Artificial intelligent (AI)

John mecarthy coined the term of Artificial intelligent in 1956 .

Human intelligence:-

1. Ability to learn .
2. Ability to solve problem .
3. Ability to think , plan and schedule .
4. Ability to memorize .
5. Ability to recognize .
6. Ability to understand and perceive **.**

Artificial intelligent definition :-

The study of how to make computers do things at the moment people do better .

Tasks / application of Artificial intelligent :-

1. **Common tasks :-**
    - **Perception (vision , speech , etc).**
    - **Natural language (understand , generate , translation) .**
    - **Common sense and reasoning .**
2. **Formal application / tasks**
    - **Games .**
    - **Mathematics .**
3. **Expert tasks**
    - **Engineering .**
    - **Medical diagnosis .**
    - **Scientific analysis .**

Knowledge Representation :- (KR)

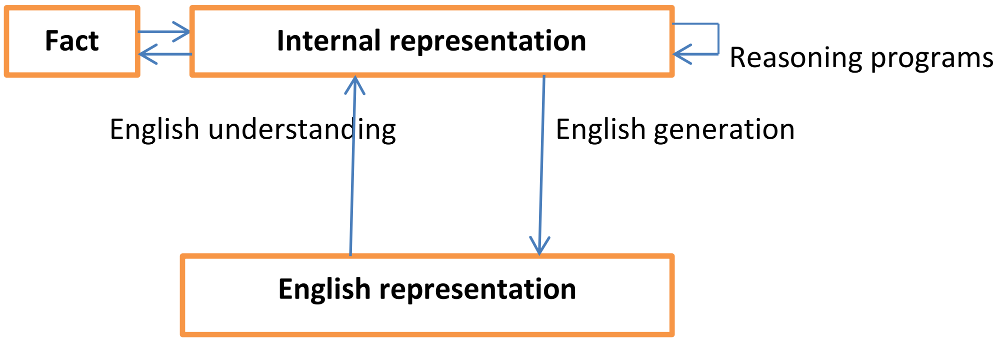
We represent object ,events (kind of actions), performance (how a particular object performs the events), meta Knowledge (Knowledge about Knowledge).

KR involves dealing with two entities :-

1. **Fact :-**

    Truth about real world [knowledge level].
2. **Representation of fact :-**

    When we manipulate [symbol level] .



<u>Note</u> :-

1. English or natural language is obvious way to represent fact , the representation is not one to one , it is one to many .

Ex :-

Each dog tail → all dogs have tail

→ Every dogs has a tail

2. When an Artificial intelligent program manipulates internal representation of fact these new representation should also be interpretable as new representation of facts .

Issues corresponding of KR :-

1. **Important attributes :-**

   Are these any important attributes that occur in many different types of problems .

   There are two important attributes

   - Instance .
   - Is a .

   And each is important because each support property of inheritance .

2. **Relationships :-**

   There are different types of relationships among attributes

   - Inverses :- A related to B

     Then B related to A
   - Existence :- attribute exist or not .

   - Techniques for reasoning about values

     Ex :- age [0-100]

- Single valued attributes
  Ex :- age is single value , phone is multi values

3. **Granularity :-**
   The level up to which knowledge represented , at what level should the knowledge be represented and a what are the primitives choosing the granularity of representation primitives are fundamental concepts such as holding , seeing , playing . and as English language is very rich language with over half million words , it is clears will find difficulty in deciding upon which words to choose as out primitives in a series of situation .

Ex :- if tom feeds a dog

Feeds(dog , tom)

If tom gives the dog a bone

Gives (tom , dog , bone)

<u>Note</u> :- if 2 statements are same meaning .

Ex :- if gives (x , food) $\longrightarrow$ feed (x)

Then we can use the first statement .

