Knowledge Representation
&
Predicate Logic

Professor Dr. A K M Akhtar Hossain

Dept. of CSE, RU

Knowledge Representation

- Knowledge representation (KR) is an important issue in both cognitive science and artificial intelligence.
  - In cognitive science, it is concerned with the way people store and process information.
  - In artificial intelligence (AI), main focus is to store knowledge so that programs can process it and achieve human intelligence.
- 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**.

Knowledge Representation

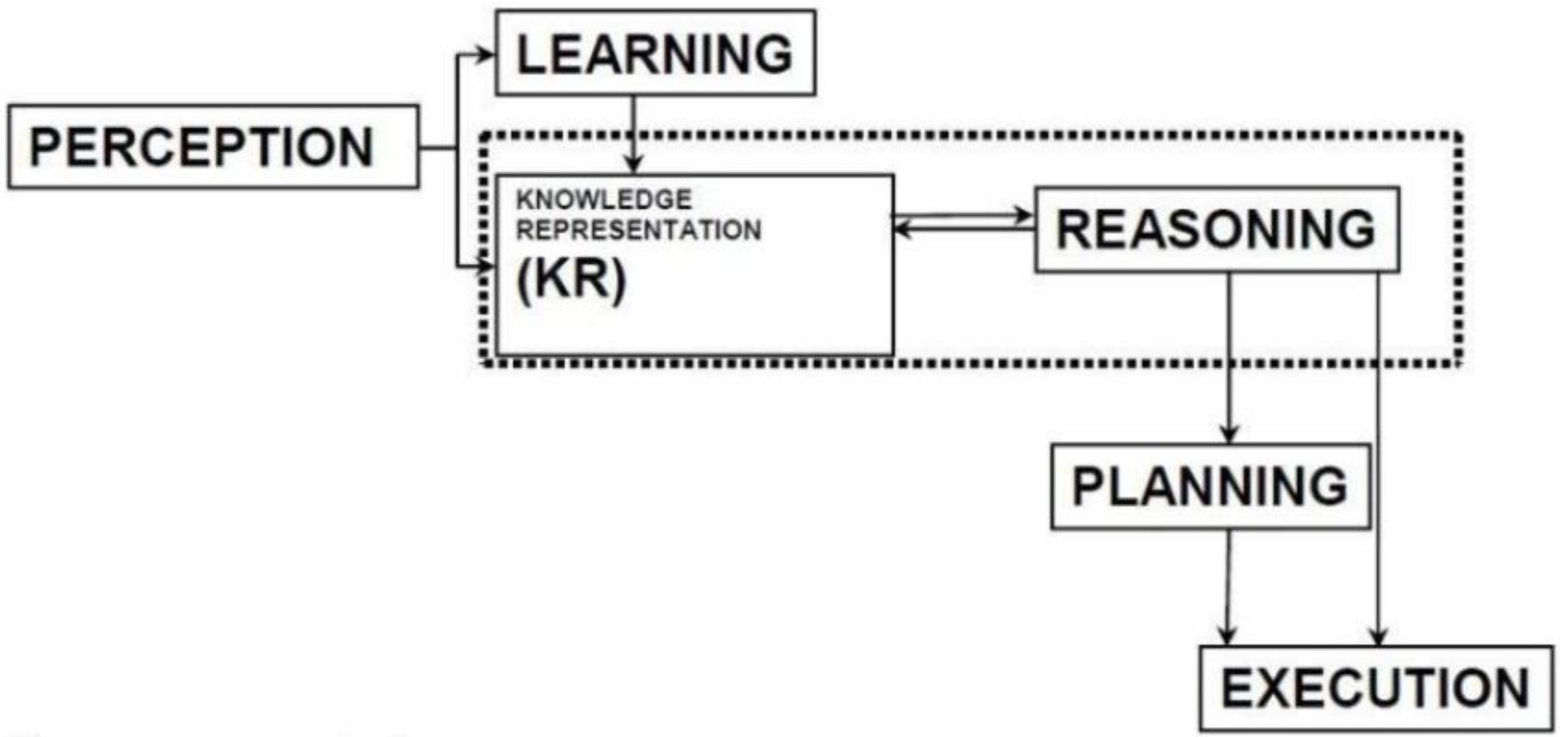

Figure: Knowledge Representation in AI

Components of knowledge Representation system

The knowledge representation function contains the following components.

