Logical Agents
-
Knowledge Representation

Dr Babu Rao K,

Professor in CSE,

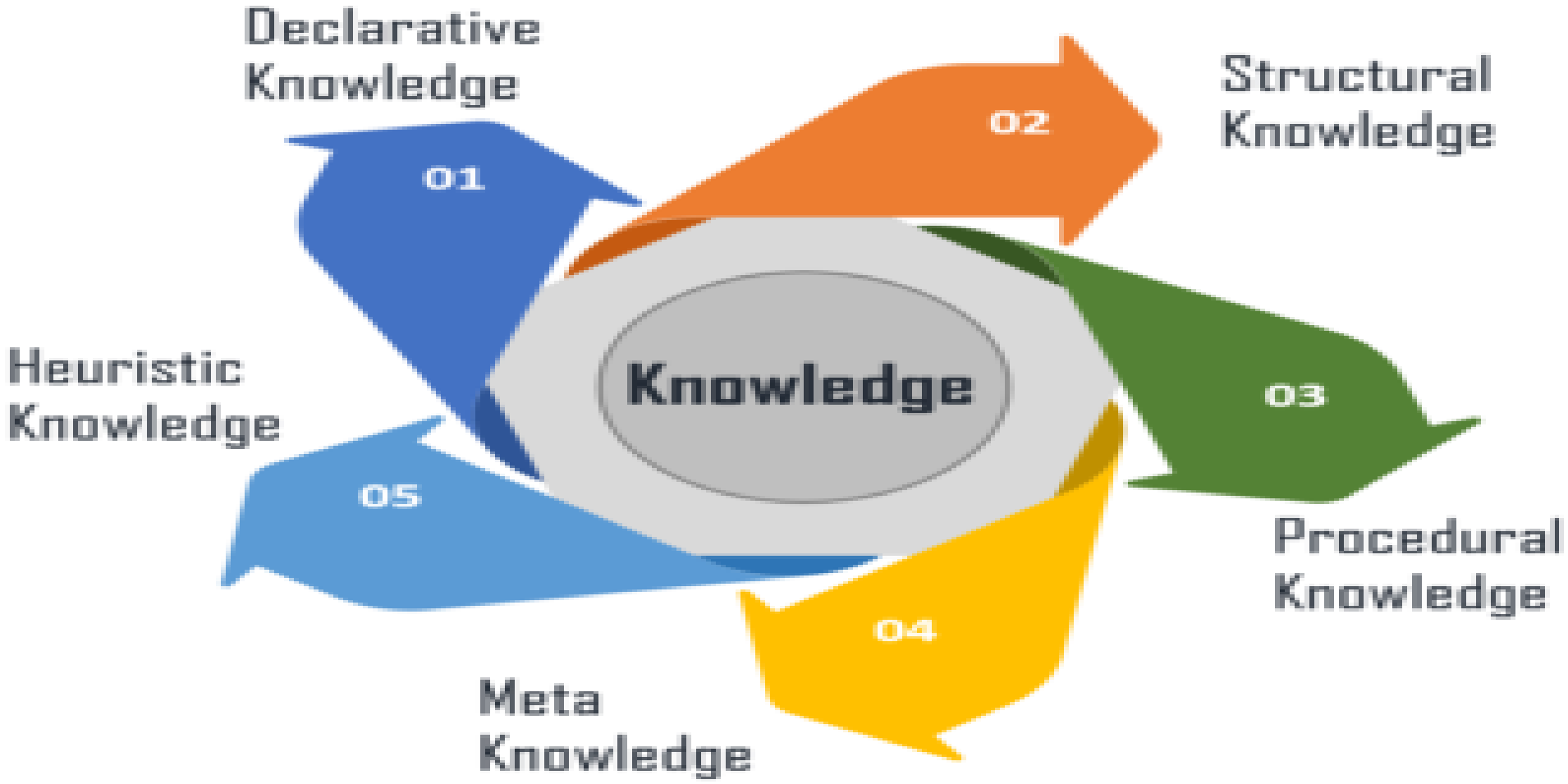
SoET, CMR University

What is Knowledge Representation?