Knowledge Representation methods :-

1. Predicate logic
2. Sematic nets
3. Frames

1. Predicate logic :-

The Predicate logic consist of 3 components :-

1. **Terms :-** it can be
   - Constants
   - Variables
   - Function involving constants and variables
2. **Predicates :-**
   A declarative sentence involving terms .
3. **Quantifier :-**
   - **Universal quantifier :-**
     ∀ for all

- **Existential quantifier :-**

  ∃ there exist

Ex :-

Every man is mortal

X is a man $\longrightarrow$ man(x)

X is mortal $\longrightarrow$ mortal (x)

$\forall \times$ [ma(x) $\longrightarrow$ mortal (x)]

<u>Note</u> :-

A predicate having n variables is called n-place predicate .

Ex:- v ={3,4,5}

$P(x) := x+2 \leq 6$

Sol :-

- $\forall \times p(x)$ is false [that means all true or every x p(x) is true] .
- $\exists x\, p(x)$ is true [some true or some x p(x) is true] .

Proposition :-

Declarative sentence either true or false but cannot be both .

Proposition can be combined using connectives to form large propositions .

Connectives:-

$\wedge$ ...and [conjunction] ny

$\vee$...or [disjunction]

$\Longrightarrow$...implies [implication / conditional]

$\Longleftrightarrow$....is equivalent [biconditional]

$\neg$...not [negation]

Examples:

Ali is a brave man

This car has 4 wheels

If weather is cold then it is winter

P $\qquad\qquad\qquad$ Q

$P \Rightarrow Q$

Condition evident or conclusion

General form :-

$\neg P$

$P \wedge Q$

$P \vee Q$

$P \Rightarrow Q$

$P \Leftrightarrow Q$

Some important equivalent :-

Double negation law $\neg(\neg P) \equiv P$

Implication law $(P \vee Q) \equiv (\neg P \Rightarrow Q)$

The contra positive law : $(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$

