CHAPTER – 3

KNOWLEDGE AND REASONING

Representation and Mapping:

In order to solve the complex problem in artificial intelligence we need a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to new problems. While solving the problems in AI we need different types of knowledge and only one knowledge representation technique is not sufficient to represent different types of knowledge. While representing knowledge we need to deal with two things:

1. ***Facts:***

Facts that are relevant in the real world and may be in different form or format. These are the things we need to represent.

2. ***Representation of Facts:***

To represent different types of facts by using some formal representation mechanism. The representation must be such that we can manipulate the facts using some standard rules.

We can assume this structure of knowledge representation in two levels as:

1. ***Knowledge level:***

At which facts including behavior of the agent and goal states are described.

2. ***Symbol level:***

At which representation of objects at the knowledge level are defined in terms of symbols that can be manipulated by computer programs.

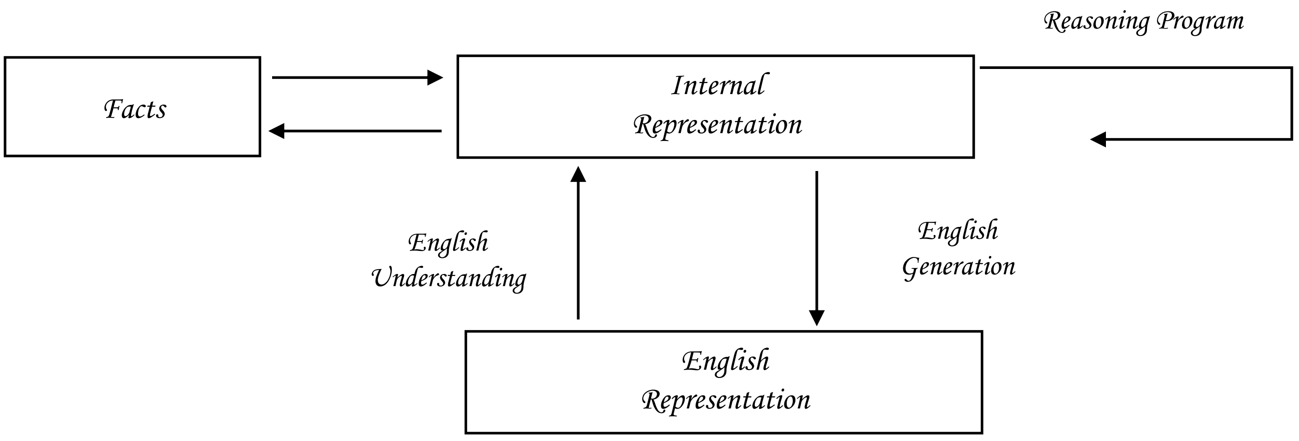


Fig: 2-Way Mapping Between Facts and Representation

The representation of knowledge in AI can be described by a 2-way mapping between facts and representations. The forward mapping maps facts to internal representation whereas backward mapping maps internal representation to facts (in natural language like the English language). The AI system can apply different reasoning program to the internal knowledge base to derive new knowledge.

Properties of Good Knowledge Representation:

A good system for the representation of knowledge in a particular domain should possess the following four properties:

1. ***Representational Adequacy:***

The ability to represent all kinds of knowledge that are needed in a particular domain.

2. ***Inferential Adequacy:***

The ability to manipulate the representational structure in such a way to derive new structure corresponding to new knowledge using the existing knowledgebase.

3. ***Inferential Efficiency:***

The ability to direct the inferential mechanism into the most productive directions by using the appropriate guideline.

4. ***Acquisitional Efficiency:***

The ability to acquire new knowledge using automatic methods wherever possible rather than dependency on human intervention.

Approaches to Knowledge Representation:

Different knowledge representation approaches are:

1. ***Simple Relational Knowledge:***

The simplest way of storing facts is to use a relational method where each fact about a set of objects is set out systematically in columns. This representation gives little opportunity for inference, but it can be used as the knowledge basis for inference engines.

2. ***Inheritable Knowledge:***

Relational knowledge is made up of objects consisting of:

- Attributes
- Corresponding associated values

We extend the base more by allowing inference mechanisms:

- Elements inherit values from being members of a class.
- Data must be organized into a hierarchy of classes.

3. ***Inferential Knowledge:***

Inferential knowledge represent knowledge as formal logic like all dogs have tails.

