Mathematical Logic – Part 1

At the end of this lecture you should be able to:

- define the term **proposition**;
- provide truth tables for the simple logical operators, including NOT, AND, OR, IMPLICATION, EQUIVALENCE and EXCLUSIVE OR;
- explain and interpret the above truth tables;
- construct truth tables for compound expressions, and determine whether two expressions are logically equivalent;
- explain the terms **tautology** and **contradiction**;
- use the laws of propositional logic to perform simple algebraic operations on logical expressions;
- provide examples of how mathematical logic is applied to the field of computing.

Mathematical Logic

- in mathematical logic we try to place a rigorous mathematical framework around everyday natural language;

- there are two main branches of logic that we will study in this course: **propositional logic** and **predicate logic**.

Propositional Logic

A **proposition** is a statement that can be either TRUE or FALSE.

We say that it has a **truth value**.

Examples of propositions that have a value of TRUE:

- The angles of a triangle add up to 180°.
- Paris is the capital of France.
- 3 + 2 = 5.

Examples of propositions that have a value of FALSE:

- The angles of a triangle add up to 360°.
- Paris is the capital of Scotland.
- 3 + 2 = 7.

We can represent propositions by variable names such as P or Q. For example:

P: It is sunny.

Q: Today is Tuesday.

Logical operators (connectives)

- we can join two simple propositions together to from a compound statement, using **operators** (or **connectives**) that try to capture the meaning of simple words like “and” and “or”;

- the truth value of a compound statement will depend on the value of each of the two simple statements;

- each connective has a set of rules that are defined in the form of a **truth table**.

The AND operator ($\wedge$)

If P and Q are propositions then:

$$P \wedge Q \text{ means: } \quad P \text{ AND } Q\,.$$

The meaning of the AND operator is defined in the following truth table (where T means true and F means false)

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \wedge Q}$
T	T	T
T	F	F
F	T	F
F	F	F

If P is the statement “It is Wednesday”, and Q is the statement “It is February”, then:

$P \wedge Q$ is the statement “It is Wednesday and it is February”.

The compound statement is true only if both of the simple statements are true – otherwise it is false.

Worked example

If P and Q represent the following statements:

P: 102 < 50

Q: Nigeria is in Africa.

What is the value of $P \wedge Q$?

<u>Solution</u>

P is false
Q is true

Looking at the third line of the truth table we see: F $\wedge$ T is false

Therefore: $P \wedge Q$ is false

The OR operator ($\vee$)

If P and Q are propositions then:

$P \vee Q$ means: P OR Q .

The meaning of the OR operator is defined in the following truth table:

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \vee Q}$
T	T	T
T	F	T
F	T	T
F	F	F

If P is the statement “It is Wednesday”, and Q is the statement “It is February”, then:

$P \vee Q$ is the statement “It is Wednesday or it is February”.

The compound statement is true if either one of the simple statements is true – it is false only when both statements are false.

Worked example

If P and Q represent the following statements:

P: 102 < 50

Q: Nigeria is in Africa.

What is the value of $P \vee Q$?

<u>Solution</u>

P is false
Q is true

Looking at the third line of the truth table we see: F $\vee$ T is true

Therefore: $P \vee Q$ is true

The NOT operator (¬)

Unlike the previous two operators, the **not** operator operates on a single proposition.

$\neg P$ is read as *not* P.

Its function is to reverse the value of the proposition.

If P is true, then $\neg P$ is false.

If P is false then $\neg P$ is true.

For example:

If P represents the statement *it is raining.*

then $\neg P$ represents the statement *it is not raining.*

The truth table for ¬

$\boldsymbol{P}$	$\boldsymbol{\neg P}$
T	F
F	T

Note: you might also come across the notation ~P and !P to mean *not* P.

Worked examples

1. Let P be "It is cold" and Q be "It is raining". Give simple sentences which represent the following statements:

   a) $\neg P$
   b) $P \wedge Q$
   c) $P \vee Q$
   d) $Q \vee \neg P$
   e) $\neg P \wedge \neg Q$
   f) $\neg \neg Q$

   <u>Solution</u>

   a) It is not cold.
   b) It is cold and it is raining.
   c) It is cold or it is raining.
   d) It is raining or it is not cold.
   e) It is not cold and it is not raining.
   f) It is raining.

2. Let P be “She is tall” and Q be “She is intelligent”. Express each of the following statements symbolically:

   a) She is tall and intelligent.
   b) She is tall, but not intelligent.
   c) It is false that she is tall or intelligent.
   d) She is neither tall nor intelligent.
   e) She is tall, or she is short and intelligent.
   f) It is not true that she is short or unintelligent.