- **Knowledge Representation** in AI describes the representation of knowledge. Basically, it is a study of how the **beliefs**, **intentions**, and **judgments** of an **intelligent agent** can be expressed suitably for automated reasoning.
- One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.
- Knowledge Representation and Reasoning (**KR, KRR**) represents information from the real world for a computer to understand and then utilize this knowledge to solve **complex real-life problems** like communicating with human beings in natural language.

- Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.
- The different kinds of knowledge that need to be represented in AI include:
  1. **Objects**
  2. **Events**
  3. **Performance**
  4. **Facts**
  5. **Meta-Knowledge**
  6. **Knowledge-base**

Different types of Knowledge.

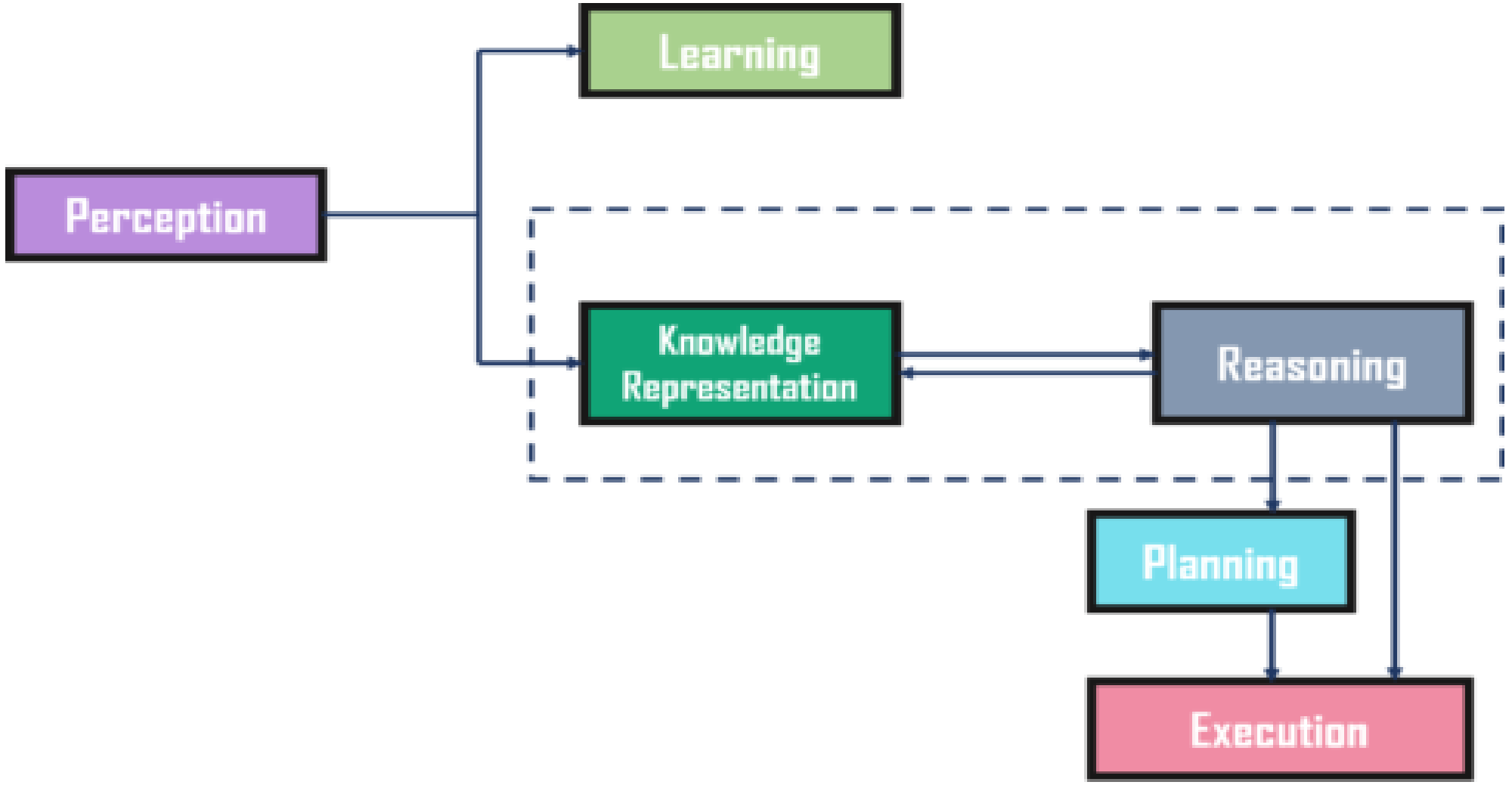


- **Declarative Knowledge** – It includes concepts, facts, and objects and expressed in a declarative sentence.
- **Structural Knowledge** – It is a basic problem-solving knowledge that describes the relationship between concepts and objects.
- **Procedural Knowledge** – This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.
- **Meta Knowledge** – Meta Knowledge defines knowledge about other types of Knowledge.
- **Heuristic Knowledge** – This represents some expert knowledge in the field or subject.

Cycle of Knowledge Representation in AI

- Artificial Intelligent Systems usually consist of various components to display their intelligent behavior.
- Some of these components include:

1. **Perception**
2. **Learning**
3. **Knowledge Representation & Reasoning**
4. **Planning**
5. **Execution**

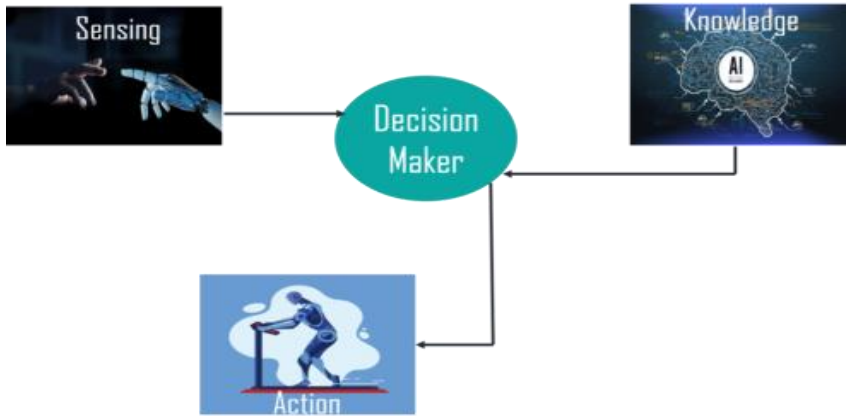

Learning
Perception
Knowledge Representation
Reasoning
Planning
Execution

- The above diagram shows the interaction of an AI system with the **real world** and the **components** involved in showing intelligence.
- The **Perception component** retrieves data or information from the environment. with the help of this component, you can retrieve data from the environment, find out the source of noises and check if the AI was damaged by anything. Also, it defines how to respond when any sense has been detected.
- Then, there is the **Learning Component** that learns from the captured data by the perception component. The goal is to build computers that can be taught instead of programming them. Learning focuses on the process of self-improvement. In order to learn new things, the system requires knowledge acquisition, inference, acquisition of heuristics, faster searches, etc.

- The main component in the cycle is **Knowledge Representation and Reasoning** which shows the human-like intelligence in the machines. Knowledge representation is all about understanding intelligence. Instead of trying to understand or build brains from the bottom up, its goal is to understand and build intelligent behavior from the top-down and focus on what an agent needs to know in order to behave intelligently. Also, it defines how automated reasoning procedures can make this knowledge available as needed.
- The **Planning and Execution** components depend on the analysis of knowledge representation and reasoning. Here, planning includes giving an initial state, finding their preconditions and effects, and a sequence of actions to achieve a state in which a particular goal holds. Now once the planning is completed, the final stage is the execution of the entire process.