De morgan's law : $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$ and

$(P \wedge Q) \equiv (\neg P \vee \neg Q)$

The commutative laws : $(P \wedge Q) \equiv (Q \wedge P)$ and

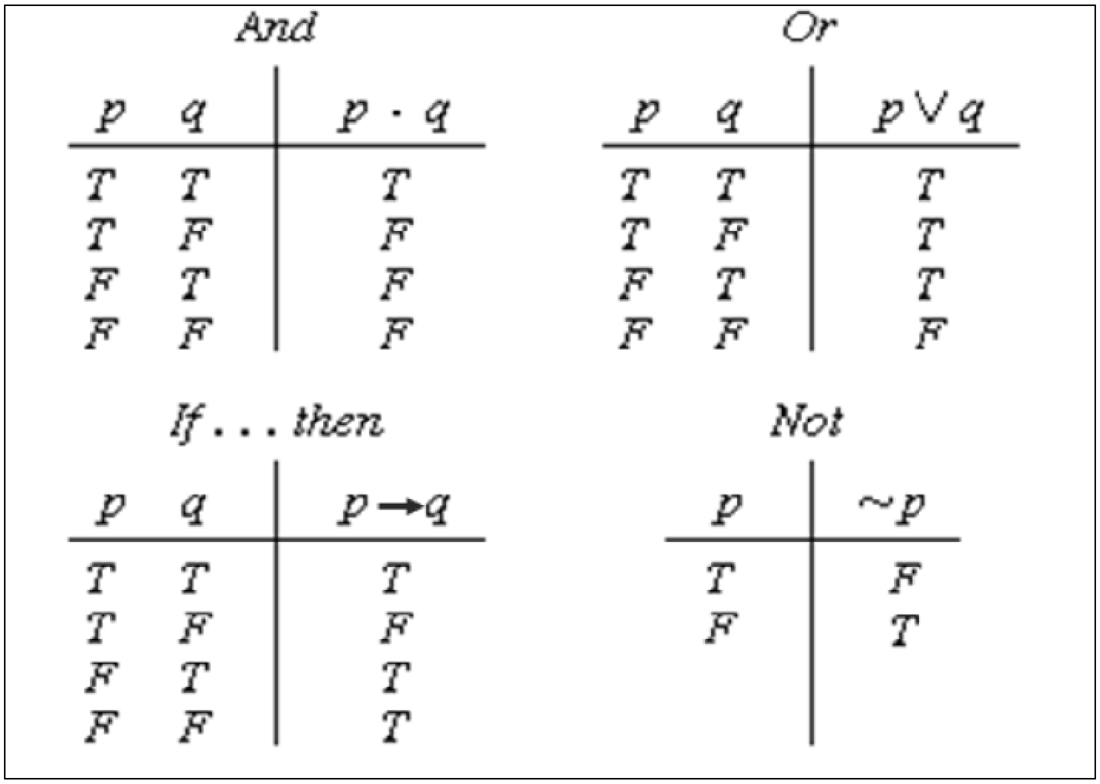
$(P \vee Q) \equiv (Q \vee P)$

Associative law : $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$

Associative law : $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$

Distributive law : $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

Distributive law : $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$



And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

Examples

(P ∧ Q) ∨(┐Q ∨ P)

P	Q	P∧Q	┐Q	┐Q∨P	(P∧Q)∨(┐Q∨P)
T	T	T	F	T	T
T	F	F	T	T	T
F	T	F	F	F	F
F	F	F	T	T	T

A→B ≡ ┐A∨B

A	B	A→B	┐A	┐A∨B
T	T	T	F	T

T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Examples of English sentences represented in predicate logic:

1- If it doesn't rain tomorrow, Tom will go to the mountains.

¬ weather (rain, tomorrow) ⟹go(tom, mountains).

2- Emma is a Doberman pinscher and a good dog.

Good dog (emma) ∧ isa (emma, Doberman)

3. All basketball players are tall.

∀ X (basketball _ player(X) ⟹tall (X))

4- Some people like anchovies.

∃ X (person(X) ∧ likes(X, anchovies)),

5- If wishes were horses, beggars would ride.

Equal (wishes , horses) ⟹ride(beggars).

6- Nobody likes taxes

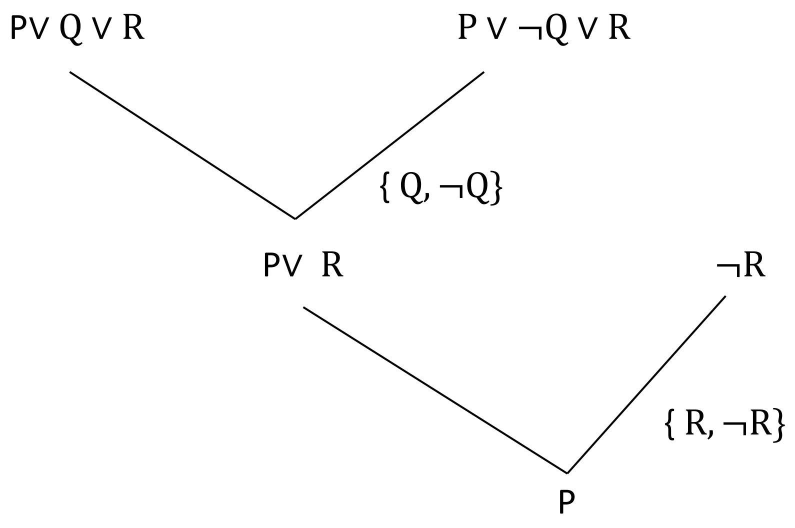
¬ ∃ X likes(X,taxes).

Resolution :-

Resolution is a technique for theorem. proving in propositional and predicate logic which attempts to show that the negation of the statement produces a contradiction with the known statements.

In the following expression, uppercase letters indicate variables (W, X, Y, and Z); lowercase letters in the middle of the alphabet indicate Constants or bound variables (I, m, and n); and early alphabetic lowercase letters indicate the predicate names (a, b, c, d, and e). To improve readability of the expressions, we use two types of brackets: ( ) and [ ]. Where possible in the derivation, we remove redundant brackets.

