T \to W$	
If I am not watching tennis, I am reading about tennis	$\neg W \to R$	$\mathbf{3}$ premises
I cannot do more than one of these activities at the same time	$\neg(T \land W) \land \neg(T \land R) \land \neg(W \land R)$	
Therefore,		
I am watching tennis	W	conclusion

In every situation that makes all the premises true, the conclusion is true as well:

T	W	R	$\neg T \to W$	$\neg W \to R$	$\neg(T \land W) \land \neg(T \land R) \land \neg(W \land R)$	W
1	1	1	1	1	0	1
1	1	0	1	1	0	1
1	0	1	1	1	0	0
1	0	0	1	0	1	0
0	1	1	1	1	0	1
0	1	0	**1**	**1**	**1**	**1**
0	0	1	0	1	1	0
0	0	0	0	0	1	0
			premises			conclusion

Incorrect argument: a simple example

If you solve every exercise in the textbook, then you will get an A.	$S \to A$
You did not solve every exercise in the textbook.	$\neg S$

Therefore

You won't get an A. $\neg A$

Incorrect argument: It is not the case that
'in every situation that makes all the premises true, the conclusion is true as well.'

So, to show that an argument is incorrect, it is **enough to find one situation** where

- all premises are true,
- but the conclusion is false.

S	A	$S \to A$	$\neg S$	$\neg A$
1	1	1	0	0
1	0	0	0	1
0	1	**1**	**1**	**0**
0	0	1	1	1

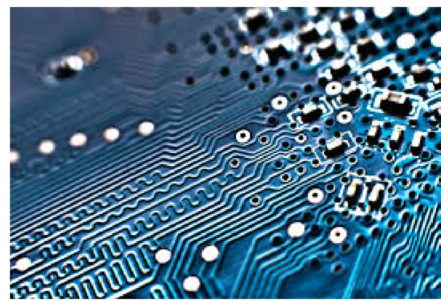
Boolean logic in computers

In the world of computers, **0** and **1** are known as **bits**

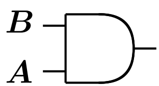
- 0 is represented by the lower voltage level (LOW), say, 0V – 0.1V
- 1 is represented by the higher voltage level (HIGH), say, 0.9V – 1.1V

All computer circuits consist of hundreds of millions of interconnected primitive elements called **gates**, which correspond to the basic logic connectives:

Basic logic gates:

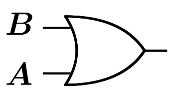


AND gate



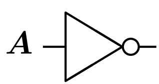
$C = A \wedge B$

OR gate



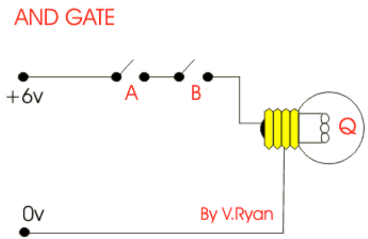
$C = A \vee B$

NOT gate

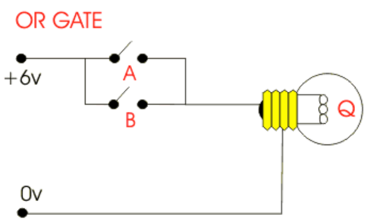


$C = \neg A$

Examples



The AND gate has two inputs, switch A and switch B. The bulb Q will only light if both switches are closed. This will allow current to flow through the bulb, illuminating the filament



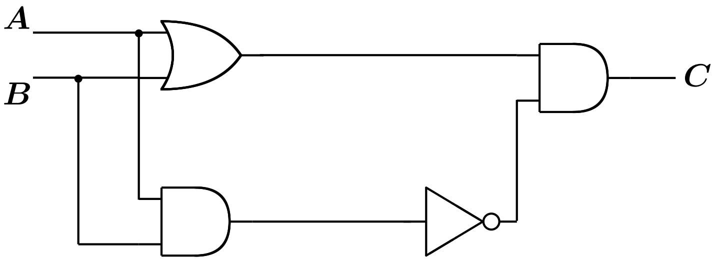
The OR gate has two inputs, switch A and switch B. The bulb Q will light if either switch A or B are closed. This will allow current to flow through the bulb, illuminating the filament

Since the 1990s, most logic gates are made of transistors

(semiconductor devices used to amplify and switch electric signals)

Transistors are so small that hundreds of thousands fit on one processing chip on a computer motherboard

Example: Boolean circuit



What does this circuit do?

