Propositional Logic

It is a way of representing knowledge.
In logic and mathematics, a **propositional calculus** or **logic** is a formal system in which formulae representing *propositions* can be formed by combining atomic propositions using *logical connectives*
Sentences considered in propositional logic are not arbitrary sentences but are the ones that are either true or false, but not both. This kind of sentences are called **propositions**.

Example
Some facts in propositional logic:

It is raning.	-	RAINING
It is sunny	-	SUNNY
It is windy	-	WINDY

If it is raining ,then it is not sunny - RAINING -> $\neg$ SUNNY

The elements of propositional logic

Simple sentences which are true or false are basic propositions. Larger and more complex sentences are constructed from basic propositions by combining them with **connectives**. Thus **propositions** and **connectives** are the basic elements of propositional logic. Though there are many connectives, we are going to use the following **five basic connectives** here:

NOT, AND, OR, IF_THEN (or IMPLY), IF_AND_ONLY_IF.

They are also denoted by the symbols:

$\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, respectively.

Inference

Inference is deriving new sentences from old.

Modus Ponens

There are standard patterns of inference (logical thinking)that can be applied to derive chains of conclusions that lead to the desired goal. These patterns of inference are called **inference rules.** The best-known rule is called **Modus Ponens** and is written as follows:

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

The notation means that, whenever any sentences of the form $a \Rightarrow \beta$ and a are given, then the sentence β can be inferred. For example, if (*WumpusAhead A WumpusAlive*) $\Rightarrow$ *Shoot* and (*WumpusAhead A WumpusAlive*) are given, then *Shoot* can be inferred.

Another useful inference rule is **And-Elimination**, which says that, from a conjunction, any of the conjuncts can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

For example, from (*WumpusAhead A WumpusAlive*), *WumpusAlive* can be inferred.

Entailment

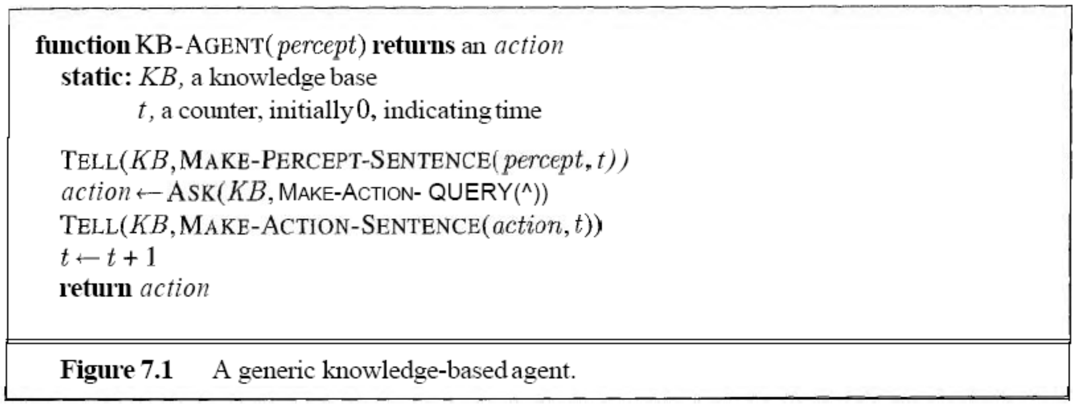
Propositions tell about the notion of truth and it can be applied to logical reasoning.

We can have logical entailment between sentences. This is known as entailment where a sentence follows logically from another sentence. In mathematical notation we write :