Ex :- find resolution for { P∨ Q ∨ R , P ∨ ¬Q ∨ R , ¬R }



Example :-

"All people who are not poor and are smart are happy. Those people who' read are not stupid. John can read is wealthy. Happy people have exciting lives. Can anyone be found with an exciting life?"

a) First change the sentences to predicate form:

We assume ∀X (smart (X) ≡ stupid (X) and ∀Y (wealthy (Y) ≡ poor (Y)), and get:

∀X(¬poor(x) ∧ smart (X) → (happy (x))

∀Y(read(Y) → smart(Y))

read (john) ∧ ¬ poor (john)

∀Z (happy (Z) → exciting (Z))

The negation of the conclusion is:

¬∃W (exciting (W))

b) These predicate calculus expressions for the happy life problem are transformed into the following clauses:

poor (X) ∨ ¬ smart (X) ∨ happy (X)

¬ read (Y) ∨smart(Y)

read (john)

¬ poor (john)

¬ happy (Z) ∨ exciting (Z)

¬ exciting(W)

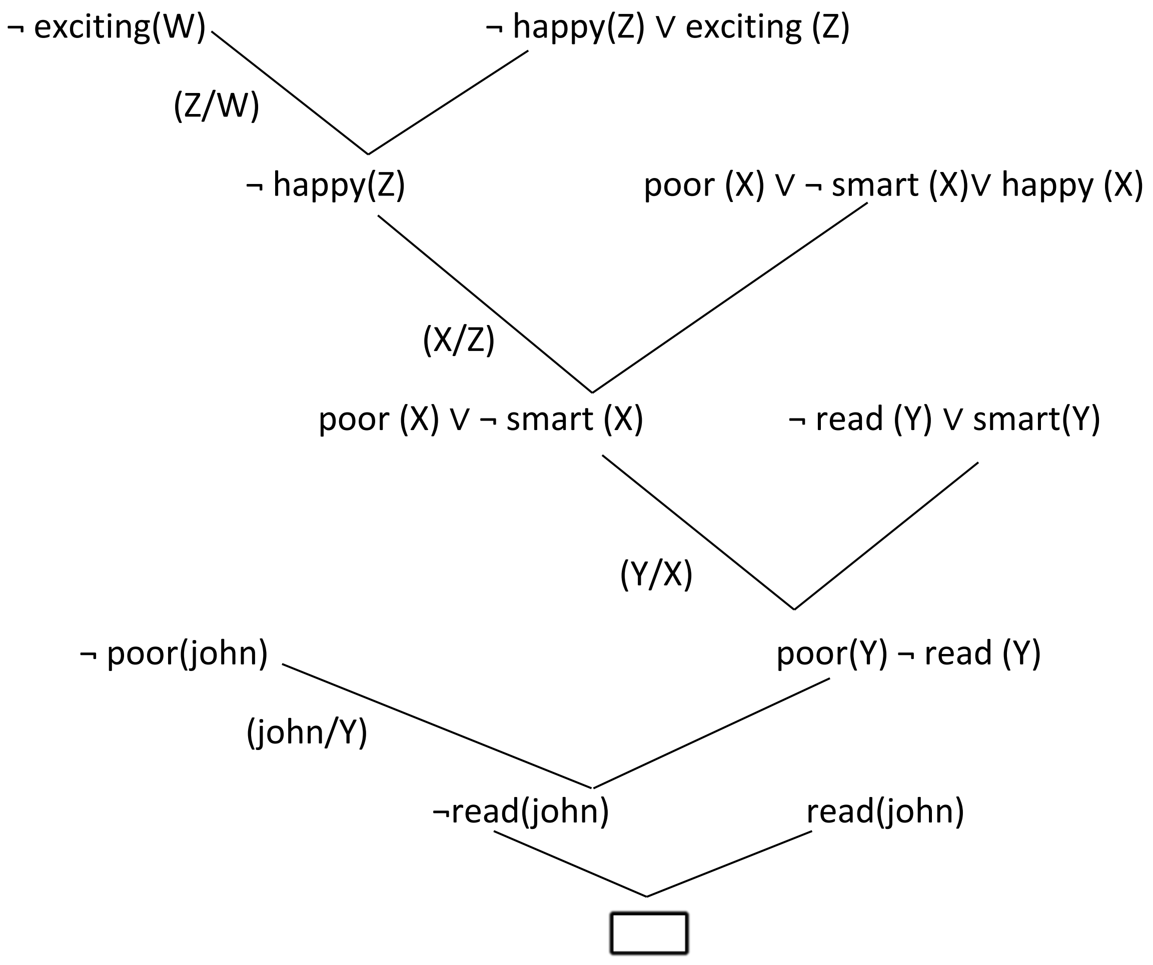


Figure (3-4):Resolution prove for the "exciting life" problem

Application resolution principle , we use **Conjunctive Normal Form (CNF)**

- Conjunction of disjunction

$(p \lor q) \land (\neg p \lor q)$

A variable or its negation is called literal

Conjunctive Normal Form (CNF)