   <u>Solution</u>

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

Constructing truth tables

Consider the following expression: $(P \land Q) \lor \neg R$

This expression contains three variables, P, Q and R; the overall value of the expression (TRUE or FALSE) will depend on the value of these three variables.

We can construct a truth table to show all possible combinations.

In general if there are n variables the number of rows in the table will be 2^n.

In this case there are 3 variables so the number of rows will be 2^3 or 8.

1. Enter all the possible combinations in columns 1 to 3.
2. Using the values for P and Q in columns 1 and 2, complete column 4 by using the rules for AND.
3. Complete column 5 by negating the values for R in column 3.
4. Using the values in columns 4 and 5, complete column 6 by using the rules for OR.

1	**2**	**3**	4	5	6
P	Q	R	$P \land Q$	$\neg R$	$(P \land Q) \lor \neg R$
T	T	T	T	F	T
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	F	T	T
F	T	T	F	F	F
F	T	F	F	T	T
F	F	T	F	F	F
F	F	F	F	T	T

Logical equivalence

Two compound statements are said to be **logically equivalent** if they have identical truth tables. We use the symbol $\equiv$ to indicate that two statements are identical.

Worked example

Prove the following identity:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

<u>Solution</u>

The left hand side

P	Q	$P \wedge Q$	$\neg(P \wedge Q)$
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

The right hand side

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

The last column of each truth table is identical – so both statements are equivalent.

De Morgan’s Law

In the previous slide we showed that:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

It can also be shown that:

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Together these laws are known as **De Morgan’s Law**, and you will remember a similar law from set theory.

The laws of propositional algebra and the laws of set theory are isomorphic – which means that “they have the same shape”.

De Morgan’s Law makes sense

In a previous example we were asked to express the following statements symbolically, where P represented “She is tall” and Q represented “She is intelligent”.

- It is false that she is tall or intelligent.
- She is neither tall nor intelligent.

In fact they mean exactly the same thing.

Our answers were:

- $\neg(P \vee Q)$
- $\neg P \wedge \neg Q$

And by de Morgan’s law these two statements are identical!

Worked examples

1. Use De Morgan's law to show that:

$\neg(\neg P \vee (P \wedge Q)) \equiv P \wedge (\neg P \vee \neg Q)$

<u>Solution</u>

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

$\equiv \neg\neg P \wedge \neg(P \wedge Q)$

$\equiv P \wedge (\neg P \vee \neg Q)$

2. Use truth tables to prove the second version of De Morgan's law:

$$\neg(P \lor Q) \equiv \neg P \land \neg Q$$

<u>Solution</u>

The left hand side

P	Q	$P \lor Q$	$\neg(P \lor Q)$
T	T	T	F
T	F	T	F
F	T	T	F
F	F	F	T

The right hand side

P	Q	$\neg P$	$\neg Q$	$\neg P \land \neg Q$
T	T	F	F	F
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

The last columns in both truth tables are the same, thus proving the identity.

Commutativity and associativity

Both the AND operator and the OR operator are commutative:

$$P \wedge Q \equiv Q \wedge P$$

$$P \vee Q \equiv Q \vee P$$

Both these operators are also associative:

$$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$$

$$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$$

You can prove the above identities by constructing truth tables.

The *implication* operator ($\Rightarrow$)

If P and Q are propositions then:

$P \Rightarrow Q$ means: IF P THEN Q.

or alternatively: P IMPLIES Q.

The truth table for $\Rightarrow$ is as follows:

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \Rightarrow Q}$
T	T	T
T	F	F
F	T	T
F	F	T

The *implication* operator (continued)

If P is the statement "It is sunny", and Q is the statement "I go shopping", then:

$P \Rightarrow Q$ is the statement "If it is sunny, I go shopping".

Look carefully at the truth table:

P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Line 1:	If the compound statement is true, then if it is sunny, I definitely go shopping.
Line 2:	If it is sunny and I do not go shopping, the compound statement must be false.
Lines 3 and 4:	If the compound statement is true, but it is not sunny, then maybe I go shopping (line 3) or maybe I do not (line 4).

The statement $P \Rightarrow Q$ being true tells us nothing about what happens if P is false (in this case it is not sunny). But if P is true then it *guarantees* that Q is true.

P is a *sufficient* condition for Q
but: P is not a *necessary* condition for Q.

