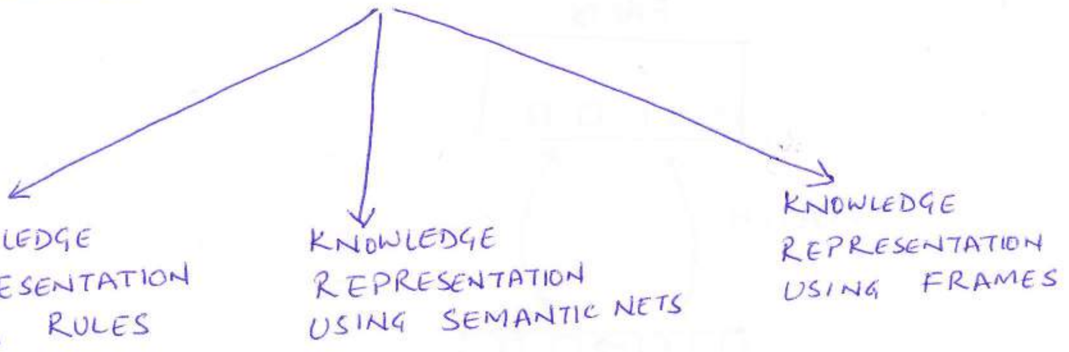
# UNIT - II

## TECHNIQUES OF KNOWLEDGE REPRESENTATION :



KNOWLEDGE REPRESENTATION USING RULES

KNOWLEDGE REPRESENTATION USING SEMANTIC NETS

KNOWLEDGE REPRESENTATION USING FRAMES

## KNOWLEDGE REPRESENTATION USING RULES :

Rules provide a formal way of representing recommendations, directives or strategies. Rules are expressed as IF-THEN statements as shown :

(1) If a flammable liquid was spilled,
call the fire department.

(2) If the pH of the spill is less than 6,
the spill material is an acid.

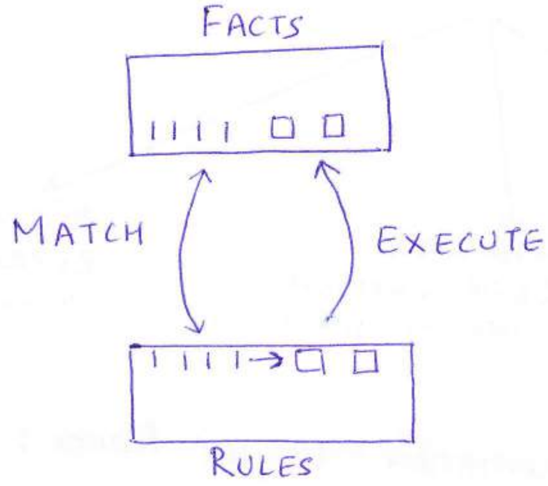
(3) If the spill material is an acid,
and the spill smells like vinegar,
the spill material is acetic acid.

Rules are written with arrows to indicate IF and THEN portions of the rules as shown :

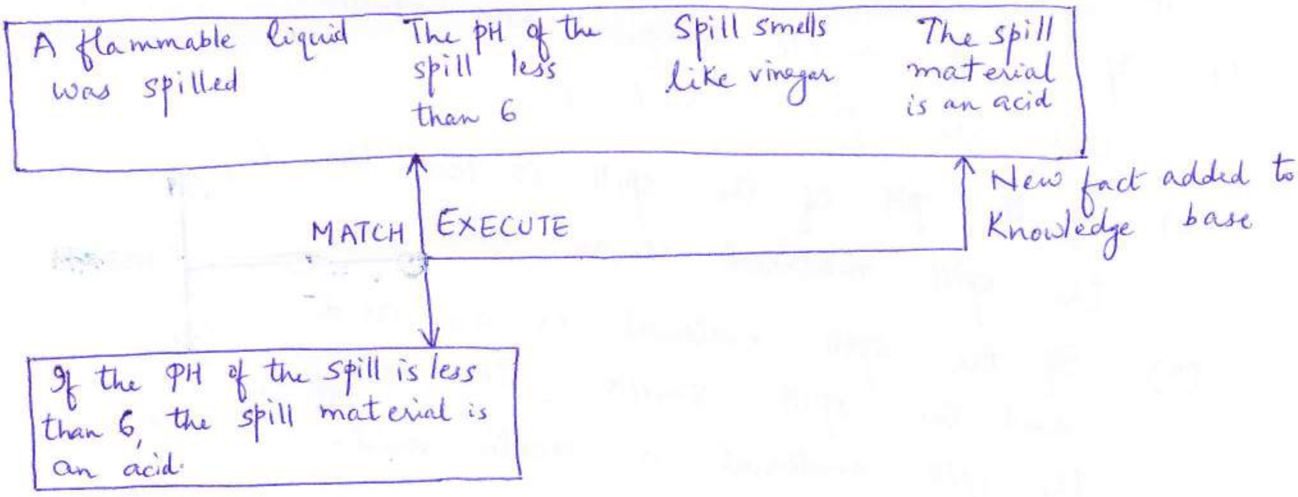
If the PH of the spill is less than 6 → the spill material is an acid

