Artificial Intelligence

Knowledge Representation

Knowledge Representation
Topics

1. **Knowledge**
2. **Knowledge Representation (KR)**
3. **KR Using Predicate Logic**
4. **KR Using Rules**

Knowledge

Knowledge is the information about a domain that can be used to solve problems in that domain.

- To solve many problems requires much knowledge, and this knowledge must be represented in the computer.
- As part of designing a program to solve problems, we must define how the knowledge will be represented.
- A **representation** of some piece of knowledge is the **internal** representation of the knowledge.
- A **representation scheme** specifies the **form** of the knowledge.
- A **knowledge base** is the representation of **all** of the knowledge that is stored by an agent.

Knowledge

An answer to the question, *"how to represent knowledge"*, requires an analysis to distinguish between knowledge "***how***" and knowledge "***that***".

■ *knowing "**how** to do something".*

e.g. "how to drive a car"

is a **Procedural knowledge***.*

■ *knowing "**that** something is true or false".*

e.g. "that is the speed limit for a car on a motorway"

is a **Declarative knowledge.**

knowledge and Representation

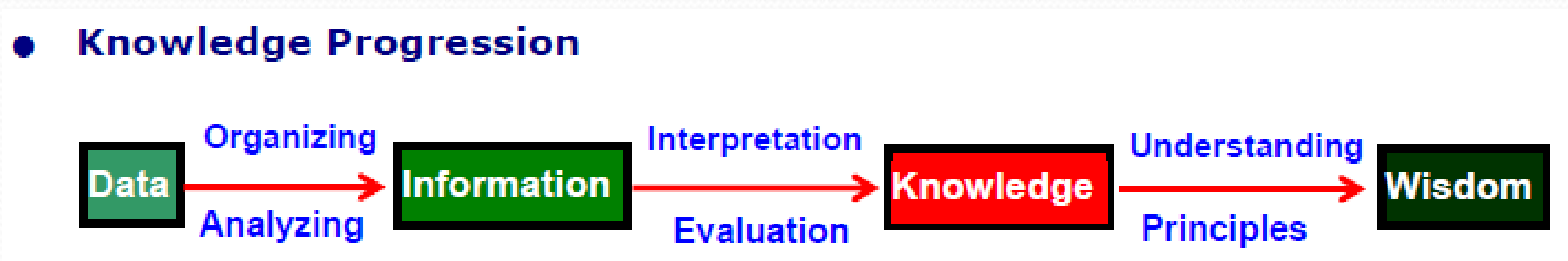
- *are two distinct entities. They play a central but distinguishable roles in intelligent system.*
- ***Knowledge is a description of the world.***

  It determines a *system's competence* by what it knows*. (Quantity)*
- ***Representation is the way knowledge is encoded.***

  It defines a *system's performance* in doing something. (Quality)
- Different types of knowledge require different kinds of representation.
- The Knowledge Representation *models/mechanisms* are often based on:
  - ◊ **Logic**
  - ◊ **Frames**
  - ◊ **Rules**
  - ◊ **Semantic Net**

1. Knowledge

- **Knowledge** is a progression that starts with ***data*** which is of limited utility.
- By organizing or analyzing the data, we understand what the data means, and this becomes ***information***.
- The interpretation or evaluation of information yield ***knowledge***.
- An understanding of the principles embodied within the knowledge is ***wisdom***.



- **Data** is viewed as collection of *disconnected facts*

Example : *It is raining.*

- **Information** emerges when *relationships among facts* are established and understood; Provides answers to *"who", "what", "where", and "when"*.

Example : *The temperature dropped 15 degrees and then it started raining.*

- **Knowledge** emerges when *relationships among patterns* are identified and understood; Provides answers as *"how"* .

Example *: If the humidity is very high and the temperature drops substantially, then atmospheres is unlikely to hold the moisture, so it rains.*

- **Wisdom** is the pinnacle of understanding, uncovers the *principles of relationships that describe patterns*. Provides answers as "*why*" .
- **Example** : *Encompasses understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients and changes.*

Knowledge Model (Bellinger 1980)

- A knowledge model tells, that as the degree of "*connectedness*" and "*understanding*" increases, we progress from *data* through *information* and *knowledge* to *wisdom.*