What is the Relation between Knowledge & Intelligence?

- In the real world, knowledge plays a vital role in intelligence as well as creating **artificial intelligent.**
- It demonstrates the intelligent behavior in **AI agents or systems**. It is possible for an agent or system to act accurately on some input only when it has the knowledge or experience about the input.



Techniques of Knowledge Representation in AI

- There are four techniques of representing knowledge such as:

Logical Representation

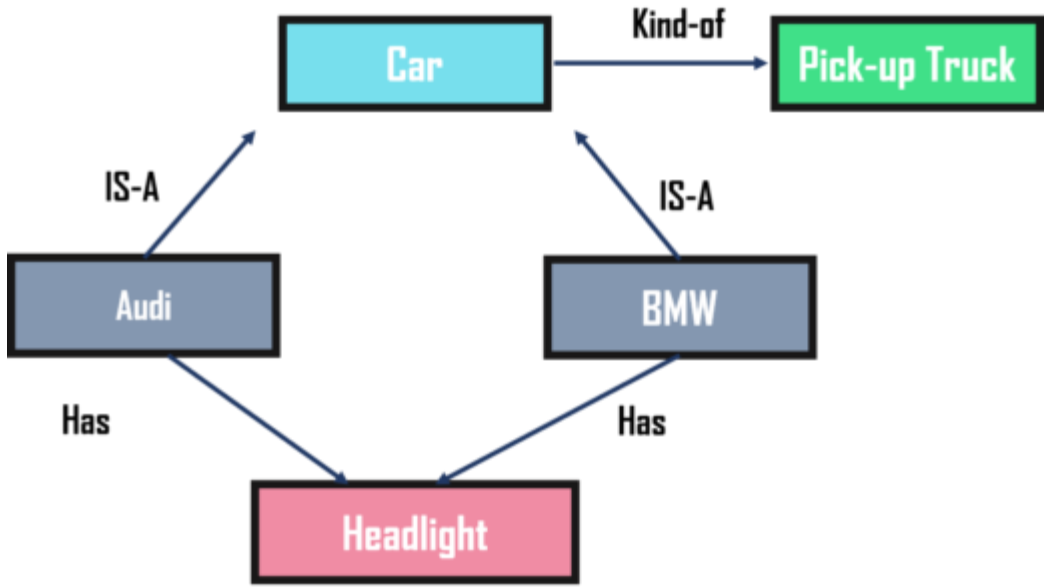
- Logical representation is a language with some **definite rules** which deal with propositions and has no ambiguity in representation.
- It represents a conclusion based on various conditions and lays down some important **communication rules**.
- Also, it consists of precisely defined syntax and semantics which supports the sound inference.
- Each sentence can be translated into logics using syntax and semantics.

Syntax	Semantics
•It decides how we can construct legal sentences in logic. •It determines which symbol we can use in knowledge representation. •Also, how to write those symbols.	•Semantics are the rules by which we can interpret the sentence in the logic. •It assigns a meaning to each sentence.

- **Advantages:**
  1. Logical representation helps to perform logical reasoning.
  2. This representation is the basis for the programming languages.
- **Disadvantages:**
  1. Logical representations have some restrictions and are challenging to work with.
  2. This technique may not be very natural, and inference may not be very efficient.

Semantic Network Representation

- Semantic networks work as an **alternative** of **predicate logic** for knowledge representation.
- In Semantic networks, you can represent your knowledge in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- Also, it categorizes the object in different forms and links those objects.
- This representation consist of two types of relations:
  - **IS-A relation (Inheritance)**
  - **Kind-of-relation**



- **Advantages:**
  - Semantic networks are a natural representation of knowledge.
  - Also, it conveys meaning in a transparent manner.
  - These networks are simple and easy to understand.
- **Disadvantages:**
  - Semantic networks take more computational time at runtime.
  - Also, these are inadequate as they do not have any equivalent quantifiers.
  - These networks are not intelligent and depend on the creator of the system.