$$\alpha \models \beta$$

knowledge based agents

The central component of a knowledge-based agent is its knowledge base, or KB. Informally, a knowledge base is a set of sentences. Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world.



```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION- QUERY(^))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Figure 7.1 A generic knowledge-based agent.

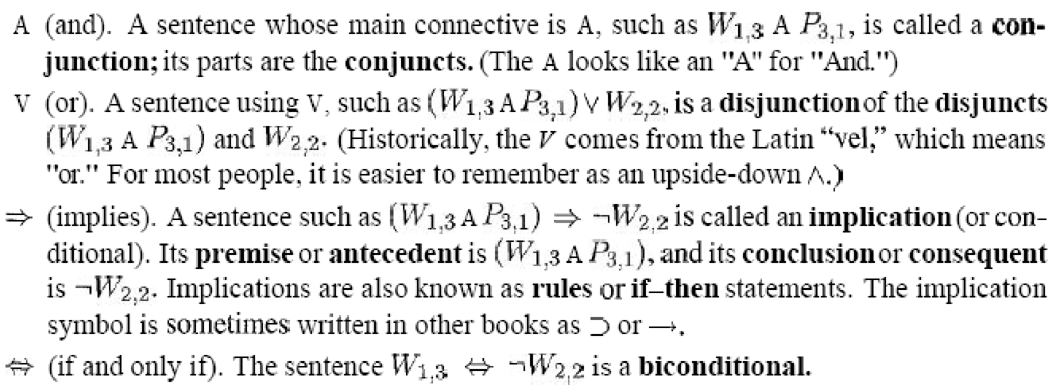
Figure 7.1 shows the outline of a knowledge-based agent program. Like all our agents, it takes a percept as input and returns an action. The agent maintains a knowledge base, KB, which may initially contain some **background knowledge.** Each time the agent program is called, it does three things. First, it TELLS the knowledge base what it perceives. Second, it ASKS the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.

Connectives used in propositional logic.

The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences** the indivisible syntactic elements-consist of a single **proposition symbol.** Each such symbol stands for a proposition that can be true or false. We will use uppercase names for symbols: P, Q, R, and so on.

Complex sentences are constructed from simpler sentences using **logical connectives.** There are five connectives in common use:

$\neg$ (not). A sentence such as $\neg W_{1,3}$ is called the **negation** of $W_{1,3}$. **A literal** is either an atomic sentence (a **positive literal)** or a negated atomic sentence (a **negative literal).**



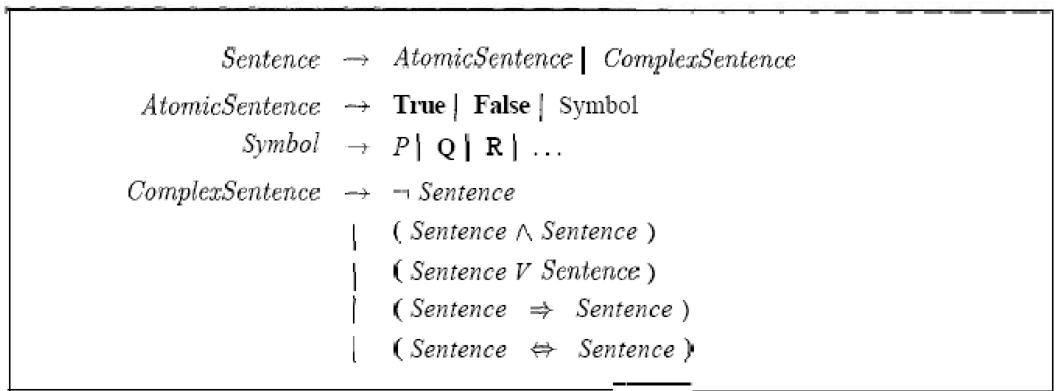
A (and). A sentence whose main connective is A, such as $W_{1,3}$ A $P_{3,1}$, is called a **conjunction;** its parts are the **conjuncts.** (The A looks like an "A" for "And.")