If $P \Rightarrow Q$ is true, then P is true only if Q is true; if I do not go shopping, it isn't sunny.

For this reason we can interpret $P \Rightarrow Q$ as: P **only if** Q.

Q is a *necessary* condition for P.

The *equivalence* operator ($\Leftrightarrow$)

If P and Q are propositions then:

$P \Leftrightarrow Q$ means: P IS EQUIVALENT TO Q.

The truth table for $\Leftrightarrow$ is as follows:

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \Leftrightarrow Q}$
T	T	T
T	F	F
F	T	F
F	F	T

The *equivalence* operator (continued)

If P is the statement “It is sunny”, and Q is the statement “I go shopping”, then:

$P \Leftrightarrow Q$ is the statement “If it is sunny, I go shopping, and if I go shopping, it is sunny”.

Look carefully at the truth table:

P	Q	$P \Leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Line 3 now gives a result of false.

So if the statement is true (lines 1 and 4), then if it is sunny I go shopping, if it is not sunny I do not go shopping.

The equivalence operator takes away the uncertainty of the implies operator. I go shopping only if it is sunny; also it is sunny only if I go shopping.

For this reason we can interpret $P \Leftrightarrow Q$ as:

P **if and only if** Q (sometimes written as P iff Q)

P is a *necessary and sufficient* condition for Q.
Q is a *necessary and sufficient* condition for P.

Worked example

Construct a truth tables for the following expression:

$$Q \Leftrightarrow (\neg P \vee \neg Q)$$

Solution

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$	$Q \Leftrightarrow (\neg P \vee \neg Q)$
T	T	F	F	F	F
T	F	F	T	T	F
F	T	T	F	T	T
F	F	T	T	T	F

The equivalence operator versus logical equivalence

It is important not to confuse the equivalence operator ($\Leftrightarrow$) with logical equivalence ($\equiv$).

The first is an *operator* that joins two propositions: each of the two propositions can be true or false, and the truth of the compound statement is determined from the truth table.

The second is used with two statements (usually compound statements) and means that they are identical because they have the same truth table.

Order of precedence of logical operators

Just as with arithmetic operators, it is not always immediately clear in which order to do things.

The order in which operations should be performed is as follows:

$$\begin{array}{c}\neg \\ \wedge \\ \vee \\ \Rightarrow \\ \Leftrightarrow\end{array}$$

However, it is recommended that brackets are always used in any compound statement that contains several operators.

For example, if you were asked to construct a truth table for this expression:

$$P \Rightarrow Q \vee R \wedge Q$$

you would start with a column for $R \wedge Q$, then do $Q \vee R \wedge Q$, and finally $P \Rightarrow Q \vee R \wedge Q$.

However, you will agree that it is a lot clearer if you write it like this:

$$P \Rightarrow (Q \vee (R \wedge Q))$$

Tautologies and contradictions

A **tautology** is an expression for which all rows of the truth table evaluate to TRUE.

For example: $P \vee \neg P$

P	$\neg P$	$P \vee \neg P$
T	F	T
F	T	T

A **contradiction** is an expression for which all rows of the truth table evaluate to FALSE.

For example: $P \wedge \neg P$

P	$\neg P$	$P \wedge \neg P$
T	F	F
F	T	F

Worked example

By constructing a truth table, determine whether the following expression is a tautology, a contradiction or neither:

$$(P \wedge (P \Rightarrow Q)) \Rightarrow Q$$

<u>Solution</u>

P	Q	$P \Rightarrow Q$	$P \wedge (P \Rightarrow Q)$	$(P \wedge (P \Rightarrow Q)) \Rightarrow Q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

All the rows in the final column are true, so the expression is a tautology.

The EXCLUSIVE OR operator (⊕)

With the OR operator, all that is required for the compound statement to be true is that either one of the simple statements is true.

With the EXCLUSIVE OR operator (sometimes known as XOR), the compound statement is true only when one (and only one) proposition is true. If both are true, then the compound statement is false.

P	Q	$P \oplus Q$
T	T	F
T	F	T
F	T	T
F	F	F

<u>Note</u>

You might observe that XOR is actually closer to the way we use the word “or” in natural language than is OR.

If you said “Tonight we are going to the cinema or we are going ice-skating”, most people would assume you were doing one or the other but not both!