- **Perception**
- **Learning**
- **KR and Reasoning**
- **Planning and Execution**

Components of knowledge Representation system

- **Perception helps in extracting the information and can be helpful in telling us the status of AI system.** It can detect any irregularity in the system and make us ready to decide whether an AI system has the potentiality of damage or not.
- **Learning component captures the data which are already sensed by the perception component.** Learning component tries to enable the computer to learn just like human instead of always programming it. This component solely tries to focus on how to self-improve the AI system.

Components of knowledge Representation system

- **KR and reasoning are used in AI to acquire knowledge in the smartest way. It focuses on the behavior of an AI agent and make sure that it more or less behaves like human.** It is used to formalize the knowledge in the knowledge base.

- **Planning and execution try to find the optimal solution of the current state and tries to understand the impact of the same.** Now it tries to seek out the solution that the final state holds and then it will try to terminate the entire process with a solution here itself.

Applications of Knowledge Representation

- Knowledge representation and reasoning are powerful tools that can be used **to build intelligent systems**.
- By representing knowledge in a computer-readable format, and by using reasoning to solve problems.
- AI applications can perform tasks that would be difficult or impossible for humans to do.

Applications of Knowledge Representation

- Knowledge representation **makes complex software easier to define and maintain than procedural code and can be used in expert system**
- **Reasoning** is the use of symbolic representations of some statements in order to derive new ones.

What is predicate logic (PL)?

- A **predicate logic** is an expression of one or more variables defined on some specific domain.
- A **predicate** with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.

Predicate logic (PL):

- A **predicate** is a statement that contains variables (**predicate** variables), and they may be true or false depending on the values of these variables.
- **Predicate logic** is the generic term for symbolic formal systems like first-order **logic**, second-order **logic**, many-sorted **logic**, or infinitary **logic**.

Predicate Logic (PL):

- The Predicate Logic has three **logical notions**.
  1) Term,
  2) Predicate
  3) Quantifier

(1). Term:

- a constant (single individual or concept i.e.,5, john etc.),
- a variable that stands for different individuals
- n-place function $\mathbf{f(t_1, \ldots, t_n)}$, where $t_1, \ldots, t_n$ are terms.
- A function is a mapping that maps **n terms** to a term.

Predicate Logic (PL):

- **Predicate:**
  - a relation that maps n terms to a truth value (T) or false value (F).

- **Quantifier:**
  - Universal (∀) quantifier
  - Existential(∃) quantifier

  - **∀ and ∃ are used for conjunction with variables.**

Examples

- "*x loves y*" is represented as **LOVE(x, y)** which maps it to true or false, when x and y get instantiated to actual values.

- "*john's father loves john*" is represented as **LOVE(*father*(john), john).**

  - Here ***father*** is a **function** that maps *john* to his father.

Examples

- ***x is greater than y*** is represented in predicate calculus as **GT(x, y)**.
- It is defined as follows:

  GT(x, y) = T , if x > y

  = F , otherwise

- Symbols like **GT** and **LOVE** are called **predicates .**

❖ **Predicates two terms and map to T or F depending upon the values of their terms.**

Examples – Cont..

- Translate the sentence "***Every man is mortal***" into Predicate formula.
- Representation of statement in predicate form
  - ❖ "**x is a man**" and "**MAN(x)**,
  - ❖ **x is mortal**" by **MORTAL(x)**
- Every man is mortal :

  $(\forall x)\ (MAN(x) \rightarrow MORTAL(x))$

Here, $\forall$x is read as "for all x" and $\rightarrow$ is read as "implies".

