INTRODUCTION

The aim of logic in computer science is to develop languages to model the situations we encounter, in such a way that we can reason about them formally. Reasoning about situations means constructing arguments about them and this need to done formally, so that the arguments are valid and can be defended or executed on a machine. Arguments to be considered should consist of a number of sentences followed by the word 'therefore' and then another sentence. The argument is valid if the sentence after the 'therefore' logically follows from the sentences before it.

LOGIC

Logic is a formal system in which the formulas or sentences have true or false values. A *logic* consists of the following:

i). **Syntax**: Specifies the symbols in the language and how they can be combined to form sentences. Hence facts about the world are represented as sentences in logic
ii). **Semantics**: Specifies what facts in the world a sentence refers to. Hence, also specifies how you assign a truth value to a sentence based on its meaning in the world.
iii). **Inference Procedure**: Mechanical method for computing (deriving) new (true) sentences from existing sentences

To build a logic-based representation:

- User defines a set of primitive symbols and the associated semantics
- Logic defines the ways of putting these symbols together so that the user can define legal sentences in the language that represent true facts in the world
- Logic defines ways of inferring new sentences from existing ones

Facts are claims about the world that are True or False, whereas a **representation** is an expression (sentence) in some language that can be encoded in a computer program and stands for the objects and relations in the world

Truth: A sentence is True if the state of affairs it describes is actually the case in the world. So, truth can only be assessed with respect to the semantics. Yet the computer does not know the semantics of the knowledge representation language, so we need some way of performing inferences to derive valid conclusions even when the computer does not know what the semantics (the interpretation) is .

TYPES OF LOGIC

i) PROPOSITIONAL LOGIC (PL)

Propositional logic consists of symbols that represent whole propositions (facts). For example B may represent 'It is sunny outside', that may be true or false. Propositions may be combined using Boolean connectives which generate sentences with more complex meanings. Little commitment is made to how things are represented thus limiting propositional logic as representation language.

In developing logics, you should not be concerned with what the meaning of sentences, but rather their logical structure.

The following are examples of propositions:

(1). The sum of the numbers 3 and 5 equals 8.
(2). Jane reacted violently to Jack's accusations.
(3). Every even natural number >2 is the sum of two prime numbers.

Examples of non propositions are:

- Could you please pass me the salt?
- Ready, steady, go!
- May fortune come your way.

A **sentence** (also called a formula or well-formed formula or wff) is defined as:

- A symbol
- If S is a sentence, then ~S is a sentence, where "~" is the "not" logical operator
- If S and T are sentences, then (S v T), (S ^ T), (S => T), and (S <=> T) are sentences.

Examples of PL sentences:

- (P ^ Q) => R (here meaning "If it is hot and humid, then it is raining")
- Q => P (here meaning "If it is humid, then it is hot")
- Q (here meaning "It is humid.")

Note: Syntactically correct logical formulas are called *well-formed formulas* (wff). Such wff are thus used to represent real-world facts.

Propositional Calculus Semantics
An **interpretation** of a set of propositions is the assignment of a truth value, either T or F to each propositional symbol. The symbol true is always assigned T, and the symbol false is assigned F.

Model:- This is the mappings from proposition symbols directly to truth or falsehood. The models of a sentence in this case are the mappings that make a sentence true.

Logical Connectors
Logical connectors (&, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$) are used to connect formulas (propositions or predicates) so as to represent more complex facts. The truth value of these expressions depends on the truth values of their components, according to the following rules:

X & Y is TRUE if X is TRUE and Y is TRUE; otherwise X & Y is FALSE.

$X \vee Y$ is TRUE if either X is TRUE or Y is TRUE or both.

$\neg X$ is TRUE if X is FALSE, and FALSE if X is TRUE.

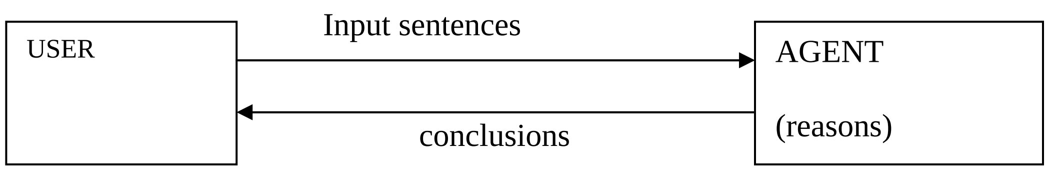
$X \rightarrow Y$ means X implies Y. This implication is true if X is false or Y is TRUE.

$X \leftrightarrow Y$ is TRUE if both X and Y are TRUE, or both X and Y are false.