Converse, inverse and contrapositive

- $Q \Rightarrow P$ is referred to as the **converse** of $P \Rightarrow Q$
- $\neg P \Rightarrow \neg Q$ is referred to as the **inverse** of $P \Rightarrow Q$
- $\neg Q \Rightarrow \neg P$ is referred to as the **contrapositive** of $P \Rightarrow Q$

The contrapositive is equivalent to the original statement, as can be seen from the truth tables below.

P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

P	Q	$\neg P$	$\neg$**Q**	$\neg Q \Rightarrow \neg P$
T	T	F	F	T
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

Algebra of propositions

You will see with the laws that follow that there are parallels with set theory.

We say there is an **isomorphism** between set theory and propositional logic – this means that some of the laws have "the same shape".

We have already seen that AND and OR are commutative and associative, as are intersection and union in set theory.

The following laws will also look familiar:

Idempotent Laws

$P \wedge P \equiv P$

$P \vee P \equiv P$

Identity Laws

$P \wedge \mathsf{F} \equiv \mathsf{F}$

$P \wedge \mathsf{T} \equiv P$

$P \vee \mathsf{F} \equiv P$

$P \vee \mathsf{T} \equiv \mathsf{T}$

Complement Laws

$P \vee \neg P \equiv \mathsf{T}$

$P \wedge \neg P \equiv \mathsf{F}$

$\neg \mathsf{T} \equiv \mathsf{F}$

$\neg \mathsf{F} \equiv \mathsf{T}$

Distributive law

$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

De Morgan's law

$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

You can see that $\wedge$ and $\vee$ correspond to $\cap$ and $\cup$ respectively, while T and F correspond to U and $\varnothing$.

In addition, negation in logic corresponds to complement in set theory.

Worked example

Use the distributive law to simplify the following expression:

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

<u>Solution</u>

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

$\equiv (P \vee \neg P) \wedge (P \vee Q)$ (Distributive Law)

$\equiv \mathrm{T} \wedge (P \vee Q)$ (Complement Law)

$\equiv P \vee Q$ (Identity Law)

Some more algebra

Look at this expression: $\neg P \vee Q$

Here is its truth table:

P	Q	$\neg P$	$\neg P \vee Q$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

Ring any bells?

This is also the truth table for $P \Rightarrow Q$.

So we have the following identity: $P \Rightarrow Q \equiv \neg P \vee Q$

(We will refer to this as identity 1)

Negating implication

Let's see what happens if we negate $P \Rightarrow Q$:

Using identity 1 from the previous slide:

$$\neg(P \Rightarrow Q) \equiv \neg(\neg P \vee Q)$$

Remember De Morgan's law? We can use this on the right-hand side of the identity:

$$\neg(\neg P \vee Q) \equiv P \wedge \neg Q$$

So: $$\neg(P \Rightarrow Q) \equiv P \wedge \neg Q$$

(We will refer to this as identity 2)

Worked examples

1. Use algebra to show that: $(P \wedge Q) \Rightarrow \neg Q \equiv \neg P \vee \neg Q$

<u>Solution</u>

$(P \wedge Q) \Rightarrow \neg Q$	
$\equiv \neg(P \wedge Q) \vee \neg Q$	Using identity 1
$\equiv \neg P \vee \neg Q \vee \neg Q$	Using De Morgan’s law
$\equiv \neg P \vee (\neg Q \vee \neg Q)$	Using the fact that $\vee$ is associative
$\equiv \neg P \vee \neg Q$	Idempotent law

2. Negate the following statement and simplify the answer.

$$(P \wedge Q) \Rightarrow R$$

<u>Solution</u>

$\neg((P \wedge Q) \Rightarrow R)$	
$\equiv (P \wedge Q) \wedge \neg R$	Using identity 2
$\equiv P \wedge Q \wedge \neg R$	Using the fact that $\wedge$ is associative

Application to computing

1. Programming

One of the most common statements used in program code is the *conditional* statement.

An example In Java might look like this:

```
if(x < 10 || y > 100)
{
   // statements go here
}
```

The || symbol in Java (and many other languages) means OR, corresponding to our logical OR. The symbol && is used in Java for the logical AND.

The compiler will use the rules we have learnt to evaluate the statement in brackets and thereby determine if it is TRUE or FALSE.

If it is true, the statements in the braces will be executed – otherwise, nothing will happen and the program will move on to the next bit of code following the braces.

In this way, the **`if`** statement implements the implication operator, $\Rightarrow$. It does not say anything about what what to do if the statement is false (other to do nothing and move on).