V(X): dog(x)→ has a tail(x)

It is popular in AI system because of following advantages:

- A set of strict rules.
- Can be used to derive more facts.

- Truths of new statements can be verified.
- Guaranteed correctness.

4. ***Procedural Knowledge:***

In this representation knowledge are encoded in some procedure. Procedure are small program that know how to do specific things, how to proceed.

Issues in Knowledge Representation:

While representing knowledge in AI using various knowledge representation method, we should consider several issues and some of them are:

1. ***Important Attributes:***

While representing knowledge about object in terms of attributes (characteristics), we need to make sure that the important attributes are handled appropriately if exist.

2. ***Relationship Among Attributes:***

The attributes we use to describe objects may be entities themselves in such case there may exist various relationships among those attributes and few relationships are:

- Inverse
- Existence of an "ISA" hierarchy
- Techniques for reasoning about values
- Single valued attributes.

3. ***Choosing the Granularity of Representation:***

At what level of detail knowledge should be represented? Either at the detailed or summary level. This issues is important while describing objects about real world.

4. ***Representing Set of Objects:***

Some properties are shared by objects. In this case to simplify the representation of object, we can create a set of objects sharing the common properties. This approach also simplifies the inference process.

5. ***Finding the Right Structures as Needed:***

Given a large amount of knowledge from different fields, it is an important issue to choose the appropriate knowledge structure before representing the knowledge and storing them into the memory.

Propositional Logic (PL):

Propositional Logic is one of the simplest way to express knowledge of proposition. A proposition is a declarative sentence that can be definitely either true or false.

A proposition can be a simple (having only one proposition) and compound (a combination of multiple propositions). To construct compound propositions we use logical connectives:

Conjunction (And '∧'), Disjunction (Or '∨'), Negation (Not '~'), Conditional (IF THEN '→'), and Bi-conditional (IF AND ONLY IF '↔').

Syntax:

The syntax of propositional logic is simple. The symbols of propositional logic are the logical constant True and False. Propositional symbols such as P & Q, the logical connectives Conjunction (And '∧'), Disjunction (Or '∨'), Negation (Not '~'), Conditional (IF THEN '→'), Bi-conditional (IF AND ONLY IF '↔') and parenthesis (). All propositions or sentences are made by putting these symbols together and using the following rules:

- The logical constant True and False are sentences by themselves.
- A propositional symbol such as P or Q is a sentence by itself.
- A sentence within a parenthesis is also a sentence like (P) and (Q).
- A sentence can be formed by combining simple sentences using the logical connectives are:
    - ✓ If P and Q are sentences then (P∧Q) is also a sentence.
    - ✓ If P and Q are sentences then (P∨Q) is also a sentence.
    - ✓ If P is a sentence then ~P is also a sentence.
    - ✓ If P and Q are sentences then P→Q is also a sentence.
    - ✓ If P and Q are sentences then P↔Q is also a sentence.

In other way:

Sentence → AutomicSentence|ComplexSentence

AutomicSentence → T|F

P|Q|R.....

ComplexSentence → (Sentence)|Sentence LogicalConnective sentence|~Sentence

LogicalConnective → ∧|∨|~|→|↔

Fig: A BNF Grammar of PL

Semantics:

The semantics of propositional logic is also quite straight forward we define it by specifying the interpretation of proposition symbols and constants and specifying the meanings of the logical connectives.

The interpretation of any propositional symbol is either True or False. The truth values of complex proposition constructed by logical connectives will be as:

P	**Q**	**~P**	**P∧Q**	**P v Q**	**P→Q**	**P↔Q**
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Rules of Inference for Propositional Logic:

Mathematical logic is often used for logical proofs. Proofs are valid arguments that determine the truth values of mathematical statements.

An argument is a sequence of statements. The last statement is the conclusion and all its preceding statements are called premises (or hypothesis). The symbol "∴", (read therefore) is placed before the conclusion. A valid argument is one where the conclusion follows from the truth values of the premises.

Rules of Inference provide the templates or guidelines for constructing valid arguments from the statements that we already have.

Rules of Inference.