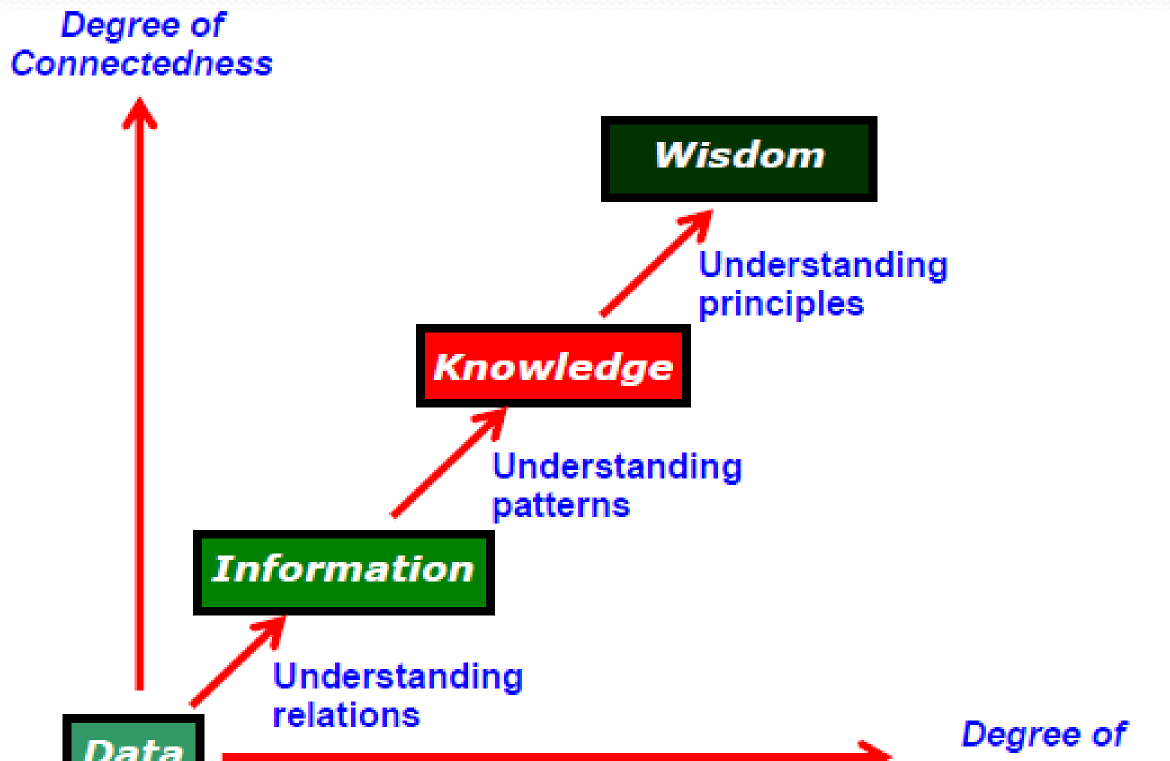
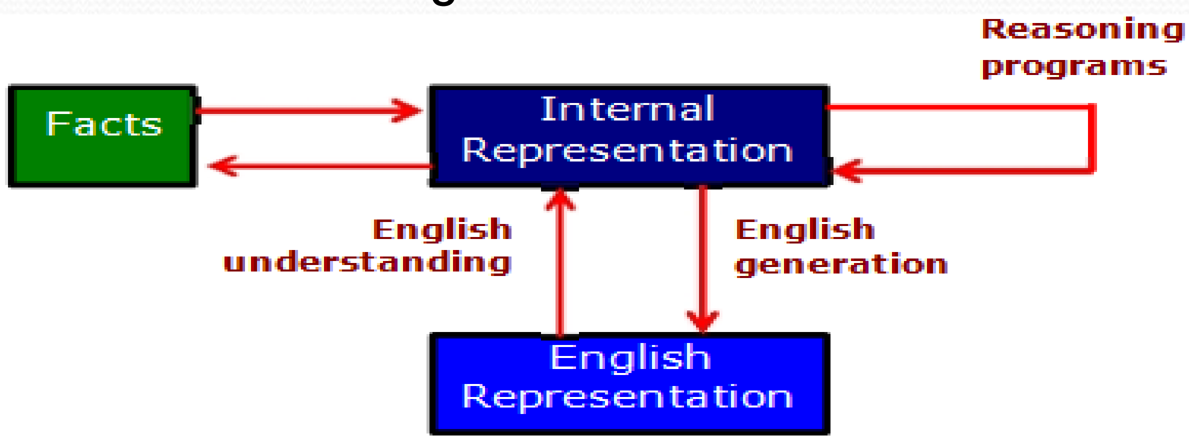


Fig. Knowledge Model

- The distinctions between *data, information, knowledge*, and *wisdom* are not very discrete.
- They are more like shades of gray, rather than black and white (Shedroff, 2001).
- "*data*" and "*information*" deal with the past; they are based on the gathering of *facts* and adding context.
- "*knowledge*" deals with the present that enable us to perform.
- "*wisdom*" deals with the future, acquire vision for what will be, rather than for what is or was.

