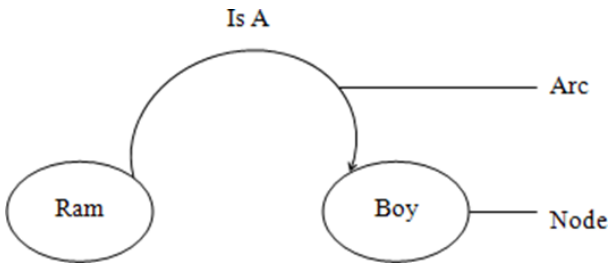
Semantic Network

A semantic network is a graphical knowledge representation technique. This knowledge representation system is primarily on network structure. The semantic networks were basically developed to model human memory. A semantic net consists of nodes connected by arcs. The arcs are defined in a variety of ways, depending upon the kind of knowledge being represented.

The main idea behind semantic net is that the meaning of a concept comes, from the ways in which it is connected to other concepts. The semantic network consists of different nodes and arcs. Each node should contain the information about objects and each arc should contain the relationship between objects. Semantic nets are used to find relationships among objects by spreading activation about from each of two nodes and seeing where the activation met this process is called intersection search.

For example: Ram is a boy.



Frame

A frame is a collection of attributes and associated values that describe some entity in the world. Frames are general record like structures which consist of a collection of slots and slot values. The slots may be of any size and type. Slots typically have names and values or subfields called facets. Facets may also have names and any number of values. A frame may have any number of slots, a slot may have any number of facets, each with any number of values. A slot contains information such as attribute value pairs, default values, condition for filling a slot, pointers to other related frames and procedures that are activated when needed for different purposes. Sometimes a frame describes an entity in some absolute sense, sometimes it represents the entity from a particular point of view. A single frame taken alone is rarely useful. We build frame systems out of collection of frames that are connected to each other by virtue of the fact that the value of an attribute of one frame may be another frame. Each frame should start with an open parenthesis and closed with a closed parenthesis.

Syntax of a frame

(<frame name>
(<slot 1> (<facet 1> <value 1>.....<value n_1>)
(<facet 2> <value 1>.....<value n_2>)
.
.
.
.
(<facet n> <value 1>..... <value n_n>))
(<slot 2> (<facet 1> <value 1>.....<value n_1>)
(<facet 2><value 2>.....<value n_2>)
.
.
))

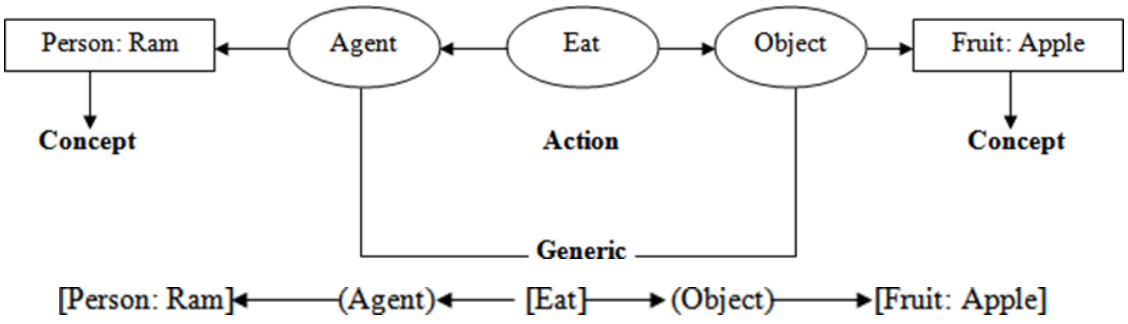
Conceptual Graphs

It is a knowledge representation technique which consists of basic concepts and the relationship between them. As the name indicates, it tries to capture the concepts about the events and represents them in the form of a graph. A concept may be individual or generic. An individual concept has a type field followed by a reference field. For example person: Ram. Here person indicates type and Ram indicates reference.