V (or). A sentence using V, such as $(W_{1,3} \text{ A } P_{3,1}) \vee W_{2,2}$, is a **disjunction** of the **disjuncts** $(W_{1,3} \text{ A } P_{3,1})$ and $W_{2,2}$. (Historically, the V comes from the Latin "vel," which means "or." For most people, it is easier to remember as an upside-down $\wedge$.)

$\Rightarrow$ (implies). A sentence such as $(W_{1,3} \text{ A } P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \text{ A } P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$. Implications are also known as **rules** or **if–then** statements. The implication symbol is sometimes written in other books as $\supset$ or $\rightarrow$.

$\Leftrightarrow$ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional.**

Figure 7.7 gives a formal grammar of propositional logic;



$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textbf{True} \mid \textbf{False} \mid \text{Symbol} \\
\textit{Symbol} & \rightarrow & P \mid \text{Q} \mid \text{R} \mid \ldots \\
\textit{ComplexSentence} & \rightarrow & \neg\, \textit{Sentence} \\
 & \mid & (\, \textit{Sentence} \wedge \textit{Sentence}\,) \\
 & \mid & (\, \textit{Sentence}\ V\ \textit{Sentence}\,) \\
 & \mid & (\, \textit{Sentence} \Rightarrow \textit{Sentence}\,) \\
 & \mid & (\, \textit{Sentence} \Leftrightarrow \textit{Sentence}\,)
\end{array}
$$

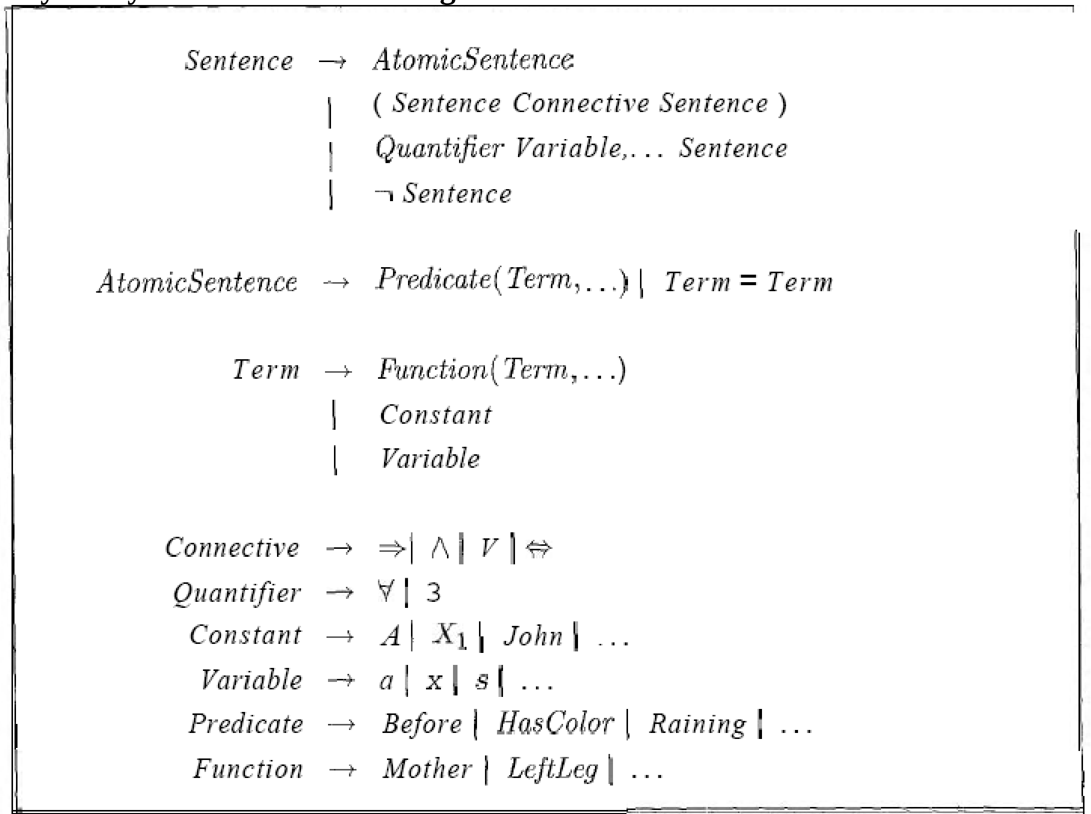
Figure 7.7 A BNF (Backus–Naur Form) grammar of sentences in propositional logic.

First order Logic

Whereas propositional logic assumes the world contains facts,
first-order logic (like natural language) assumes the world contains

Objects: people, houses, numbers, colors, baseball games, wars, …
Relations: red, round, prime, brother of, bigger than, part of, comes between, …
Functions: father of, best friend, one more than, plus, …

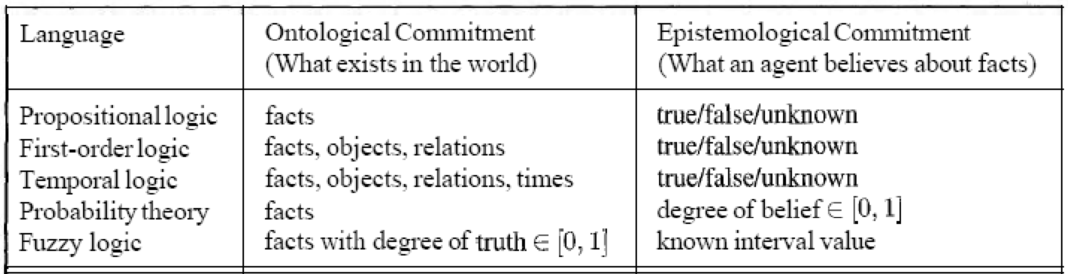
Specify the syntax of First-order logic in BNF form.