- Represent the circuit as a **Boolean equation**

$$\boxed{C \;=\; (A \vee B) \wedge \neg(A \wedge B)}$$

- Construct the truth-table

A	B	$(A$	$\vee$	$B)$	$\wedge$	$\neg$	$(A$	$\wedge$	$B)$
0	0	0	0	0	0	1	0	0	0
0	1	0	1	1	1	1	0	0	1
1	0	1	1	0	1	1	1	0	0
1	1	1	1	1	0	0	1	1	1

The circuit, the truth-table and the formula $(A \vee B) \wedge \neg(A \wedge B)$ represent the **Boolean function** known as **exclusive or** and denoted by **XOR**, or $A \oplus B$

Boolean functions of one argument

(intuitive explanation of 'function': https://en.wikipedia.org/wiki/Function_(mathematics))

A	0
0	0
1	0

— **constant function** 0 (always returns 0 and doesn't depend on A)

draw a Boolean circuit for this function

A	1
0	1
1	1

— **constant function** 1 (always returns 1 and doesn't depend on A)

draw a Boolean circuit for this function

A	A
0	0
1	1

— **identical function** 0 (always returns the input A)

draw a Boolean circuit for this function

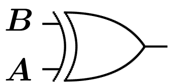
A	$\neg A$
0	1
1	0

— **function NOT** or $\neg$ (inverts the input A)

draw a Boolean circuit for this function

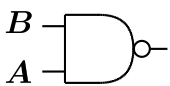
Boolean functions of two arguments

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \oplus B$	$A \leftrightarrow B$	$A \mid B$	$A \downarrow B$
0	0	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0
1	0	0	1	0	1	0	1	0
1	1	1	1	1	0	1	0	0

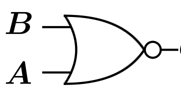
XOR gate 

$C = A \oplus B$

$A \leftrightarrow B$ — **equivalence** (if, and only if), equalivalent to $(A \rightarrow B) \wedge (B \rightarrow A)$

NAND gate 

$C = A \mid B \;=\; \neg(A \wedge B)$ **Scheffer stroke**

NOR gate 

$C = A \downarrow B \;=\; \neg(A \vee B)$ **Pierce arrow**

- What functions are missing here?
- What is the number of Boolean functions of two arguments?

Important questions

There are very many Boolean functions: 2^{2^n} distinct functions of n variables

For example, there are $2^{2^5} = 4,294,967,296$ functions with 5 inputs

We don't know *a priori* which of them are required in computer architecture

- Is it possible to fix some, relatively simple set(s) of Boolean functions (gates), using which one can built **all** other Boolean functions?

We have already seen that the same Boolean functions can be realised in different ways using different gates.

Of course we need smallest possible circuits (formulas)...

- How to build 'optimal' Boolean circuits (formulas)?
- How to simplify Boolean circuits (formulas)?
- What basic gates to choose?

We consider some aspects of these problems.

Example: the majority function

Suppose we want to realise, using only the AND, OR and NOT gates, the **majority function** $\mu(A, B, C)$ whose output takes the same value as the **majority** of inputs:

A	B	C	$\mu(A, B, C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Each triple of truth-values for A, B, C on the left-hand side of the table for which $\mu(A, B, C) = 1$ can be represented as conjunctions in the following way:

$0 \quad 1 \quad 1$ is represented by $\neg A \wedge B \wedge C$

it equals 1 if and only if $A = 0$, $B = 1$, $C = 1$

$1 \quad 0 \quad 1$ is represented by $A \wedge \neg B \wedge C$

it equals 1 if and only if $A = 1$, $B = 0$, $C = 1$

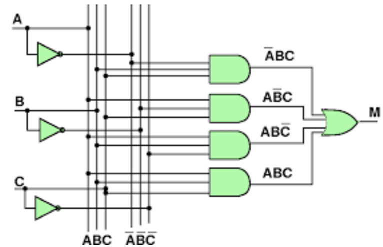
etc.

The function $\mu(A, B, C)$ can then be realised as a disjunction of the resulting four conjunctions:

$$\boxed{(\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)}$$

Example: the majority function (cont.)

- Use the formula above to construct a Boolean circuit for $\mu(A, B, C)$



- Can you simplify it?

Consider, for instance, the formula

$$\boxed{(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)}$$

- Does it define the same function? (Construct the truth-table)
- Is the corresponding circuit simpler?
- Can you simplify it?

  what about the formula $(A \wedge (B \vee C)) \vee (B \wedge C)$?