In a rule-based expert system, the domain knowledge is represented as sets of rules that are checked against a collection of facts or knowledge about the current situation.

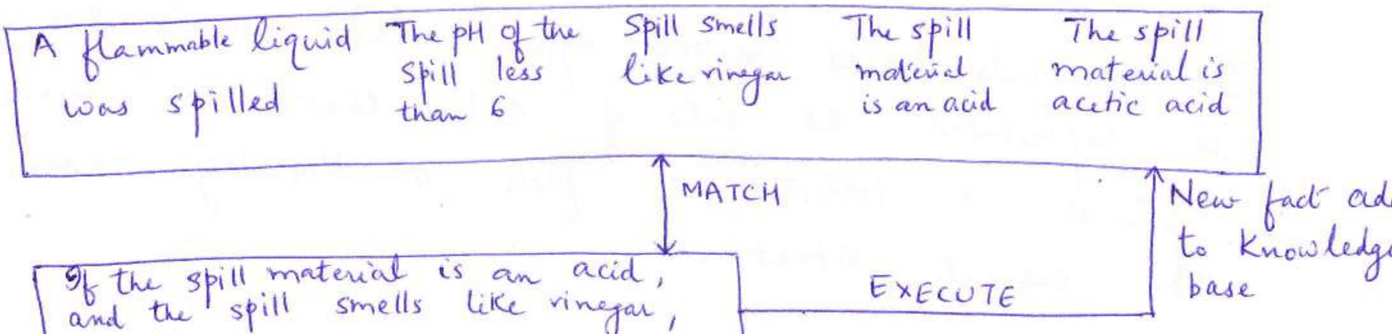
A rule interpreter compares the IF portions of rules with the facts and executes the rule whose IF portion matches the facts as shown:



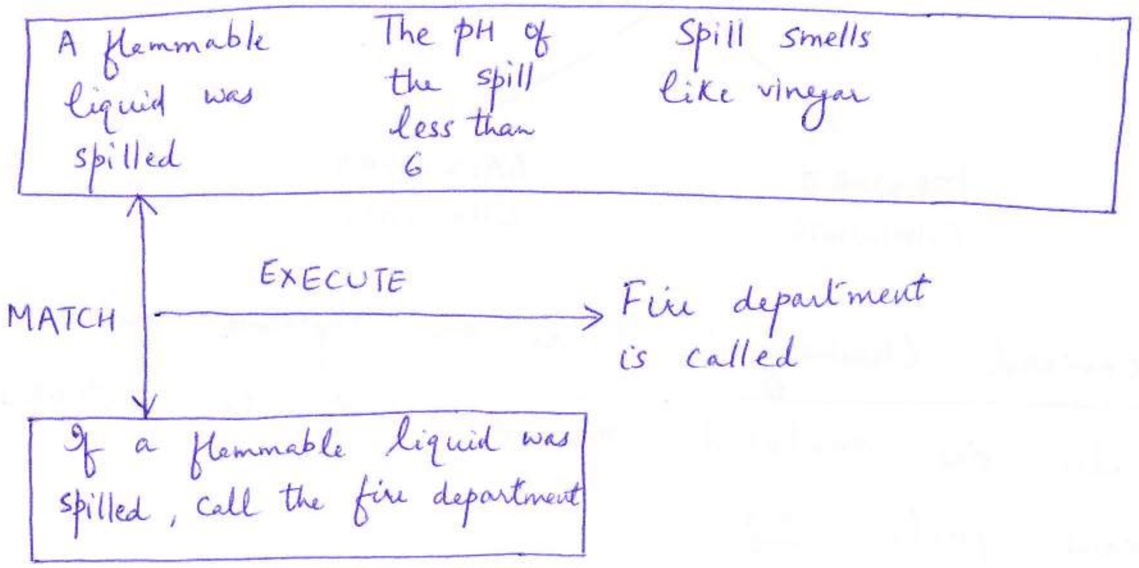
The rule's action may modify the set of facts in the Knowledge base by adding a new fact as shown:


A flammable liquid was spilled
The pH of the spill less than 6
Spill smells like vinegar
The spill material is an acid
New fact added to Knowledge base
MATCH
EXECUTE
If the PH of the spill is less than 6, the spill material is an acid.

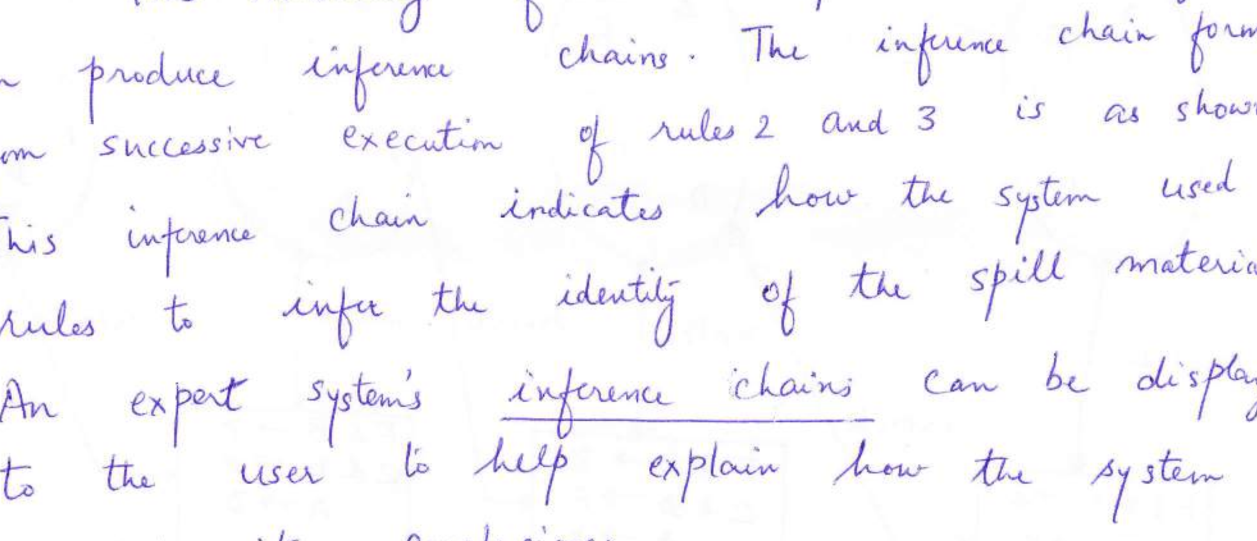
The new facts added to the Knowledge base can themselves be used to form matches with the IF portion of rules as shown:


A flammable liquid was spilled
The pH of the spill less than 6
Spill smells like vinegar
The spill material is an acid
The spill material is acetic acid
MATCH
New fact ad
to Knowledge base
If the spill material is an acid, and the spill smells like vinegar,
EXECUTE

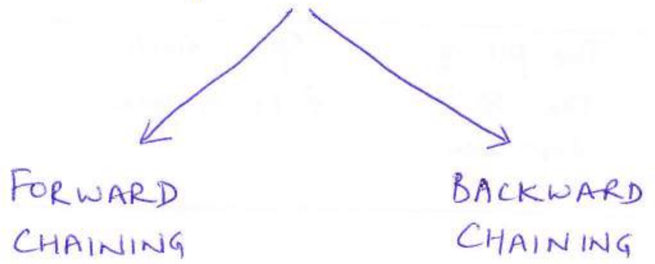
The action taken when the rule fires may directly affect the real world as shown:


A flammable liquid was spilled
The pH of the spill less than 6
Spill smells like vinegar
MATCH
EXECUTE
Fire department is called
If a flammable liquid was spilled, call the fire department

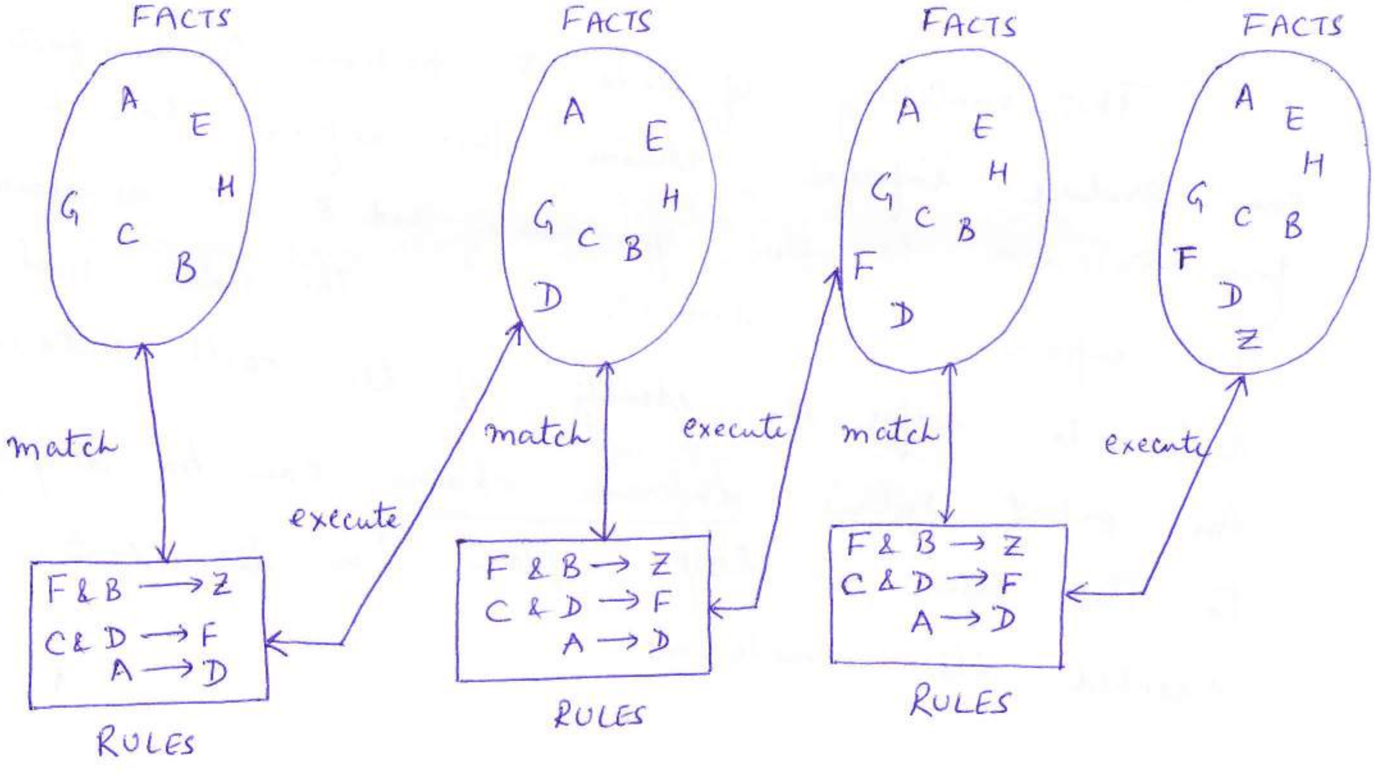
This matching of rule IF portions to the facts can produce inference chains. The inference chain formed from successive execution of rules 2 and 3 is as shown. This inference chain indicates how the system used the rules to infer the identity of the spill material. An expert system's inference chains can be displayed to the user to help explain how the system reached its conclusions.


The pH of the spill less than 6
The spill material is an acid
Spill smells like vinegar
The spill material is acetic acid

There are <u>two ways</u> in which rules can be used in rule - based system.



(a) <u>Forward Chaining</u> : It is an inference method where rules are matched against facts to establish new facts. <u>e.g.</u>



The rules use letters to stand for situations or concepts.