Mapping between Facts and Representation

- Knowledge is a collection of "*facts*" from some domain.
- We need a representation of *"facts"* that can be manipulated by a program.
- Normal English is insufficient, too hard currently for a computer program to draw inferences in natural languages.
- Thus some symbolic representation is necessary.
- Therefore, we must be able to map *"facts to symbols"* and *"symbols to facts"* using *forward and backward representation mapping.*
- **Example :** Consider an English sentence



Facts	Representations
◇ **Spot is a dog**	A *fact* represented in *English sentence*
◇ **dog (Spot)**	Using *forward mapping function* the above *fact* is represented in *logic*
◇ $\forall x : dog(x) \rightarrow hastail(x)$	A *logical representation* of the *fact* that *"all dogs have tails"*

Now using ***deductive mechanism*** we can generate a new representation of object :

◇ **hastail (Spot)**	A new object representation
◇ **Spot has a tail** ***[it is new knowledge]***	Using *backward mapping function* to generate English sentence

Logic

Logic is concerned with the truth of statements about the world.
Generally each statement is either *TRUE* or *FALSE*.
Logic includes : *Syntax* , *Semantics and Inference Procedure*.

◊ **Syntax :**

Specifies the *symbols* in the language about how they can be combined to form sentences.

The facts about the world are represented as sentences in logic.

◊ **Semantic :**

Specifies how to assign a truth value to a sentence based on its *meaning* in the world.

It Specifies what facts a sentence refers to.

A fact is a claim about the world, and it may be *TRUE* or *FALSE*.

◊ **Inference Procedure :**

Specifies *methods* for computing new sentences from the existing sentences.

Logic as a KR Language

Logic is a language for reasoning, a collection of rules used while doing logical reasoning.

Logic is studied as KR language in artificial intelligence.

◊ **Logic** is a formal system in which the formulas or sentences have true or false values.

◊ **Logics are of different types :** *Propositional logic, Predicate logic, Temporal logic, Modal logic, Description logic* etc;

They represent things and allow more or less efficient inference.

◊ **Propositional logic and Predicate logic are fundamental to all logic.**

- ***Propositional Logic*** is the study of statements and their connectivity.
- ***Predicate Logic*** is the study of individuals and their properties.

2.1 Logic Representation

Logic can be used to represent simple facts.

The *facts* are claims about the world that are *True* or *False*.

To build a Logic-based representation :

- ◊ **User** defines a set of primitive *symbols* and the associated *semantics.*
- ◊ **Logic** defines ways of putting symbols together so that user can define legal *sentences* in the language that represent *TRUE* **facts**.
- ◊ **Logic** defines ways of inferring *new sentences* from existing ones.
- ◊ **Sentences** - either *TRUE* or *false* but not both are called ***propositions****.*
- ◊ A declarative sentence expresses a *statement* with a proposition as content;

example: the declarative "**snow is white**" expresses that **snow is white**;

further, "**snow is white**" expresses that *snow is white* is **TRUE**.

Propositional Logic (PL)

- A **proposition** is a sentence, written in a language, that has a truth value (i.e., it is true or false) in a world.
- A proposition is built from atomic propositions using logical connectives.
- An **atomic proposition**, (or **atom)**, is a Symbol that starts with a lower-case letter.
- an atom is something that is true or false.
- For example:
  - *sunny*
  - *AI_is_fun*,
  - *lit_l*$_1$,
  - *live_outside*

Propositional Logic (PL)

- A proposition is a statement, which in English would be a declarative sentence.
- Every proposition is either TRUE or FALSE but not Both.

Examples:

(a) The sky is blue.

(b) Snow is cold.

(c) 12 * 12=144

‡ A sentence is smallest unit in propositional logic.

‡ If proposition is true, then truth value is "true" .

‡ If proposition is false, then truth value is "false" .