Rule of Inference	*Tautology*	*Name*
p $p \to q$ $\therefore q$	$[p \wedge (p \to q)] \to q$	Modus ponens
$\neg q$ $p \to q$ $\therefore \neg p$	$[\neg q \wedge (p \to q)] \to \neg p$	Modus tollens
$p \to q$ $q \to r$ $\therefore p \to r$	$[(p \to q) \wedge (q \to r)] \to (p \to r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$[(p \vee q) \wedge \neg p] \to q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \to (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \to p$	Simplification
p q $\therefore p \wedge q$	$[(p) \wedge (q)] \to (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \to (q \vee r)$	Resolution

FIRST ORDER PREDICATE LOGIC (FOPL):

To express the real world knowledge propositional logic may not be sufficient. To express the knowledge about real world including quantity and variables we use FOPL, also known as First Order Logic.

Syntax:

Basic elements in FOPL are:

1. **Constant Symbols**:

Generally denotes as P, Q, Ram, etc. express the constant values of real world.

2. **Predicate Symbols:**

Use to express relations or functional mapping from the elements of a particular domain to the values True or False. For example: MARRIED, EQUAL, etc.

3. **Function Symbols:**

Function symbols denotes relations of one object with another for example FATHER_OF, BROTHER_OF, etc.

4. **Variables:**

Variables are the term that can assume different values over the given domain. For example: x, y, z…

5. **Quantifiers:**

Two quantifier's symbols, Universal Quantifiers (∀) and Existential Quantifiers (∃) are used for the indication of quantity. Universal quantifiers are used to denote "For All", like $\forall_x$ and Existential quantifier is used to denote "For some" like $\exists_x$.

Logical Connectives:

They are used to connect sentences and they are Conjunction (And '⋀'), Disjunction (Or '⋁'), Negation (Not '~'), Conditional (IF THEN '→'), Bi-conditional (IF AND ONLY IF '↔').

Sentence → Atomic Sentence | Sentence Connective Sentence | Quantifier Variable Sentence | ~ Sentence | (Sentence)

Atomic Sentence → Predicate (Term …)

Term → Function (Term…) | Constant | Variable

Connective → ⋀ | ⋁ | → | ↔

Quantifier → ∀ | ∃

Constant → P | Q | Ram | …

Variable → x | y | z | …

Predicate → BEFORE | AFTER | MARRIED | …

Function → FATHER_OF | Left_Leg_Of | …

Few Examples:

a. Marcus was a man.
b. Marcus was a Pompeian.
c. All Pompeian were Roman.
d. Caesar was a ruler.
e. All Roman were either loyal to Caesar or hated him.
f. Everyone is loyal to someone.
g. People only try to assassinate rulers they are not loyal to.

h. Marcus tied to assassinate Caesar.

Let's assume that:

MAN (x): x is a man.

POMP (x): x is a Pompeian.

ROMAN (x): x is a Roman.

RULER (x): x is a ruler.

LOYALTO (x, y): x is loyal to y.

HATE (x, y): x hates y.

PEOPLE (x): x is a people

TRYTOASSASS (x, y): x tries to assassinate y

Now The Symbolic Representation Of Above Statements Will Be As:

a. MAN (Marcus)
b. POMP (Marcus)
c. $\forall_x$ ROMAN (x) $\rightarrow$ ROMAN (x)
d. RULE (Caesar)
e. $\forall_x$ ROMAN (x) $\rightarrow$ (LOYALTO (x, Ceaser) $\vee$ HATE (x, Caesar))
f. $\forall_x \exists_y$ LOYALTO (x, y)
g. $\forall_x \exists_y$ PEOPLE (x) $\wedge$ TRYTOASSASS (x, y) $\wedge$ RULER (y) $\rightarrow$ ~ LOYALTO (x, y)
h. TRYTOASSASS (Marcus, Caesar)

Conversion to Clausal Form:

To transform a sentence into clausal form follow the following steps:

Step 1:

Eliminate all implication ($\rightarrow$) and equivalence ($\leftrightarrow$) symbol. For this use:

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

$(P \leftrightarrow Q) \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

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

Step 2:

Move all negative symbol into the individual atoms. For this use the following rules:

DeMorgan's Rules:

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

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

$\sim \forall_x P(x) \equiv \exists_x \sim P(x)$

$\sim \exists_x P(x) \equiv \sim \forall_x P(x)$

Double Negation Rule:

$\sim(\sim P) \equiv P$

Step 3:

Standardized variables if necessary, so that each quantifiers have different variables. For example:

Replace: $\forall_x P(x) \land \exists_x Q(x)$

With: $\forall_x P(x) \land \exists_y Q(y)$

Step 4:

Move all quantifiers to the left of the expression without changing their relative order.