F & B → Z means

IF : Both situation F and situation B exist

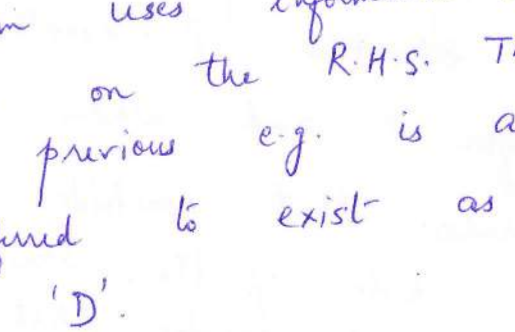
THEN : Situation Z also exists.

The Known set of facts is Known as <u>database</u>.

Each time the set of rules is tested against the data base, only the first rule that matches is executed. The rule A → D is only executed once even though it always matches the database.

- The first rule that fires is A → D because A is already in the data base. As a result, the existence of 'D' is inferred and 'D' is placed in the database.
- This causes the second rule C & D → F to fire and as a result 'F' is inferred and placed in the data base.
- This in turn causes the third rule F & B → Z to fire, placing 'Z' in the data base.

This technique is Known as forward chaining because the search for new information seems to be proceeding in the direction of the arrows separating the L.H.S. and R.H.S. of the rules. The system uses information on L.H.S. to derive information on the R.H.S. The inference chain produced by the previous e.g. is as shown. Situation 'Z' was inferred to exist as well as situations 'F' and 'D'.



Inference Chain

## Disadvantage of Forward Chaining:

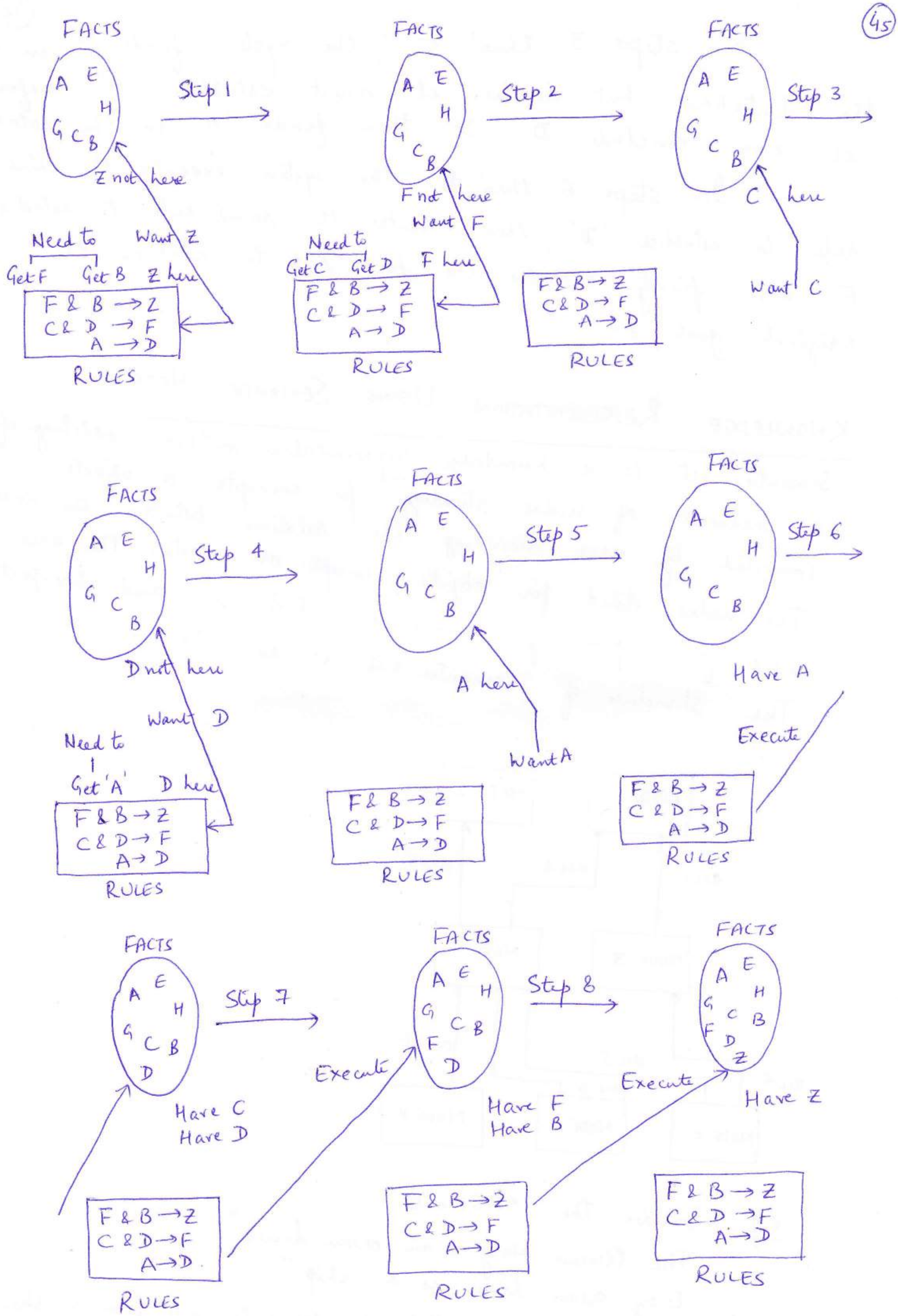
A real expert system would not have just three rules but have hundreds or thousands of rules. If a system just to find 'Z' is used, then many rules would be executed that had nothing to do with 'Z'. A large no. of inference chains and situations could be derived that were valid but unrelated to 'Z'. So if only one fact has to be inferred like 'Z', forward chaining could waste both time and money.