1- Take literal
2- Take disjunction of literal
3- Take Conjunction of disjunction

Ex :- find resolve for $\{ p \rightarrow q \; , \; q \rightarrow r \}$

$\neg p \lor q \quad , \quad \neg q \lor r$

$$\begin{tikzcd}
(\neg p \lor q) \arrow[dr, dash] & \land & (\neg q \lor r) \arrow[dl, dash, "{\{ q \, , \, \neg q \}}"] \\
& \neg p \lor r &
\end{tikzcd}$$

Production Rules for Knowledge Representation :-

A production system (or production rule system) is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior. These rules, termed productions, are a basic representation found useful in automated planning, expert systems and action selection. A production system provides the mechanism necessary to execute productions in order to achieve some goal for the system.

Productions consist of two parts: a sensory precondition (or "IF" statement) and an action (or "THEN"). If a production's precondition matches the current state of the world, then the production is said to be triggered. If a production's action is executed, it is said to have fired. A production system also contains a database, sometimes called working memory, which maintains data about current state or knowledge, and a rule interpreter. The rule interpreter must provide a mechanism for prioritizing productions when more than one is triggered.

One of the most popular approaches to knowledge representation is to use production rules, sometimes called IF-THEN rules. They can take various forms.e.g.

```
IF condition THEN action

IF premise   THEN conclusion

IF proposition p1 and proposition p2 are true

  THEN proposition p3 is true
```

Some of the benefits of IF-THEN rules are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. New rules may be added and old ones deleted usually independently of other rules.

A rule based system will contain global rules and facts about the knowledge domain covered. During a particular run of the system a database of local knowledge may also be established, relating to the particular case in hand. One of the most widely used tutorial examples of rule based systems is Mycin, an expert system which was designed to assist doctors with the diagnosis and treatment of bacterial infection. It uses the rule based approach and also demonstrates the way in which uncertainty (both in observations and in the reasoning process) may be handled.

Mycin was designed to help the doctor to decide whether a patient has a bacterial infection, which organism is responsible, which drug may be appropriate for this infection, and which may be used on the specific patient.

The global knowledge base contains facts and rules relating for example symptoms to infections, and the local database will contain particular observations about the patient being examined. A typical rule in Mycin is as follows:

```
IF the identity of the germ is not known with certainty

   AND the germ is gram-positive

   AND the morphology of the organism is "rod"

   AND the germ is aerobic

THEN there is a strong probability (0.8) that the germ is of type enterobacteriaceae
```

Note that a probability or certainty factor (C.F.) is given, reflecting the strength of the original expert's confidence in the inference made in this rule. In other words, the confidence in the conclusion assuming the premises are true. The premises are, in fact, established from observations either in the laboratory or from the patient, and may themselves have an element of uncertainty associated with them. In the above example it may only be known that the germ is aerobic with a probability of 0.5.

The certainty factor associated with a conclusion in MYCIN is calculated from the certainty factor of the premises, the certainty factor of the rule and any existing certainty factors for the conclusion if it has been obtained already from some other rules.

The way in which the knowledge base is used is determined by the inference engine. It is a basic principle of production systems that each rule should be an independent item of knowledge and essentially ignorant of other rules. The inference engine could then simply "fire" rules at any time when its premises are satisfied.

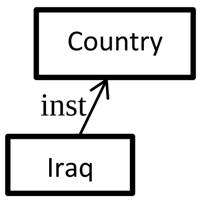
If several rules could all fire at once the inference engine must have a mechanism for "conflict resolution". This may be achieved, for example, by having some predefined order, perhaps on the basis of the strength of the conclusion, or alternatively on the basis of frequency of rule usage.