The truth table below summarizes the above operations

		AND	OR	IMPLICATION	NEGATION	EQUIVALENCE
X	Y	$X \wedge Y$	$X \vee Y$	$X \rightarrow Y$	$\neg X$	$X \leftrightarrow Y$
T	T	T	T	T	F	T
T	F	F	T	F	F	F
F	T	F	T	T	T	F
F	F	F	F	T	T	T

Validity: - A valid sentence (also called a tautology) is a sentence that is True under all interpretations. Hence, no matter what the world is actually like or what the semantics is, the sentence is True. For example "It's raining or it's not raining.". Validity usually enables the machine to check premises to determine if the conclusion is true.
A sentence of the form: Premises $\Rightarrow$ Conclusion is valid if all the rows of $\Rightarrow$ are all true. A truth table can be used to check if a sentence is valid.



Contradiction:- An inconsistent sentence (also called a contradiction) is a sentence that is False under all interpretations. Hence the world is never like what it describes. For example, "It's raining and it's not raining."

Entailment: Sentence P **entails** sentence Q, written P |= Q, means that whenever P is True, so is Q. In other words, all models of P are also models of Q

Inference:- The process of determining if a sentence if true is called *inference*. The machine can perform inference by building truth tables for the sentence and checking that in the sentence Premises ⇒ Conclusion, that the row is all true for the ⇒.

Prove in Propositional Calculus using a truth table
Using truth table, show the validity of the following statement:

P∧Q ⇔ Q∧ P

P	**Q**	**P∧Q**	**Q∧ P**	**P∧Q ⇔ Q∧ P**
T	T	T	T	T
T	F	F	F	T
F	T	F	F	T
F	F	F	F	T

The sentence is True under all interpretations (tautology). Therefore it is valid

The truth table method of inference is **complete** for PL (Propositional Logic) because we can always enumerate all 2^*n* rows for the *n* propositional symbols that occur. (The truth table method of inference is *not* complete for FOL (First-Order Logic).)

Exercise:

1. Show the validity of the following statements:

(i) P∧(Q∧R) ⇔ (P∧Q) ∧R
(ii) P∨(Q∨R) ⇔ (P∨Q) ∨R
(iii) P∧Q ⇔ Q∧ P
(iv) P∨Q ⇔ Q∨ P
(v) P∧(Q∨R) ⇔ (P∧Q) ∨ (P∧R)
(vi) P∨ (Q∧R) ⇔ (P∨Q) ∧ (P∨R)
(vii) ¬(P∨Q) ⇔ ¬Q∧ ¬P
(viii) P⇒Q ⇔ ¬Q⇒ ¬P
(ix) ¬¬P ⇔ P
(x) P⇒Q ⇔ ¬P∨Q
(xi) P⇔Q ⇔ P⇒Q ∧ Q⇒ P
(xii) P∧¬P ⇔ False
(xiii) P∨¬P ⇔ True

2. Using the "weather" sentences from above, let KB = (((P ^ Q) => R) ^ (Q => P) ^ Q) corresponding to the three facts we know about the weather: (1) "If it is hot and humid, then it is raining," (2) "If it is humid, then it is hot," and (3) "It is humid." Now let's ask the query "Is it raining?" That is, is the query sentence R entailed by KB? Using the truth-table approach to answering this query we have (i.e. KB => R)

Rules of Inference
The inference rules of logic are used to infer new facts from the explicitly represented ones. The following is a list of some most commonly used inference rules.

i) **Modus Ponens (or implication-Elimination): -** If $(P \rightarrow Q)$ and P is true, then Q is true, written also as $(((P \rightarrow Q)$ and $P)$ |-- Q)
e.g.
- → P (meaning " it is hot")
- → Q => P ("If it is humid, then it is hot")
- → Q (meaning "It is humid.")

If we know that "Q => P " and "Q", then we can infer "P"

ii) **And-Elimination:** given a conjunction, you can infer any conjunct. Represented as:
$A_1 \wedge A_2 \wedge A_3 \wedge \ldots\ldots \wedge A_n \mid\!- A_i$
iii) **And-Introduction:** given a list of sentences, you can infer their conjunction. Represented as: $A_1, A_2, A_3, \ldots\ldots, A_n \mid\!- A_1 \wedge A_2 \wedge A_3 \wedge \ldots\ldots \wedge A_n$

iv) **Or-Introduction**: given a sentence, you can infer all its disjunctions with any thing else. Represented as: $A_I \mid\!- A_1 \vee A_2 \vee A_3 \vee \ldots\ldots \vee A_n$

v) **Double-Negation Elimination:** given a doubly negated sentence infer a positive sentence. Represented as: $\neg\neg A \mid\!- A$

vi) **Unit resolution:** given a disjunction, if one of the disjuncts is false then infer the other. This is a special case of resolution. Represented as: $A \vee B, \neg B \mid\!- A$, alternatively: $A \vee B, \neg A \mid\!- B$

vii) **Resolution:** implication is transitive. Represented as: $A \vee B, \neg B \vee C \mid\!- A \vee C$ or equivalently: $\neg A \Rightarrow B, B \Rightarrow C \mid\!- \neg A \Rightarrow C$.