(b) Backward Chaining: It is more cost-effective than forward chaining. It is an inference method where the system starts with what it wants to prove. e.g. that situation 'Z' exists and only executes rules that are relevant to establishing it. e.g.

In step 1, the system is told to establish that situation 'Z' exists. It first checks the database for Z and when that fails searches for rules that conclude 'Z', i.e. have 'Z' on the right side of the arrow. It finds the rule F & B → Z and decides that it must establish 'F' and 'B' in order to conclude 'Z'.

In step 2, the system tries to establish 'F', first checking the database and then finding a rule that concludes 'F'. From this rule, C & D → F, the system decides it must establish 'C' and 'D' to

'F'

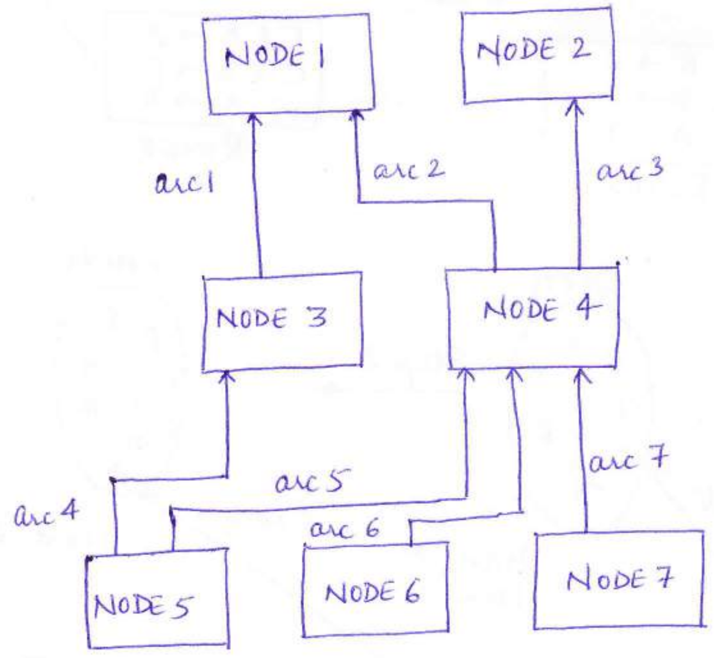

FACTS
A E H G C B
Step 1
Z not here
Want Z
Need to
Get F
Get B
Z here
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B
Step 2
F not here
Want F
Need to
Get C
Get D
F here
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B
Step 3
C here
Want C
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B
Step 4
D not here
Want D
Need to
Get 'A'
D here
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B
Step 5
A here
Want A
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B
Step 6
Have A
Execute
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B D
Step 7
Have C
Have D
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B F D
Step 8
Execute
Have F
Have B
F & B → Z
C & D → F
A → D
RULES
FACTS
A E H G C B F D Z
Execute
Have Z
F & B → Z
C & D → F
A → D
RULES