Forward and Backward chaining through the rules may be used. The two systems each have their advantages and disadvantages and in fact answer different types of question. For example, in Mycin a forward chaining system might answer the question "what do these symptoms suggest?" whereas a backward chaining system might answer the question "does this patient suffer from a pelvic abscess?" In general, rules and goals may need to be constructed differently for forward and backward chaining systems.

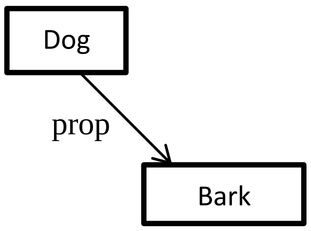
2. Semantic Net

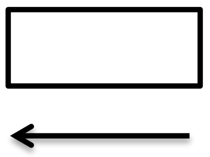
It is consist of a set of nodes and arcs , each node is represented as a rectangle to describe the objects, the concepts and the events. The arcs are used to connect the nodes and they divided to three parts:-

1. Is a: $\longrightarrow$ subclass of entity.
2. Inst $\longrightarrow$ particular instance of a class



3. Prop $\longrightarrow$ property link





تمثيل الأفعال والأحداث و الكائنات

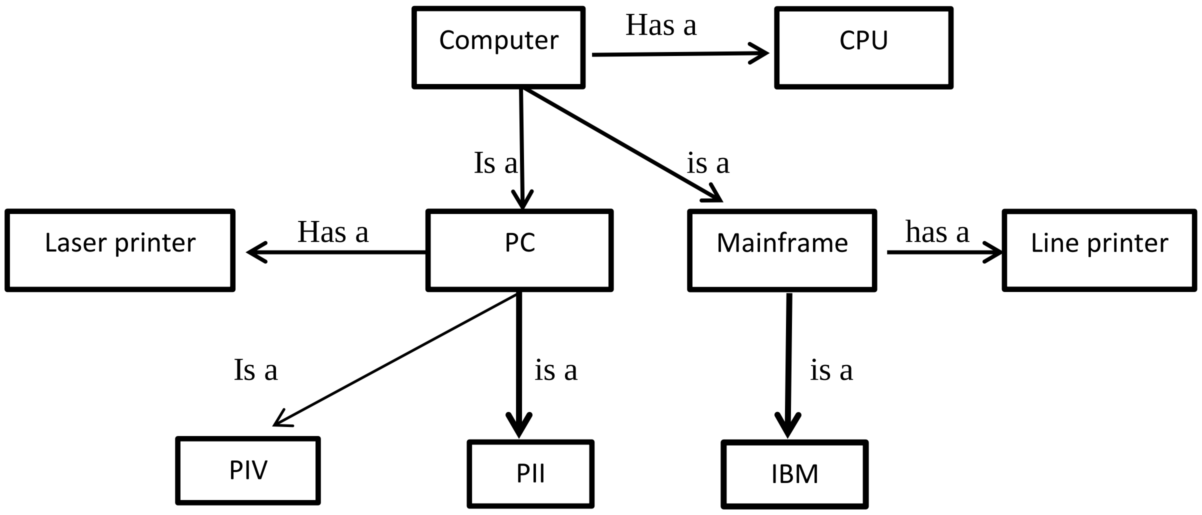
لتمثيل العلاقة بين الكائنات

في وصف اللغات الطبيعية فأن arcs تخرج من الفعل توضح او تشير الى الفاعل (agent)

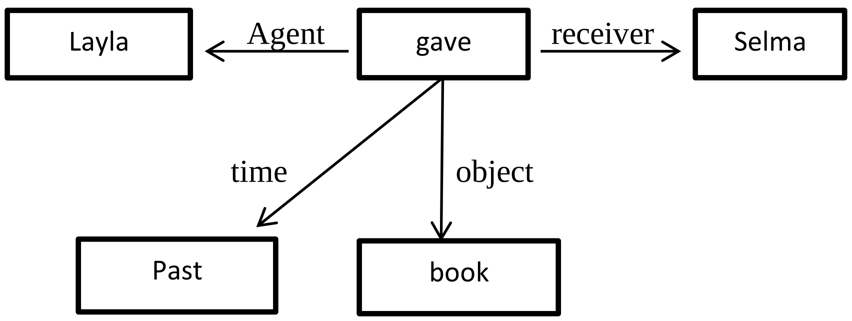
والمستقبل (Receiver) والكائن (object) كما تشير الى وقت حدوث الفعل أي في الماضي , الحاضر او المستقبل.