viii) **The ∀-elimination (or universal specialization) rule: -** For any *well-formed formulas* (wff) of the form "$\forall x\ \Phi(x)$", we can conclude "$\Phi(A)$" for any individual "A". This rule is also written as $((\forall x\ \Phi(x))$ |-- $\Phi(A))$
For instance, if we know that
$\forall x\ Man(x) \rightarrow Mortal(x)$, we can apply this to the individual Socrates, to get
$Man(Socrates) \rightarrow Mortal(Socrates)$

Monotonicity
Monotonicity occurs where adding rules does not change the status of some other rules. Consider a knowledge base KB, that entails some sentences. Now suppose that some sentences are added. Then the sentences entailed in KB are also entailed in the new knowledge base. This may also be expressed as:

If $KB_1 \mid\!- A$ then $(KB_1 \cup KB_2) \mid\!- A$.

Using Inference Rules to Prove a Query/Goal/Theorem
A proof is a sequence of sentences, where each sentence is either a premise or a sentence derived from earlier sentences in the proof by one of the rules of inference. The last sentence is the query (also called goal or theorem) that needs to be proved.

Example of a prove for the "weather problem" given above.

1.	Q	Premise
2.	Q => P	Premise
3.	P	Modus Ponens(1,2)
4.	(P ^ Q) => R	Premise
5.	P ^ Q	And Introduction(1,3)
6.	R	Modus Ponens(4,5)

Two Important Properties for Inference

- **Soundness**: If KB |- Q then KB |= Q
  That is, if Q is derived from a set of sentences KB using a given set of rules of inference, then Q is entailed by KB. Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid.
- **Completeness**: If KB |= Q then KB |- Q
  That is, if Q is entailed by a set of sentences KB, then Q can be derived from KB using the rules of inference. Hence, inference produces *all* entailments, or all valid sentences can be proved from the premises.

Propositional Logic is Too Weak a Representational Language

Propositional Logic (PL) is not a very expressive language because:

- Hard to identify "individuals." E.g., Mary, 3
- Can't directly talk about properties of individuals or relations between individuals. E.g., tall(Bill)
- Generalizations, patterns, regularities can't easily be represented. E.g., all triangles have 3 sides

ii) FIRST ORDER PREDICATE LOGIC

Predicate logic or first order logic(FOL or FOPC), provides a way of representing the world in terms of objects and predicates on objects (properties of objects or relations between objects). *Connectives* are used to combine predicates.

- The objects from the real world are represented by constant symbols (a,b,c,...). e.g. the symbol "Tom" may represent a certain individual called Tom.
- Properties of objects may be represented by predicates applied to those objects (P(a), ...): e.g. "male(Tom)" represents that Tom is a male.
- Relationships between objects are represented by predicates with more arguments: e.g. "father (Tom, Bob)" represents the fact that Tom is the father of Bob.
- The value of a predicate is one of the boolean constants T (i.e. true) or F (i.e. false). e.g. "father(Tom, Bob) = T" means that the sentence "Tom is the father of Bob" is true. "father(Tom, Bob) = F" means that the sentence "Tom is the father of Bob" is false.

Besides constants, the arguments of the predicates may be functions (f,g,...) or variables (x,y,...).

Constant symbols are specifications of interpretations of particular objects. For example john in one interpretation may refer to good Leader, and in another to a friend. Some of the symbols include: A, B, C, Kimani.

Predicate symbols are specifications of interpretations in which particular relations are associated with particular models. In other words a symbol refers to a group of relations, such as brother, round, etc.

Tuple is a collection of objects arranged in a fixed order. For example consider a model of brotherhood consisting of John, Richard and Hood. The following set of tuples may be defined:

{<John, Richard>,

<Hood, Richard>}

Quantifiers
Quantifiers are statements that allow descriptions about the universe (objects environments) at once instead of numerating the objects, i.e. it extends PL by allowing quantification over individuals of a given domain of discourse. Two standard quantifiers in predicate logic are universal and existential quantifiers.

i) **Universal quantification (∀) : -** This quantification is used when the predicate is true for all objects. For example: ∀(x), cat(x) ⇒ mamal(x).

ii) **Existential quantification (∃): -** This quantifier is used when a predicate P is true for some object in the universe. For example some is tall may be denoted as ∃x tall(x). Someone is short is denoted as ∃x short(x). Someone likes a given person is denoted by ∃x, a likes(a, x).