Replace: $\forall_x P(x) \land \exists_y Q(y)$

With: $\forall_x \exists_y P(x) \land Q(y)$

Step 5:

The process of eliminating existential quantifier is known as Solemnization. In this step there are two different processes as:

a. If the left most quantifier in expression is an existential quantifier then replace all occurrences of the existentially quantified variable with any constant not appearing elsewhere in the expression and remove the quantifiers. For example:

   Replace: ~~$\exists_x$~~ $\forall_y \exists_z (P(x) \land Q(y) \land R(z))$

   With: $\forall_y \exists_z (P(a) \land Q(y) \land R(z))$

b. For each existential quantifier that is preceded by one or more universal quantifiers, replace all occurrences of existentially quantified variable by a function symbol not appearing elsewhere in the expression. The arguments to the function must be all the universally quantified variables, preceding the existential quantifier and remove the existential quantifier. For example:
   Replace: $\forall_y \exists_z (P(a) \land Q(y) \land R(z))$
   With: $\forall_y (P(a) \land Q(y) \land R(f(y)))$

Step 6:

Remove all universal quantifiers and put the expression into CNF (Conjunctive Normal Form). For example: $P \land (P \lor Q) \land (P \lor R \lor S)$

Step 7:

Drop all conjunction symbol, writing each clause in a separate line. For example: $P\ (P \lor Q)\ (P \lor R \lor S)$

Example:

Convert the following expressions into clausal form:

$\exists_x \forall_y (\forall_z P(f(x), y, z)) \rightarrow (\exists_u Q(x, u) \land \exists_v R(y, v))$

Step 1:

On eliminating the implication ($\rightarrow$).

$\exists_x \forall_y (\sim (\forall_z P (f(x), y, z))) \lor ((\exists_u Q (x, u) \land \exists_v R (y, v)))$

Step 2:

Moving negation to individual predicate.

$\exists_x \forall_y (\exists_z \sim P (f(x), y, z)) \lor ((\exists_u Q (x, u) \land \exists_v R (y, v)))$

Step 3:

No need to perform this step.

Step 4:

Moving all quantifiers to the left of the expression.

$\exists_x \forall_y \exists_z \exists_u \exists_v (\sim P (f(x), y, z)) \lor ((Q (x, u) \land R (y, v)))$

Step 5:

Eliminating the left most existential quantifier ($\exists_x$) replacing the variable "x" by a constant "a".

$\forall_y \exists_z \exists_u \exists_v (\sim P (f (a), y, z)) \lor ((Q (a, u) \land R (y, v)))$

Now, on eliminating existential quantifier's $\exists_z \exists_u \exists_v$ by replacing the occurrences of variables z, u, and v with the functions g(y), h(y) and i(y) respectively.

$\forall_y (\sim P (f (a), y, g(y))) \lor ((Q (a, h(y)) \land R (y, i(y))))$

Step 6:

Eliminating universal quantifier and converting the expression into CNF. We have,

$\forall_y (\sim P (f (a), y, g(y))) \lor ((Q (a, h(y)) \land R (y, i(y))))$

$\equiv (\sim P (f (a), y, g(y))) \lor ((Q (a, h(y)) \land R (y, i(y))))$

$\equiv (\sim P (f (a), y, g(y)) \lor Q (a, h(y))) \land (\sim P (f (a), y, g(y)) \lor R (y, i(y)))$

Step 7:

On dropping conjunction symbol, different clauses are:

$(\sim P (f (a), y, g(y)) \lor Q (a, h(y)))$

$(\sim P (f (a), y, g(y)) \lor R (y, i(y)))$

Resolution in Predicate Logic:

Algorithm:

Step 1:

Convert all the statements in clause form.

Step 2:

Negate the conclusion and convert it into clause form.

Step 3:

Repeat until a contradiction is found or no progress can be made,

a. Select any two clauses and call them as parent clauses.
b. Apply resolution on them.
c. If the resolvent is the empty clause then the contradiction has been found. If it is not then add it to the set of available clauses for further procedure.

Example:

a. Marcus was a man.
b. Marcus was a Pompeian.
c. All Pompeians were Roman.
d. Caesar was a ruler.
e. All Roman were either loyal to Caesar or hated him.
f. Everyone is loyal to someone.
g. Man only try to assassinate rulers, they are not loyal to.
h. Marcus tied to assassinate Caesar.