Example1:

Computer has many part like a CPU and the computer divided into two type, the first one is the mainframe and the second is the personal computer ,Mainframe has line printer with large sheet but the personal computer has laser printer , IBM as example to the mainframe and PIII and PIV as example to the personal computer.



Example 2: Layla gave Selma a book



3. Frames :

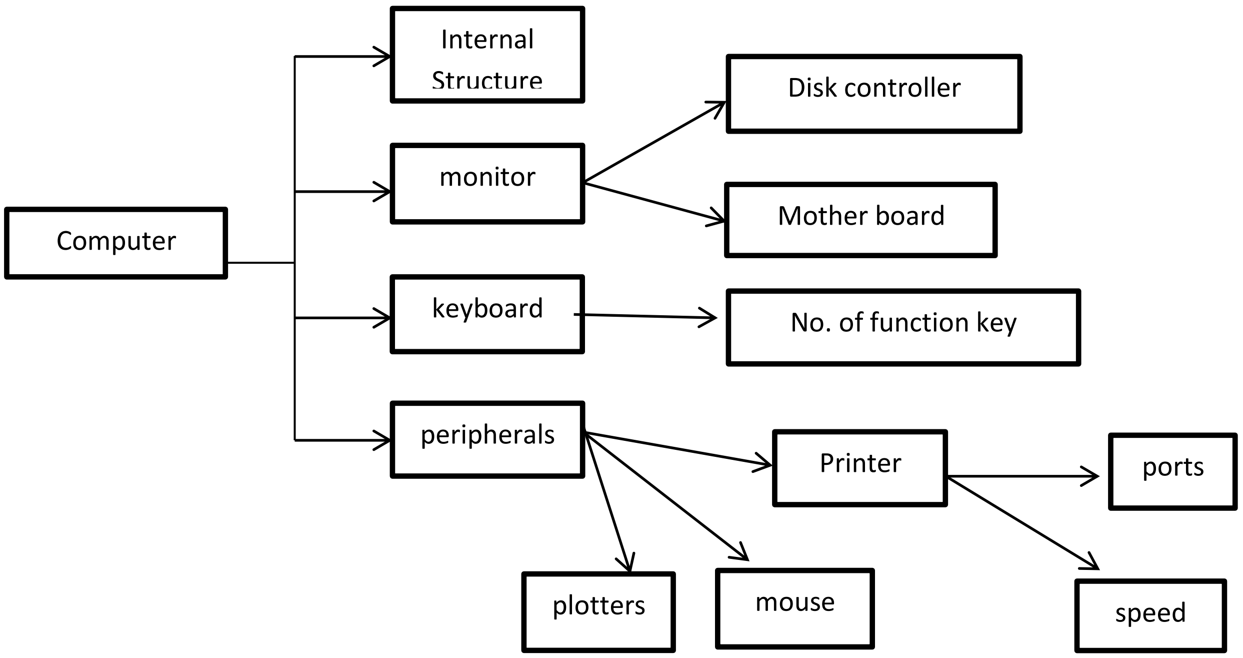
Frames are more structured form of backing knowledge . Frames are organized into hierarchies or network of frames . lower level frames inherit information from upper level . nodes (frames) are connected using links .

1. Ako (a kind of object) .
2. Is-a / inst (connect a particular instance to a class frame) .
3. A-part-of (connect frames one of which in a part of other) .

Frames has a lot of slot , each slot has number of facets , facet may be value , default , range , etc.

Frame-list(node-name, parent, [child]).

Slot-list(node-name, parent).