An individual concept should be represented within a rectangle in graphical representation and within a square bracket in linear representation. The generic concept should be represented within an oval in graphical representation and within a parenthesis in linear representation. Conceptual graph is a basic building block for associative network. Concepts like AGENT, OBJECT, INSTRUMENT, and PART are obtained from a collection of standard concepts. New concepts and relations can be defined from these basic ones. These are also basic building block for associative network. A linear conceptual graph is an elementary form of this structure. A single conceptual graph is roughly equivalent to a graphical diagram of a natural language sentence where the words are depicted as concepts and relationships.

Consider an example

"Ram is eating an apple "



Conceptual Dependency

It is another knowledge representation technique in which we can represent any kind of knowledge. It is based on the use of a limited number of primitive concepts and rules of formation to represent any natural language statement. Conceptual dependency theory is based on the use of knowledge representation methodology was primarily developed to understand and represent natural language structures. The conceptual dependency structures were originally developed by Roger C SChank in 1977.

If a computer program is to be developed that can understand wide phenomenon represented by natural languages, the knowledge representation should be powerful enough to represent these concepts. The conceptual dependency representation captures maximum concepts to provide canonical form of meaning of sentences. Generally there are four primitives from which the conceptual dependency structure can be described. They are

a. ACTS : Actions

b. PPs : Objects (Picture Producers)

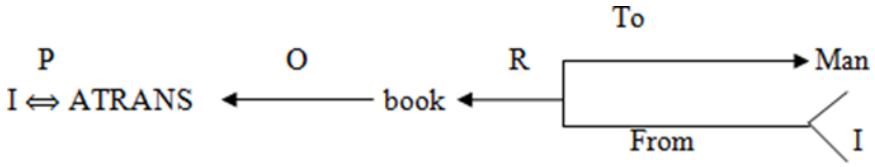
c. AAs : Modifiers of Actions (Action Aiders)

d. PAs : Modifiers of PPs (Picture Aiders)

e. TS : Time of action

Conceptual dependency provides both a structure and a specific set of primitives at a particular level of granularity, out of which representation of particular pieces of information can be constructed.

For example



Script

It is another knowledge representation technique. Scripts are frame like structures used to represent commonly occurring experiences such as going to restaurant, visiting a doctor. A script is a structure that describes a stereotyped sequence of events in a particular context. A script consist of a set of slots. Associated with each slot may be some information about what kinds of values it may contain as well as a default value to be used if no other information is available. Scripts are useful because in the real world, there are no patterns to the occurrence of events. These patterns arise because of clausal relationships between events. The events described in a script form a giant casual chain. The beginning of the chain is the set of entry conditions which enable the first events of the script to occur. The end of the chain is the set of results which may enable later events to occur. The headers of a script can all serve as indicators that the script should be activated.

Once a script has been activated, there are a variety of ways in which it can be useful in interpreting a particular situation. A script has the ability to predict events that has not explicitly been observed. An important use of scripts is to provide a way of building a single coherent interpretation from a collection of observation. Scripts are less general structures than are frames and so are not suitable for representing all kinds of knowledge. Scripts are very useful for representing the specific kinds of knowledge for which they were designed.

A script has various components like:

1) Entry condition 2) Tracks

3) Result 4) Probs

5) Roles 6) Scenes

7) Result

Monotonic Reasoning

Traditional systems based on predicate logic are monotonic. Here number of statements known to be true increases with time. New statements are added and new theorems are proved, but the previously known statements never become invalid.

In monotonic systems there is no need to check for in consistencies between new statements and the old knowledge. When a proof is made, the basis of the proof need not be remembered, since the old statements never disappear. But monotonic systems are not good in real problem domains where the information is incomplete, situations change and new assumptions are generated while solving new problems.

Logical Reasoning

Logical reasoning is the process of using a rational, systematic series of steps based on sound mathematical procedures and given statements to arrive at a conclusion.

In logical reasoning, an if-then statement (also known as a conditional statement) is a statement formed when one thing implies another.