Using above statements and resolution proof show that Marcus hated Caesar.

Let's assume that:

MAN (x): x is a man.

POMP (x): x is a Pompeian.

ROMAN (x): x is a Roman.

RULER (x): x is a ruler.

LOYALTO (x, y): x is loyal to y.

HATE (x, y): x hates y.

PEOPLE (x): x is a people

TRYTOASSASS (x, y): x tries to assassinate y

Now The Symbolic Representation Of Above Statements Will Be As:

a. MAN (Marcus)
b. POMP (Marcus)
c. $\forall_x$ ROMAN (x) $\rightarrow$ ROMAN (x)
d. RULE (Caesar)
e. $\forall_x$ ROMAN (x) $\rightarrow$ (LOYALTO (x, Caesar) $\vee$ HATE (x, Caesar))
f. $\forall_x \exists_y$ LOYALTO (x, y)
g. $\forall_x \exists_y$ MAN (x) $\wedge$ RULER (y) $\wedge$ RULER (y) $\wedge$ TRYTOASSASS (x, y) $\rightarrow \sim$ LOYALTO (x, y)
h. TRYTOASSASS (Marcus, Caesar)

Conclusion: HATE (Marcus, Caesar)

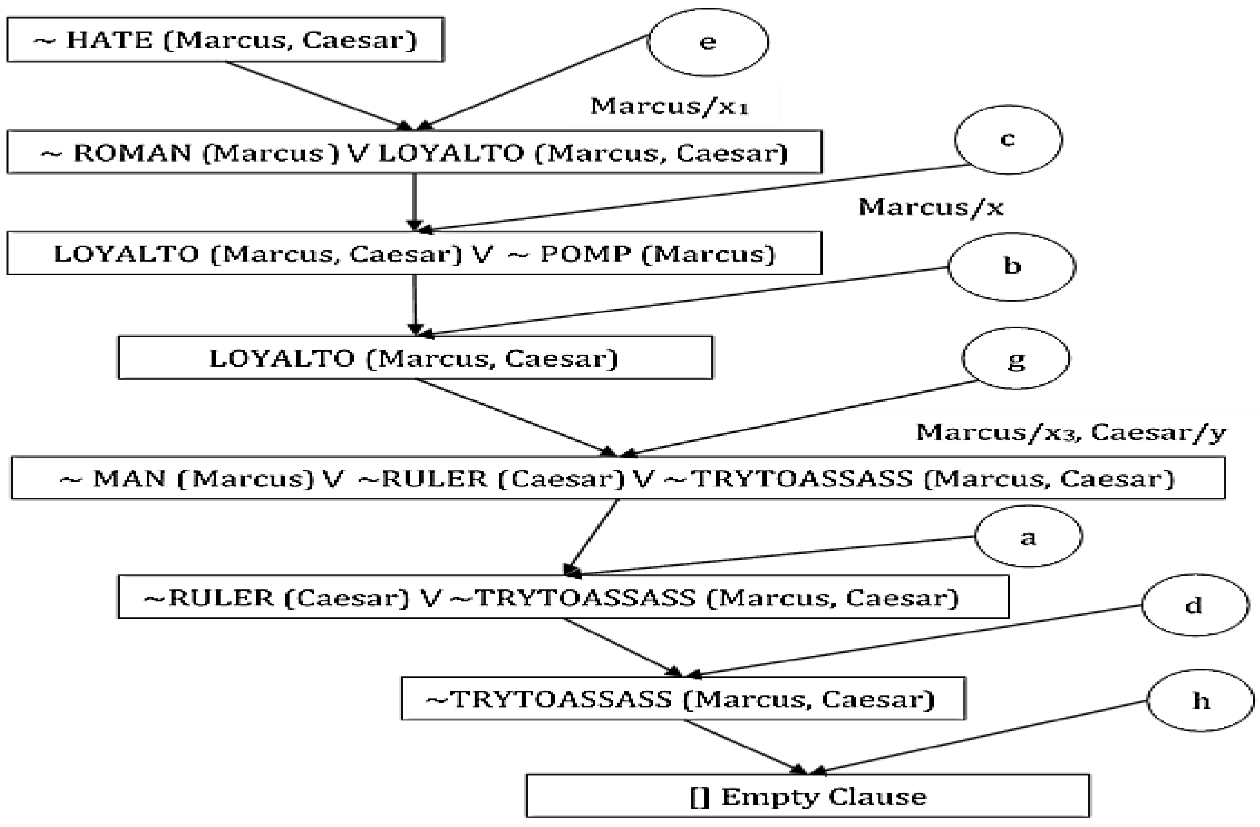
Now, converting all the given statements into clausal form we have:

a. MAN (Marcus)
b. POMP (Marcus)
c. $\forall_x \sim$ POMP (x) ⋁ ROMAN (x)
   ≡ ~ POMP (x) ⋁ ROMAN (x)
d. RULER (Caesar)
e. $\forall_x \sim$ ROMAN (x) ⋁ (LOYALTO (x, Caesar) ⋁ HATE (x, Caesar))
   ≡ $\forall_x \sim$ ROMAN (x) ⋁ LOYALTO (x, Caesar) ⋁ HATE (x, Caesar)
   ≡~ ROMAN (x_1) ⋁ LOYALTO (x_1, Caesar) ⋁ HATE (x_1, Caesar)
f. $\forall_x$ LOYALTO (x, f(x))
   ≡ LOYALTO (x_2, f(x_2))
g. $\forall_x \forall_y \sim$ (MAN (x) ⋀ RULER (y) ⋀ TRYTOASSASS (x, y)) ⋁ LOYALTO (x, y)
   ≡ $\forall_x \forall_y \sim$ MAN (x) ⋁ ~RULER (y) ⋁ ~TRYTOASSASS (x, y) ⋁ LOYALTO (x, y)
   ≡ ~ MAN (x_3) ⋁ ~RULER (y) ⋁ ~TRYTOASSASS (x_3, y) ⋁ LOYALTO (x_3, y)
h. TRYTOASSASS (Marcus, Caesar)

Now, Negation of Conclusion.

i. ~ HATE (Marcus, Caesar), which is already in clause form.

Now applying the proof by resolution:


~ HATE (Marcus, Caesar)
e
Marcus/x1
~ ROMAN (Marcus) ⋁ LOYALTO (Marcus, Caesar)
c
Marcus/x
LOYALTO (Marcus, Caesar) ⋁ ~ POMP (Marcus)
b
LOYALTO (Marcus, Caesar)
g
Marcus/x3, Caesar/y
~ MAN (Marcus) ⋁ ~RULER (Caesar) ⋁ ~TRYTOASSASS (Marcus, Caesar)
a
~RULER (Caesar) ⋁ ~TRYTOASSASS (Marcus, Caesar)
d
~TRYTOASSASS (Marcus, Caesar)
h
[] Empty Clause

Knowledge Representation using Rules:

Declarative and Procedural Knowledge:

A declarative representation is one in which knowledge is specified as facts but the use to which that knowledge is, is not given. To use a declarative representation we must connect (augment) it with a program that specifies what is to be done to knowledge and how. For example: a database containing various data is the example of declarative knowledge and a program written in any language specifies how to use those data.

A procedural representation is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself. To use procedural representation, we need to augment it with an interpreter that follows the instruction given in the knowledge. For example: an algorithm that provide step by step procedure to solve a given problem with the available set of declarative facts (input data).

Logic Programming:

Logic programming is a programming logic paradigm in which logical ascertains are viewed as program. PROLOG and LISP are two commonly used logic programming languages. The syntax of PROLOG is very much similar to the representation of knowledge in FOPL.

In logic programming like PROLOG, first of all, we build a knowledgebase by representing various facts and knowledge in a standard form. After that we can write different types of queries and PROLOG provides answers of those queries based on inference rules.

While writing programs in PROLOG we represent facts and rules according to the syntax of PROLOG. After this the program is compile. Now, we can write different goals or queries and the answer of those goals are provided in Boolean form or in the form of certain facts.

Forward and Backward Reasoning/Chaining:

The object of a search procedure is to discover a path through a problem space from an initial state to goal state. There are actually two direction in which such search can proceed:

a. Forward, from the initial states
b. Backward, from the goal states

Reasoning forward from the initial states:

In this approach we consider the initial state as the root of the search tree and begin building of a tree by applying operators available to generate next state or nodes. This process continues until a state that matches the goal state is generated.

Reasoning backward from the goal states:

In this case the goal state is considered as the root of the tree and we apply back operators to generate next level node. This process continues until a state that matches the initial state is generated.