Frame –list(computer,_ ,[Internal structure, monitor, keyboard , plotters]).

Frame-list(Internal structure, computer, [disk controller, mother board]).

Frame- list(printer, peripheral, [speed, ports]).

Slot-list(motherboard, Internal structure).

Slot-list(mouse, peripheral).

Scripts :-

A **script** is a structure that prescribes a set of circumstances which could be expected to follow on from one another.

It is similar to a thought sequence or a chain of situations which could be anticipated.

It could be considered to consist of a number of slots or frames but with more specialised roles.

Scripts are beneficial because:

- Events tend to occur in known runs or patterns.
- Causal relationships between events exist.
- Entry conditions exist which allow an event to take place
- Prerequisites exist upon events taking place. E.g. when a student progresses through a degree scheme or when a purchaser buys a house.

The components of a script include:

1. Entry Conditions :-

   these must be satisfied before events in the script can occur.

2. Results :-

   Conditions that will be true after events in script occur.

3. Props :-

   Slots representing objects involved in events.

4. Roles :-

   Persons involved in the events.

5. Track :-

Variations on the script. Different tracks may share components of the same script.

6. Scenes :-

The sequence of events that occur. Events are represented in conceptual dependency form.

Scripts are useful in describing certain situations such as robbing a bank. This might involve:

- Getting a gun.
- Hold up a bank.
- Escape with the money.

Advantages of Scripts:

- Ability to predict events.
- A single coherent interpretation may be build up from a collection of observations.

Disadvantages:

- Less general than frames.
- May not be suitable to represent all kinds of knowledge.

Expert Systems

Expert systems are computer programs that are constructed to do the kinds of activities that human experts can do such as design, compose, plan, diagnose, interpret, summarize, audit, give advice.
The work such a system is concerned with is typically a task from the worlds of business or engineering/science or government.
Expert system programs are usually set up to operate in a manner that will be perceived as intelligent: that is, as if there were a human expert on the other side of the video terminal.
Expert systems generally use automated reasoning and the so-called weak methods, such as search or heuristics, to do their work.

Expert System Using

Here is a short nonexhaustive list of some of the things expert
systems have been used for:
• To approve loan applications, evaluate insurance risks, and evaluate investment opportunities for the financial community.

• To help chemists find the proper sequence of reactions to create new molecules.
• To configure the hardware and software in a computer to match the unique arrangements specified by individual customers.

• To diagnose and locate faults in a telephone network from tests and trouble reports.
• To identify and correct malfunctions in locomotives.
• To help geologists interpret the data from instrumentation at the drill tip during oil well drilling.

• To help physicians diagnose and treat related groups of diseases, such as infections of the blood or the different kinds of cancers.
• To help navies interpret hydrophone data from arrays of microphones on the ocean floor that are used t\u the surveillance of ships in the vicinity.
• To figure out a chemical compound's molecular structure from experiments with mass spectral data and nuclear magnetic resonance.
• To examine and summarize volumes of rapidly changing data that are generated too last for human scrutiny, such as telemetry data from Landsat satellites.

Expert System Architecture and Components

The architecture of the expert system consists of several components

1. **User Interface**

   The user interacts with the expert system through a user interface that make access more comfortable for the human and hides much of the system complexity. The interface styles includes questions and answers, menu-driver, natural languages, or graphics interfaces.

2. Explanation Processor

The explanation part allows the program to explain its reasoning to the user. These explanations include justifications for the system's conclusion (HOW queries), explanation of why the system needs a particular piece of data (WHY queries).

3. Knowledge Base

The heart of the expert system contains the problem solving knowledge (which defined as an original collection of processed information) of the particular applications, this knowledge is represented in several ways such as if-then rules form.

4. Inference Engine

The inference engine applies the knowledge to the solution of actual problems. It s the interpreter for the knowledge base. The inference engine performs the recognize act control cycle.

The inference engine consists of the following components:-

1. Rule interpreter.
2. Scheduler
3. HOW process
4. WHY process
5. knowledge base interface.

5. Working Memory

It is a part of memory used for matching rules and calculation. When the work is finished this memory will be raised.