Example :

Sentence	Truth value	Proposition (Y/N)
"Grass is green"	"true"	Yes
"2 + 5 = 5"	"false"	Yes
"Close the door"	-	No
"Is it hot out side ?"	-	No
"x > 2" where x is variable	-	No (since x is not defined)
"x = x"	-	No (don't know what is "x" and "="; "3 = 3" or "air is equal to air" or "Water is equal to water" has no meaning)

- Propositions can be built from simpler propositions using logical connectives.
- A **proposition** is either an atomic proposition or a **compound proposition** of the form:
- $\neg p$ (read "not *p*") --the **negation** of *p*
- $p \wedge q$ (read "*p* and *q*")--the **conjunction** of *p* and *q*
- $p \vee q$ (read "*p* or *q*")--the **disjunction** of *p* and *q*
- $p \rightarrow q$ (read "*p* implies *q*")--the **implication** of *q* from *p*
- $p \leftarrow q$ (read "*p* if *q*")--the **implication** of *p* from *q*
- $p \leftrightarrow q$ (read "*p* if and only if *q*" or "*p* is equivalent to *q*")
  - where *p* and *q* are propositions.

Propositional Logic Terms

- **Statement**
  - *Simple* statements (sentences), TRUE or FALSE, that does not contain any other statement as a part, are ***basic propositions***;
- **Symbols**
  - lower-case letters, **p, q, r**, are symbols for simple statements (***basic propositions)***.
  - *Large*, *compound or complex* statement are constructed from basic propositions by combining them with ***connectives***.

Connective or Operator

The connectives join simple statements into compounds, and joins compounds into larger compounds.

Example of a formula : $((((a \wedge \neg b) \vee c \rightarrow d) \leftrightarrow \neg (a \vee c))$

Connectives and Symbols in decreasing order of operation priority

Connective	*Symbols*					**Read as**
assertion	P					**"p is true"**
negation	$\neg p$	~	!		NOT	**"p is false"**
conjunction	$p \wedge q$	·	&&	&	AND	"both p and q are true"
disjunction	P v q	||	|		OR	"either p is true, or q is true, or both "
implication	$p \rightarrow q$	$\supset$	$\Rightarrow$		if ..then	**"if p is true, then q is true"** **" p implies q "**
equivalence	$\leftrightarrow$	$\equiv$	$\Leftrightarrow$		if and only if	"p and q are either both true or both false"

Note : The propositions and connectives are the basic elements of propositional logic.

Truth Value

- The truth value of a statement is its TRUTH or FALSITY
- Example :

p is either *TRUE* or *FALSE*,

~p is either *TRUE* or *FALSE*,

p v q is either *TRUE* or *FALSE*, and so on.

use " **T** " or " **1** " to mean *TRUE*.

use " **F** " or " **0** " to mean *FALSE*

***Truth table* defining the basic *connectives* :**

p	q	¬p	¬q	p ∧ q	p v q	p→q	p ↔ q	q→p
T	T	F	F	T	T	T	T	T
T	F	F	T	F	T	F	F	T
F	T	T	F	F	T	T	F	F
F	F	T	T	F	F	T	T	T

p → **q** is equivalent to **¬p** ∨ **q**

❑ Tautologies

A proposition that is always true is called a "tautology". e.g.,

$(P \lor \neg P)$ is always true regardless of the truth value of the proposition **P**

❑ Contradictions

A proposition that is always false is called a "contradiction". e.g.,

$(P \land \neg P)$ is always false regardless of the truth value of the proposition **P**.

❑ Contingencies

A proposition is called a "contingency" , if that proposition is neither a *tautology* nor a *contradiction* . e.g., $(P \lor Q)$ is a contingency.

❑ Antecedent, Consequent

These two are parts of conditional statements.

In the conditional statements, $p \rightarrow q$, the

1st statement or "**if - clause**" (here **p**) is called **antecedent** ,

2nd statement or "**then - clause**" (here **q**) is called **consequent.**

Argument

An argument is a demonstration or a proof of some statements.

Example : **"That bird is a crow ; therefore, it's black.“**

- Any argument can be expressed as a compound statement.
- In logic, an argument is a set of one or more meaningful declarative sentences (or "propositions") known as the *premise* along with another meaningful declarative sentence (or "proposition") known as the *conclusion.*
- **Premise** is a proposition which gives reasons, grounds, or evidence for accepting some other proposition, called the conclusion.
- **Conclusion** is a proposition, which is purported to be established on the basis of other propositions (Premises).
- Take all the premises, conjoin them, as the antecedent of argument , and the conclusion as the consequent.
- This implication statement is :

If (premise1 ∧ premise2) -> conclusion