Frame Representation

- A frame is a **record** like structure that consists of a **collection of attributes** and values to describe an entity in the world. These are the AI data structure that divides knowledge into substructures by representing stereotypes situations. Basically, it consists of a collection of slots and slot values of any type and size. Slots have names and values which are called facets.
- **Advantages:**
  - It makes the programming easier by grouping the related data.
  - Frame representation is easy to understand and visualize.
  - It is very easy to add slots for new attributes and relations.
  - Also, it is easy to include default data and search for missing values.
- **Disadvantages:**
  - In frame system inference, the mechanism cannot be easily processed.
  - The inference mechanism cannot be smoothly proceeded by frame representation.
  - It has a very generalized approach.

Production Rules

- In production rules, agent checks for the **condition** and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. Whereas, the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.
- The production rules system consists of three main parts:
  - **The set of production rules**
  - **Working Memory**
  - **The recognize-act-cycle**

- **Advantages:**
  - The production rules are expressed in natural language.
  - The production rules are highly modular and can be easily removed or modified.
- **Disadvantages:**
  - It does not exhibit any learning capabilities and does not store the result of the problem for future uses.
  - During the execution of the program, many rules may be active. Thus, rule-based production systems are inefficient.

Representation Requirements

- A good knowledge representation system must have properties such as:
- **Representational Accuracy:** It should represent all kinds of required knowledge.
- **Inferential Adequacy**: It should be able to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
- **Inferential Efficiency**: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
- **Acquisitional efficiency**: The ability to acquire new knowledge easily using automatic methods.

- They make strong claims about how the intelligence of humans is achieved—not by purely reflex mechanisms but by processes of **reasoning** that operate on internal **representations** of knowledge.
- In AI, this approach to intelligence is embodied in knowledge-based agents.
- The problem-solving agents know things, but only in a very limited, inflexible sense.
- we take this step to its logical conclusion, so to speak—we develop **logic** as a general class of representations to support knowledge-based agents.

- The central component of a knowledge-based agent is its **knowledge base**, or KB.
- A knowledge base is a set of **sentences**. (Here “sentence” is used as a technical term. It is related but not identical to the sentences of English and other natural languages.)
- Each sentence is expressed in a language called a **knowledge representation language** and represents some declaration about the world.
- Sometimes we dignify a sentence with the name **axiom**, when the sentence is taken as given without being derived from other sentences.
- 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**.

- A knowledge-based agent can be built simply by TELLing it what it needs to know. Starting with an empty knowledge base, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment.
- This is called the **declarative** approach to system building.
- In contrast, the **procedural** approach encodes desired behaviors directly as program code.

Logic

- This section summarizes the fundamental concepts of logical representation and reasoning.
- These beautiful ideas are independent of any of logic's particular forms.
- we said that knowledge bases consist of sentences. These sentences are expressed according to the **syntax** of the representation language, which specifies all the sentences that are well formed.
- The notion of syntax is clear enough in ordinary arithmetic: "x + y = 4" is a well-formed sentence, whereas "x4y+ =" is not.

- A logic must also define the **semantics** or meaning of sentences.
- The semantics defines the **truth** of each sentence with respect to each **possible world**.
- For example, the semantics for arithmetic specifies that the sentence “x + y = 4” is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.
- In standard logics, every sentence must be either true or false in each possible world—there is no “in between.”
- If a sentence α is true in model m, we say that m **satisfies** α or sometimes m **is a model of** α. We use the notation M(α) to mean the set of all models of α.

Propositional Logic

- We now present a simple but powerful logic called **propositional logic**.
- We cover the syntax of propositional logic and its semantics—the way in which the truth of sentences is determined.
- Then we look at **entailment**—the relation between a sentence and another sentence that follows from it—and see how this leads to a simple algorithm for logical inference.