Syntax and semantics for Propositional Logic

- **Valid statements or sentences in PL(Predicate Logic) are determined according to the rules of propositional syntax**.
- This syntax governs the combination of basic building blocks such as propositions and logical connectives.
- **Propositions are elementary atomic sentences.**

Cont……

- **Propositional Logic may be either true or false but may take on no other value.**
- **Examples** (**Simple propositions**):
  - It is raining.
  - It is a shining day.
  - Snow is white.
  - Snow is black.
  - People live on the Earth.
  - People live on the Moon.

Cont….

- **Examples (Compound propositions):**
  - It is raining and the wind is blowing.
  - If you study hard you will be rewarded.
  - The sum of 10 and 20 is not 40.
  - The sum of 20 and 10 is 40.
- T and F are special symbols having the values true and false.

Conti…

- **Logical Connectives:**

Symbol	Meaning
~	for not or **negation**
&	for **and** or **conjunction**
v	For **or** or **disjunction**
→	For **if … then** or **implication**
↔	For **if and only if** or **double implication**

Syntax

- The syntax of PL is defined recursively as follows:
- T and F are formulas.
- IF P and Q are formulas, the following are also formulas:
- (˜P)
- (P&Q)
- (PVQ)
- (P→Q)
- (P↔Q)

Examples

- 1. Marcus was a man.
- PL: Man(Marcus)
- 2. Marcus was a Pompeian.
- PL: Pompeian(Marcus)
- 3. All Pompeian's were Romans.
- PL: $\forall x$:Pompeian(x) $\rightarrow$ Roman(x)
- 4. Caesar was a ruler.
- PL: Rular(Caesar)

Examples

- Represent the following facts in predicate logic:
- *(i). All employees earning Tk. 3,00,000/= or more per year have to pay taxes.*
- $\forall x\ ((E(x)\ \&\ GE(i(x), \mathit{300000})) \rightarrow T(x))$
- *(ii). People only try to assassinate rulers they are not loyal to.*
- $\exists y: \forall x : person(x) \land ruler(y) \land tryassassinate(x, y) \rightarrow \neg loyalto\ (x, y)$
- *(iii). John likes all kinds of food.*
- *Like (john, all-kinds-of-food)*

Semantics

- **The semantics or meaning of a sentence is just the value true or false: that is, it is as assignment of a truth value to the sentences.**
- An interpretation for a sentence or group of sentences in an assignment of the truth value to each propositional symbol.

Conti…

- **Example:** Consider the statement = (P & ~Q)
- Clearly, there are four distinct interpretations for this sentences.

Interpretation	P	Q
1	True	False
2	True	True
3	False	True
4	False	False

Semantic Rules for statements

Consider t and t' denotes true statements, f and f' denotes false statements, and a is any statement.

Rules Number	True Statements	False Statements
1.	T	F
2.	$\bar{f}$	$\bar{t}$
3.	t&t'	f&a
4.	tVa	a&f
5.	aVt	fVf'
6.	a→t	t→f
7.	f→a	t↔f
8.	t↔t'	f↔t
9.	f↔f'	

Example:

- Let ***I*** assign true to ***P*** , false to ***Q*** and false to ***R*** in statement = **$((P \& {}^{-}Q) \rightarrow R) \vee Q$**.
- **What is the meaning of the statement?**

Answer:

- Rule 2 gives **^{-}Q** as **true.**
- Rule 3 gives **$(P \& {}^{-}Q)$** as **true**.
- Rule 6 gives **$(P \& {}^{-}Q) \rightarrow R$** as **false**.
- **Rule 5** gives the statement **$((P \& {}^{-}Q) \rightarrow R) \vee Q$** value as **false**.

- **THE END**
- **THANKS**