Connection between ∃ and ∀
∃ and ∀ are linked by negation of each other. In the universe ∀ represents a conjunction and ∃ represents a disjunction.

Everyone dislikes paper		No one likes paper
∀x ¬likes(x, paper)	≡	¬∃x likes(x, paper).
Everyone likes sweets		Nobody does not like sweets
∀x likes(x, sweets)	≡	¬∃x ¬likes(x, sweets).

Atomic sentences: - Are sentences that consist of a predicate name followed by a number of arguments (arity).

Axioms: - Axioms are basic facts about a domain. Axioms and definitions are used to prove theorems. *Independent axioms* are those axioms that cannot be derived from other axioms.

An axiomatic system
To solve problems by using the logic representation one has to define an axiomatic system. An *axiomatic system* consists of a set of facts and inference rules (the axioms), representing real-world knowledge (situations). It can be used, for instance, to infer new facts or to prove that a certain fact is true.

Example
The following is an example of an axiomatic system:

1. Marcus was a man
2. Marcus was a Pompeian
3. All Pompeians were Romans
4. Caesar was a ruler
5. All Romans were either loyal to Caesar or hated him
6. Everyone is loyal to someone
7. People only try to assassinate rulers they are not loyal to
8. Marcus tried to assasinate Caesar
9. All men are persons

Qs. Translate these sentences into formulas in predicate logic

1. man(Marcus)
2. Pompeian(Marcus)
3. $\forall$x (Pompeian(x) $\rightarrow$ Roman(x))
4. ruler(Caesar)
5. $\forall$x (Roman(x) $\rightarrow$ loyalto(x,Caesar)$\vee$hate(x,Caesar))
6. $\forall$x $\exists$y (person(x) $\rightarrow$ person(y) & loyalto(x,y))
7. $\forall$x $\forall$y (person(x) & ruler(y) & tryassassinate(x,y) $\rightarrow$ $\neg$loyalto(x,y))
8. tryassassinate(Marcus,Caesar)
9. $\forall$x (man(x) $\rightarrow$ person(x))

Natural deduction: It involves applying a sequence of rules of inferences using either sentences in the KB or sentences derived earlier in the proof, until the conclusion sentences is derived

Example:
In the above axiomatic system, use natural deduction to prove that Marcus was not loyal to Caesar ($\neg$loyalto(Marcus, Caesar))

i).	From	$\forall$x (man(x) $\rightarrow$ person(x))
	deduce	man(Marcus) $\rightarrow$ person(Marcus) by universal specialization.
ii).	From	man(Marcus) $\rightarrow$ person(Marcus)
	and	man(Marcus)
	deduce	person(Marcus) by modus ponens.
iii).	From	$\forall$x $\forall$y (person(x) & ruler(y) & tryassassinate(x,y) $\rightarrow$ $\neg$loyalto(x,y)
	deduce	person(Marcus) & ruler(Caesar) & tryassassinate(Marcus, Caesar) $\rightarrow$ $\neg$loyalto(Marcus, Caesar)
		by applying the universal specialization twice.
iv).	From	person(Marcus) & ruler(Caesar) & tryassassinate(Marcus, Caesar) $\rightarrow$ $\neg$loyalto(Marcus, Caesar)
		and person(Marcus) & ruler(Caesar) & tryassassinate(Marcus, Caesar)
	deduce	$\neg$loyalto(Marcus, Caesar)
		by modus ponens.

The following is the proof tree:

$$\begin{tikzcd}
 & \neg\text{ loyalto(Marcus,Caesar)} & \\
\text{person(Marcus)} \arrow[ur] & \text{ruler(Caesar)} \arrow[u] & \text{tryassassinate(Marcus,Caesar)} \arrow[ul] \\
\text{man(Marcus)} \arrow[u] & &
\end{tikzcd}$$

Notice that a variety of processes, such as matching, substitution, and application of modus ponens are involved in the production of a proof.
A system implementing such proof procedures is called a *natural deductive system* because it uses a method similar to the one used by humans.
The main advantage is the understandability of the proof process and of the found proof.

The main disadvantage is that there are many different types of inference rules of logic (besides universal specialization and modus ponens). This complicates very much the proof process.
Note: It is possible to replace all the inference rules of logic with a single one, called the resolution rule, and to define an axiomatic system that uses only this rule. This requires, however, a modified syntax for logic, called the clausal form of logic.