Four factors influences either forward or backward reasoning:

1. Are there more possible initial states or goal states? We should like to move from the smaller set of state to the larger set of states.
2. In which direction is the branching factor is greater? We would like to proceed in the direction with the lower branching factor.
3. Will the program be asked to justify to a user? If so it is important to proceed in a direction that correspond more closely with the way the user will think.

4. What kind of event is going to trigger a problem solving episode? If it is the arrival of a new fact, forward reasoning makes sense. If it is a query to which a response is desired, backward reasoning is more natural.

Symbolic Reasoning under Uncertainty:

In many problem domains it is not possible to create models with complete and consistent facts. We may need to solve problems with uncertain and incomplete models. A variety of logical frameworks and conceptual methods have been purposed for handling such situation (problem). Two commonly use approaches for symbolic reasoning under uncertainty are:

1. Non-Monotonic Reasoning:

In this approach the axioms and/or inference rules are expended to make it possible to reason with incomplete information. These systems have a properties that at any given moment, a statement is either believed to be true, believed to be false or not believed to be either.

While dealing with reasoning with uncertainty following issues must be addressed:

a. How can the knowledgebase be extended to allow inferences to be made on the basic of lack of knowledge as well as on the presence of knowledge?
b. How can the knowledgebase be updated properly when a new fact is added to the system or when an old fact is removed?
c. How can knowledge be used to help resolve conflicts when there are several inconsistent non-monotonic inferences that can be derived based on existing knowledgebase?

Logics for Non-Monotonic Reasoning:

Various formal approaches are used for reasoning in a monotonic environment. There are different logical method or logic used for monotonic reasoning and for all of them we need to find the following things:

a. Defines the set of possible worlds that could exist given the facts that we do have.
b. Provides a way to say that we prefer to believe in some model rather than others.
c. Provides the bases for a practical implementation of this kind of reasoning.
d. Corresponds to our intuitions about how this kind of reasoning works.

For non-monotonic reasoning two methods are commonly used and they are:

i. Abduction:

Standard logic performs deduction where as non-monotonic reasoning perform abduction. In abduction, we say that if we are given the facts: A → B and B, we can draw a conclusion "A".

ii. Inheritance:

One very common use of non-monotonic reasoning is a basis for inheriting attributes from already existing classes or objects. It is generally helpful when there exist a hierarchy relationship among objects or classes used for defining knowledge.

2. Statistical Reasoning:

We have several representation techniques that can be used to model belief systems in which, at any given point in time, a particular fact is believed to be true, believed to be false or not considered to be either. But for some kinds, it is useful to be able to describe beliefs that are not certain but for which there is some supporting evidence. For example, problems that contain genuine randomness. E.g. Card Games

Problems that could, in principle modeled using the techniques that we used in uncertainty. For such problems, statistical measures may serve a very useful function as summaries of the world; rather than looking for all possible exceptions we can use a numerical summary that tells us how often an exception of some sort can be expected to occur.

This is useful for dealing with problems where there are randomness and unpredictability (such as in games of chance) and also for dealing with problems where we could, if we had sufficient information, work out exactly what is true. To do all this in a principled way requires techniques for probabilistic reasoning.

Probability & Bayes Theorem:

An important goal for many problem-solving systems is to collect evidence as the system goes along and to modify its behavior on the basis of evidence. To model this behavior we need a statistical theory of evidence. Bayesian statistics is such a theory. The fundamental notion of Bayesian statistics is that of conditional probability.

Read the expression as the probability of Hypothesis H given that we have observed evidence E. To compute this, we need to take into account the prior probability of H (the probability that we would assign to H if we had no evidence) & the extent to which E provides evidence of H. To do this we need to define a universe that contains an exhaustive, mutually exclusive set of Hi's, among which we are trying to discriminate.

Bayer's' Theorem States Then That,

$$P(H_i/E) = \frac{P\,(E/H_i)\cdot P(H_i)}{\sum_{n=1}^{k} P(E/H_n)\cdot P(H_n)}$$

Where,

- P (Hi\E) = The probability that hypothesis Hi true given evidence E
- P (E\Hi) = The probability that we will observe evidence E given that hypothesis i is true.
- P (Hi) = The a priori probability that hypothesis *i* is true in the absence of any specific evidence. These probabilities called prior probabilities.
- K = The number of possible hypotheses.