- **Syntax**
- The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences** consist of a single **proposition symbol**. Each such symbol stands for a proposition that can be true or false.
- We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example: P, Q, R, W1,3 and North.
- There are two proposition symbols with fixed meanings: **True** is the always-true proposition and **False** is the always-false proposition.
- **Complex sentences** are constructed from simpler sentences, using parentheses and **logical connectives**.

- There are five connectives in common use:

¬ (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**).

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

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

⇒ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, 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 ⊃ or →.

⇔ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**. Some other books write this as ≡.

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
&\mid \neg\ \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\text{OPERATOR PRECEDENCE} &: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

- **Semantics**
- Having specified the syntax of propositional logic, we now specify its semantics.
- The semantics defines the rules for determining the truth of a sentence with respect to a particular model.
- In propositional logic, a model simply fixes the truth value—*true* or *false*—for every proposition symbol.
- For example, if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, then one possible model is

$$m1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\} .$$

With three proposition symbols, there are 23 = 8 possible

- For complex sentences, we have five rules, which hold for any subsentences P and Q in any model *m* (here “iff” means “if and only if”):
  - ❖ ¬P is true iff P is false in m.
  - ❖ P ∧ Q is true iff both P and Q are true in m.
  - ❖ P ∨ Q is true iff either P or Q is true in m.
  - ❖ P ⇒ Q is true unless P is true and Q is false in m.
  - ❖ P ⇔ Q is true iff P and Q are both true or both false in m

❖The rules can also be expressed with truth tables that specify the truth value of a complex sentence for each possible assignment of truth values to its components.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

the sentence $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$, evaluated in m_1, gives $true \wedge (false \vee true) = true \wedge true = true$. Exercise 7.3 asks you to write the algorithm PL-TRUE?(s, m), which computes the truth value of a propositional logic sentence s in a model m.

The truth tables for "and," "or," and "not" are in close accord with our intuitions about the English words. The main point of possible confusion is that $P \vee Q$ is true when P is true or Q is true *or both*. A different connective, called "exclusive or" ("xor" for short), yields false when both disjuncts are true.[7] There is no consensus on the symbol for exclusive or; some choices are $\dot{\vee}$ or $\neq$ or $\oplus$.

- Intelligent agents need knowledge about the world in order to reach good decisions.
- Knowledge is contained in agents in the form of **sentences** in a **knowledge representation language** that are stored in a **knowledge base**.
- A knowledge-based agent is composed of a knowledge base and an inference mechanism. It operates by storing sentences about the world in its knowledge base, using the inference mechanism to infer new sentences, and using these sentences to decide what action to take.
- A representation language is defined by its **syntax**, which specifies the structure of sentences, and its **semantics**, which defines the **truth** of each sentence in each **possible world** or **model**.
- The relationship of **entailment** between sentences is crucial to our understanding of reasoning. A sentence α entails another sentence β if β is true in all worlds where α is true. Equivalent definitions include the **validity** of the sentence $\alpha \Rightarrow \beta$ and the **unsatisfiability** of the sentence $\alpha \wedge \neg\beta$.
- Inference is the process of deriving new sentences from old ones. **Sound** inference algorithms derive only sentences that are entailed; **complete** algorithms derive all sentences that are entailed.
- **Propositional logic** is a simple language consisting **of proposition symbols** and **logical connectives**. It can handle propositions that are known true, known false, or completely unknown.
- The set of possible models, given a fixed propositional vocabulary, is finite, so entailment can be checked by enumerating models. Efficient **model-checking** inference algorithms for propositional logic include backtracking and local search methods and can often solve large problems quickly.

First – Order Logic

- Unfortunately, propositional logic is too puny a language to represent knowledge of complex environments in a concise way.
- **first-order logic,** which is sufficiently expressive to represent a good deal of our commonsense knowledge.
- It also either subsumes or forms the foundation of many other representation languages and has been studied intensively for many decades.
- 1st begin with a discussion of **representation languages in general**; 2nd covers the **syntax** and **semantics** of **first-order logic**; 3rd first-order logic for simple representations.