Logics for Nonmonotonic reasoning

Monotonicity is fundamental to the definition of first-order predicate logic, we need some logics to support nonmonotonicity. The formal approach should possess all the features given,

- Defines the set of possible words according to the facts we have.
- Provides a way to say, it is preferable to believe in some model rather others.
- Provides the basis for implementation

Default Reasoning:

Default reasoning is one type of non-monotonic reasoning, which treats conclusions as, believed until a better reason is found to believe something else. Two different approaches to deal with non-monotonic system are:

- Nonmonotonic logic
- Default logic

Minimalist Reasoning

Minimalist reasoning follows the idea that "there are many fewer true statements than false ones. If something is true and relevant it makes sense to assume that is has been entered into our knowledge base. Therefore, assume that the only true statements are those that necessarily must be true in order to maintain the consistency of knowledge base".

Two kinds of minimalist reasoning are:

(1) Closed World Assumption (CWA).

(2) Circumscription.

Statistical Reasoning

Statistical Reasoning, in which each of the rules is inferences, should be assigned with some numerical measure. It consider either true or false but not both.

Bay's Theorem

The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence. In other words, it allows scientists to combine new data with their existing knowledge or expertise. The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes, and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (ie, the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise.

Bayes' Theorem

- **Bayes' theorem lets us calculate a conditional probability:**

$$P(B \mid A) = \frac{P(A \mid B) \cdot P(B)}{P(A)}$$

- **P(B) is the prior probability of B.**
- **P(B | A) is the posterior probability of B**

Certainty Factor

One of the unique aspects of MYCIN was its ability to handle

―uncertainty‖. Unlike a game playing situation or a planning system, diagnosis is fraught with uncertainty. In order to model that uncertainty, MYCIN introduced a concept called Certainty Factors (CF). Based on backward chaining, MYCIN's algorithm works as follows:

–Add ―diagnose-and-treat‖ to working memory along with all reported patient data.

–Repeat

- •Identify all rules that can provide the conclusion currently sought
- •Match right hand sides (that is, search for rules whose right hand sides match anything in working memory)
- •Use conflict resolution to identify a single rule
- •Fire (execute) that rule

–Find and remove a piece of knowledge which is no longer needed

–Find and modify a piece of knowledge now that more specific information is known

–Add a new sub goal (left-hand side conditions that need to be proved)

–Until the action ―done‖ is added to working memory

Based on the rules, MYCIN would go through several distinct phases, identify the illness (es), and order tests as needed to refine the identification, generate a treatment. MYCIN also had the capability of answering questions during the diagnosis. For instance, when the user is asked a question or to perform a test, the user can ask ―why?‖ in which case MYCIN would explain why. Similarly, when a diagnostic conclusion and/or treatment is reported, the user can ask why MYCIN decided that. The explanation is essentially the rules that are in the current chain of logic. For instance, given rule 52, if MYCIN is attempting to determine if pseudomonas is a proper identification of the illness and MYCIN asks the user if the patient has serious burns and the user asks ―why?‖, MYCIN can report by saying ―if this is true, and given that we already know that the patient‘s blood culture sample has been identified as having a morphology of rod and is gram negative, then this would allow us to conclude the possibility of pseudomonas.

Dempster-Shafer Theory

Dempster–Shafer theory is a generalization of the Bayesian theory of subjective probability; whereas the latter requires probabilities for each question of interest, belief functions base degrees of belief (or confidence, or trust) for one question on the probabilities for a related question. These degrees of belief may or may not have the mathematical properties of probabilities; how much they differ depends on how closely the two questions are related.[5] Put another way, it is a way of representing epistemic plausibilities but it can yield answers that contradict those arrived at using probability theory.

Often used as a method of sensor fusion, Dempster–Shafer theory is based on two ideas: obtaining degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule[6] for combining such degrees of belief when they are based on independent items of evidence. In essence, the degree of belief in a proposition depends primarily upon the number of answers (to the related questions) containing the proposition, and the subjective probability of each answer. Also contributing are the rules of combination that reflect general assumptions about the data.

Fuzzy Logic

In this context, FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make decisions, only much faster.