```
       Sentence → AtomicSentence
                | ( Sentence Connective Sentence )
                | Quantifier Variable,... Sentence
                | ¬ Sentence

 AtomicSentence → Predicate(Term,...) | Term = Term

           Term → Function(Term,...)
                | Constant
                | Variable

     Connective → ⇒ | ∧ | V | ⇔
     Quantifier → ∀ | 3
       Constant → A | X1 | John | ...
       Variable → a | x | s | ...
      Predicate → Before | HasColor | Raining | ...
       Function → Mother | LeftLeg | ...
```

Figure 8.3 The syntax of first-order logic with equality, specified in Backus–Naur form. (See page 984 if you are not familiar with this notation.) The syntax is strict about parentheses; the comments about parentheses and operator precedence on page 205 apply equally to first-order logic.

Comparison between different knowledge representation languages.



Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Figure 8.1 Formal languages and their ontological and epistemological commitments.

The syntactic elements of First Order Logic

The basic syntactic elements of first-order logic are the symbols that stand for objects,
relations, and functions. The symbols,come in three kinds:
a) constant symbols, which stand for objects;
b) predicate symbols, which stand for relations;
c) and function symbols, which stand for functions.
We adopt the convention that these symbols will begin with uppercase letters.
Example:
Constant symbols :
Richard and John;

predicate symbols :
Brother, OnHead, Person, King, and Crown;
function symbol :
LeftLeg.

Quantifiers

There is need to express properties of entire collections of objects,instead of enumerating the objects by name. Quantifiers let us do this.
FOL contains two standard quantifiers called

a) Universal (∀**)** and
b) Existential (∃**)**

Universal quantification

(∀x) P(x) : means that P holds for **all** values of x in the domain associated with that variable
E.g., (∀x) dolphin(x) => mammal(x)

Existential quantification

(∃ x)P(x) means that P holds for **some** value of x in the domain associated with that variable
E.g., (∃ x) mammal(x) ^ lays-eggs(x)
Permits one to make a statement about some object without naming it

Universal Quantifiers with an example.

Rules such as "All kings are persons," is written in first-order logic as
∀x King(x) => Person(x)
where ∀ is pronounced as “ For all ..”

Thus, the sentence says, "For all *x,* if *x* is a king, then z is a person."
The symbol x is called a variable(lower case letters)
The sentence ∀**x** P,where P is a logical expression says that P is true for every object x.

Existential quantifiers with an example.

Universal quantification makes statements about every object.
It is possible to make a statement about some object in the universe without naming it,by using an existential quantifier.

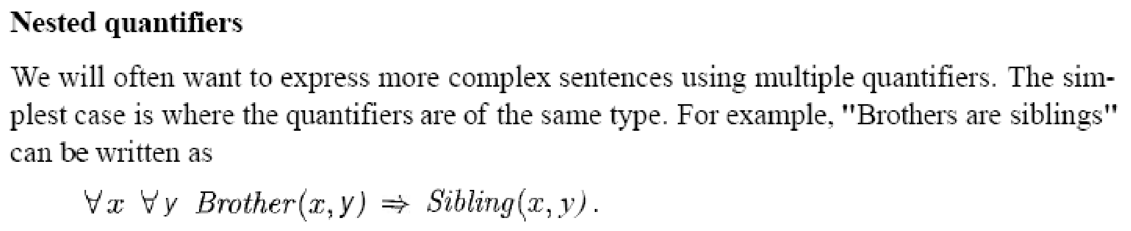
Example

“King John has a crown on his head”

∃x Crown(x) ^ OnHead(x,John)

∃x is pronounced “There exists an x such that ..” or “ For some x ..”

Nested quantifiers



Nested quantifiers

We will often want to express more complex sentences using multiple quantifiers. The simplest case is where the quantifiers are of the same type. For example, "Brothers are siblings" can be written as

$$\forall x\ \forall y\ \ Brother(x, y) \Rightarrow Sibling(x, y).$$

Example-2

“Everybody loves somebody” means that
for every person,there is someone that person loves

∀x ∃ y Loves(x,y)

Connection between ∀ and ∃

“Everyone likes icecream “ is equivalent
“there is no one who does not like ice cream”

This can be expressed as :

∀x Likes(x,IceCream) is equivalent to
¬ ∃ ¬Likes(x,IceCream)