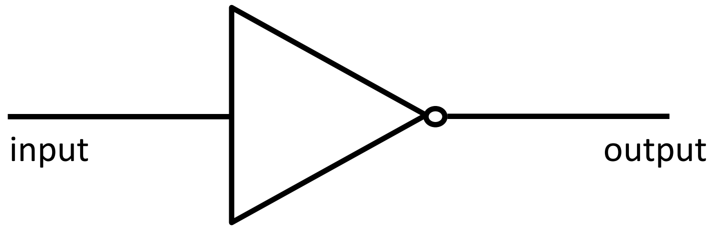
2. Digital electronics

- modern electronic devices such as mobile telephones and computers depend on digital electronics;
- at the heart of digital electronics are **logic circuits**;
- such circuits depend on pulses of electricity to make the circuit function;
- for example, if the current is high, this is represented as ‘1’, if the current low, this is represented as ‘0’;
- this makes it possible to represent and process binary numbers in a computer system.

Logic gates

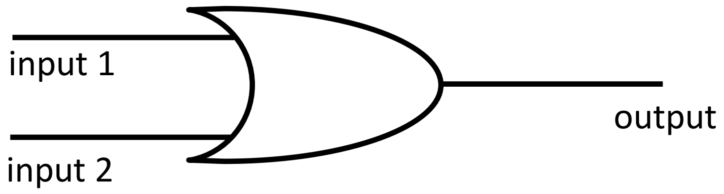
- mathematical logic is based on a two-valued system (TRUE and FALSE);
- in digital electronics we also have a two-valued system but have 1 and 0 instead of TRUE and FALSE;
- many of the operations that need to be performed on binary numbers within a computer system mirror the logical operations such as NOT, AND, OR;
- we are able to build **logic gates** that perform these, and other more complex, operations;
- three examples are shown on the next slide.

NOT gate



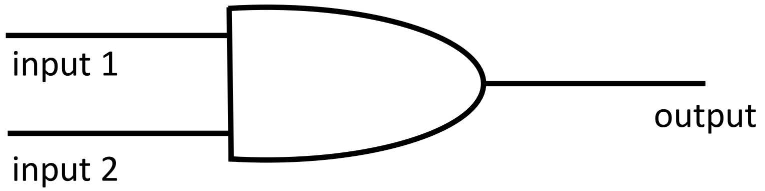
Input	Output
1	0
0	1

OR gate



Input 1	Input 2	Output
1	1	1
1	0	1
0	1	1
0	0	0

AND gate



Input 1	Input 2	Output
1	1	1
1	0	0
0	1	0
0	0	0

Notation in other texts

Unfortunately, you will see different notation used in different texts. Some texts use single-barred arrows for implication and equivalence:

So instead of $\Rightarrow$ and $\Leftrightarrow$

you will see $\rightarrow$ and $\leftrightarrow$.

To make matters worse, in texts that use this notation you will often see $\Leftrightarrow$ used for logical equivalence instead of $\equiv$.

In fact, some mathematicians actually make a distinction between the single-barred arrow and the double one.

They argue that you should use the single-barred arrows for the logical connectives that join propositions. But you should use the double-barred arrow for an **assertion** – that is a statement that is *true*.

In this notation:

$P \rightarrow Q$ means that P implies Q, and P and Q can take any values.

But: $P \Rightarrow Q$ means that P implies Q is *true* (so in this case we can't have P true and Q false).

For interest only

Three-valued logic

Classical logic assumes that all expressions evaluate to TRUE or FALSE;

However, in reality, this is not always the case when evaluating an expression, because sometimes an expression can be undefined. For example, the expression 0/0.

Also, undefined terms are very common in programming situations; for example, when a variable is first declared and has not yet been assigned a value.

A system of three-valued logic has therefore been developed to deal with the possibility that a proposition could have the value TRUE, FALSE or UNDEFINED.

The truth tables for AND and OR in this system are shown below:

P	Q	$P \wedge Q$
T	T	**T**
T	F	**F**
T	UNDEFINED	**UNDEFINED**
F	T	**F**
F	F	**F**
F	UNDEFINED	**F**
UNDEFINED	T	**UNDEFINED**
UNDEFINED	F	**F**
UNDEFINED	UNDEFINED	**UNDEFINED**

P	Q	$P \vee Q$
T	T	**T**
T	F	**T**
T	UNDEFINED	**T**
F	T	**T**
F	F	**F**
F	UNDEFINED	**UNDEFINED**
UNDEFINED	T	**T**
UNDEFINED	F	**UNDEFINED**
UNDEFINED	UNDEFINED	**UNDEFINED**