Universal sets of Boolean functions

The method shown on page 24 can be used to represent **any Boolean function** by means of a formula with the connectives $\neg$, $\wedge$ and $\vee$

if there is no 1 among the function values then this function is $\mathbf{0}$, which can be represented as $A \wedge (\neg A)$

We say that $\{\neg, \wedge, \vee\}$ is a **universal** set of Boolean connectives/functions

- Are there other universal sets of Boolean formulas?
- Can simplifications like those on page 25 be done in a systematic way?

Boolean formulas φ, ψ are called **equivalent** if their truth-tables are identical. In this case we write $\varphi \equiv \psi$.

(Greek letters φ, ψ, χ are often used to denote formulas)

As equivalent formulas φ and ψ determine the same Boolean function, we can use either of them to construct Boolean circuits

Useful equivalences

(φ and ψ are arbitrary Boolean formulas)

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$
$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$
(De Morgan laws)

$\neg\neg\varphi \equiv \varphi$ (the law of double negation)

$\neg\varphi \vee \varphi \equiv \mathbf{1}$ (the law of the excluded middle, 'to be or not to be')

$\neg\varphi \wedge \varphi \equiv \mathbf{0}$ (the law of contradiction)

$\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$ (distributivity of $\wedge$ over $\vee$)

$\varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)$ (distributivity of $\vee$ over $\wedge$)

$\varphi \wedge \mathbf{1} \equiv \varphi, \quad \varphi \wedge \mathbf{0} \equiv \mathbf{0}, \quad \varphi \vee \mathbf{1} \equiv \mathbf{1}, \quad \varphi \vee \mathbf{0} \equiv \varphi, \quad \varphi \wedge \varphi \equiv \varphi, \quad \varphi \vee \varphi \equiv \varphi$

It follows, for instance, that
$$\varphi \vee \psi \equiv \neg((\neg\varphi) \wedge (\neg\psi))$$
$$\varphi \wedge \psi \equiv \neg((\neg\varphi) \vee (\neg\psi))$$

Thus, we can express $\vee$ by means of $\neg$ and $\wedge$;
likewise, $\wedge$ can be expressed by means of $\neg$ and $\vee$

So both $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are universal (e.g., $\varphi \to \psi \equiv \neg\varphi \vee \psi$)

How to show equivalence: method 1

$$
\begin{aligned}
A \vee (A \wedge B) &\equiv (A \wedge 1) \vee (A \wedge B) \\
&\equiv (A \wedge (B \vee \neg B)) \vee (A \wedge B) \\
&\equiv (A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B) \\
&\equiv (A \wedge B) \vee (A \wedge B) \vee (A \wedge \neg B) \\
&\equiv (A \wedge B) \vee (A \wedge \neg B) \\
&\equiv A \wedge (B \vee \neg B) \\
&\equiv A \wedge 1 \\
&\equiv A
\end{aligned}
$$

Thus, $A \vee (A \wedge B) \equiv A$

How to show equivalence: method 2

Exercise 1: Show that $P \oplus Q \equiv (P \vee Q) \wedge \neg(P \wedge Q)$

Solution:

P	Q	$P \oplus Q$
1	1	0
1	0	1
0	1	1
0	0	0

P	Q	$P \vee Q$	$P \wedge Q$	$\neg(P \wedge Q)$	$(P \vee Q) \wedge \neg(P \wedge Q)$
1	1	1	1	0	0
1	0	1	0	1	1
0	1	1	0	1	1
0	0	0	0	1	0

Exercise 2: Show that $P \leftrightarrow Q \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$

Solution:

P	Q	$P \leftrightarrow Q$
1	1	1
1	0	0
0	1	0
0	0	1

P	Q	$P \wedge Q$	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$	$(P \wedge Q) \vee (\neg P \wedge \neg Q)$
1	1	1	0	0	0	1
1	0	0	0	1	0	0
0	1	0	1	0	0	0
0	0	0	1	1	1	1

Universality of NAND

To prove that | (or NAND) is universal, it is enough to show using NAND we can express NOT and AND:

- $\neg A \equiv (A \mid A) \equiv \neg(A \wedge A) \equiv \neg A,$ because $A \wedge A \equiv A$
- $A \wedge B \equiv (A \mid B) \mid (A \mid B)$ why?

Exercise 1: Show that NOR is also universal

Exercise 2: A **2-to-1 multiplexer** has three inputs, say A_0, A_1 and S; the output is A_0 if $S = 0$ and A_1 if $S = 1$.

Design a Boolean circuit for the 2-to-1 multiplexer.

In general, a multiplexer selects one of many input signals and outputs that into a single line