In steps 3 thro' 5, the system finds 'C' in the database but decides it must establish 'A' before it can conclude 'D'. It then finds 'A' in the databa

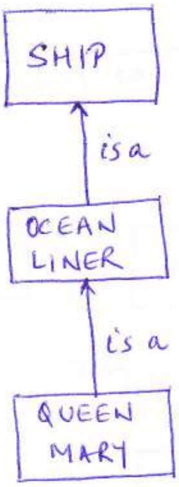
In steps 6 thro' 8, the system executes the third rule to establish 'D', then executes the second rule to establish 'F' and finally executes the first rule to establish the original goal, Z.

## KNOWLEDGE REPRESENTATION USING SEMANTIC NETS:

Semantic net is a Knowledge representation method consisting of a network of nodes standing for concepts or objects connected by arcs describing the relations between the nodes. The nodes stand for objects, concepts or events. The arcs used for representing hierarchies include 'isa' and 'has-part'. The structure of semantic net is as shown:

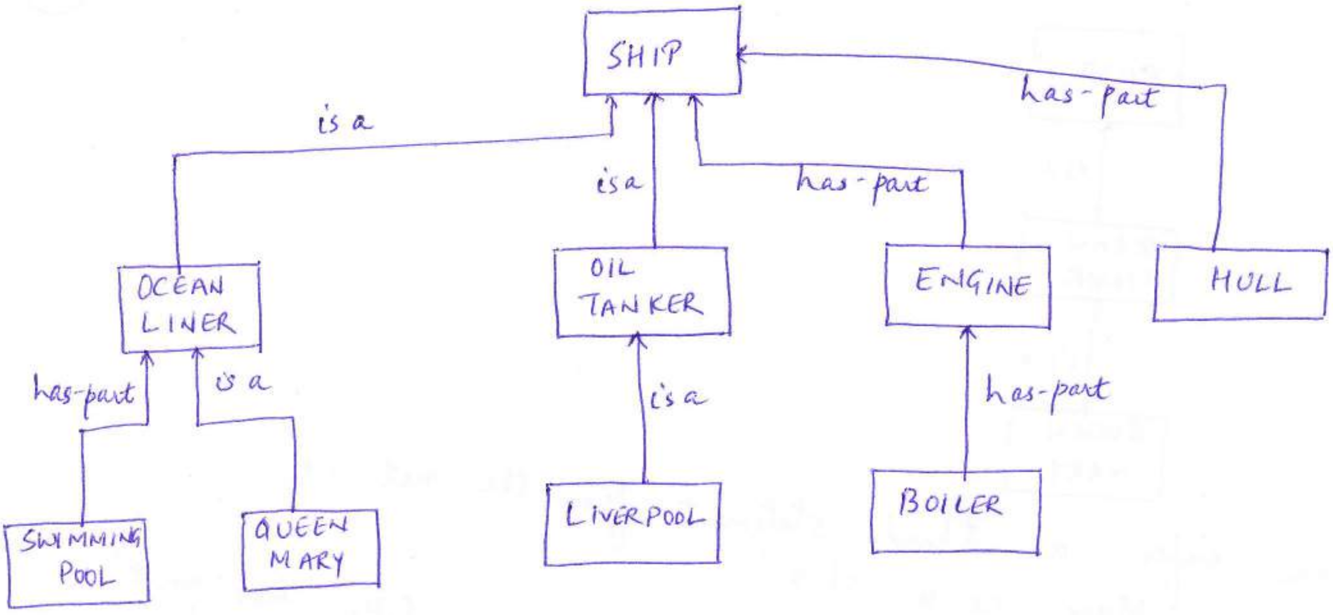


e.g. Consider the statements
" The Queen Mary is an ocean liner "
" Every ocean liner is a ship "
net form as shown:



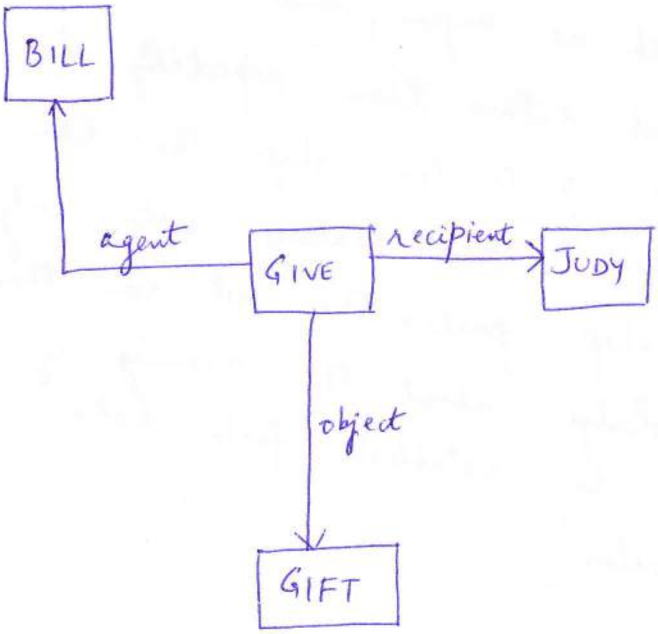
We can infer a third statement from the net i.e. "The Queen Mary is a ship"

The 'is a' relation and others like 'has-part' relation establish a property inheritance hierarchy in the net. It means that items lower in the net can inherit properties from items higher up in the net. This saves space since information about similar nodes doesn't have to be repeated at each node. It can be stored in one central location. e.g. In the ship semantic net the parts of a ship, such as engine, hull and boiler are stored once at ship level rather than repeatedly at lower levels like ship type or particular ship. This can save huge amounts of space even when dealing with only hundreds of ships and ship parts. The net can then be searched using Knowledge about the meaning of the relations in the arcs to establish facts like "The Queen Mary has a boiler"



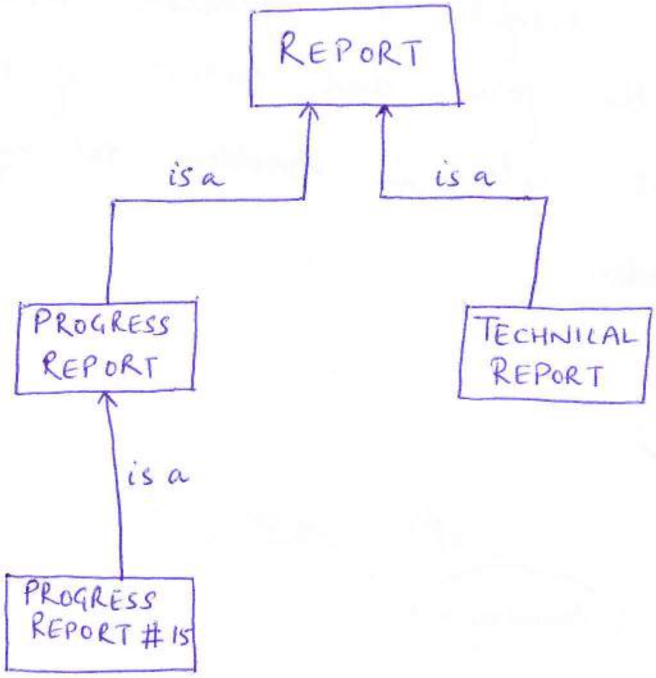
Semantic nets have been successfully used in 'natural language' research to represent complex sentences expressed in English and use arcs such as agent, object and recipient. e.g. The arcs define the relations between the predicate (GIVE) and the concepts (such as JUDY and GIFT) associated with that predicate.

Sentence : Bill gives Judy a gift.



# KNOWLEDGE REPRESENTATION USING FRAMES:

Frame refers to a special way of representing common concepts and situations. It is a network of nodes and relations organized in a hierarchy where the topmost nodes represent general concepts and the lower nodes more specific instances of those concepts. e.g. The concept of written report can be organized as shown:



The concept at each node is defined by a collection of attributes (e.g. name, color, size) and values of those attributes (e.g. Smith, red, small), where the attributes are called slots. Each slot can have procedures (arbitrary pieces of computer code) attached to it which are executed when the information in the slot (the values of the attribute) is changed.

Each slot can have any no. of procedures attached to it. The three types of procedures attached to slots are:

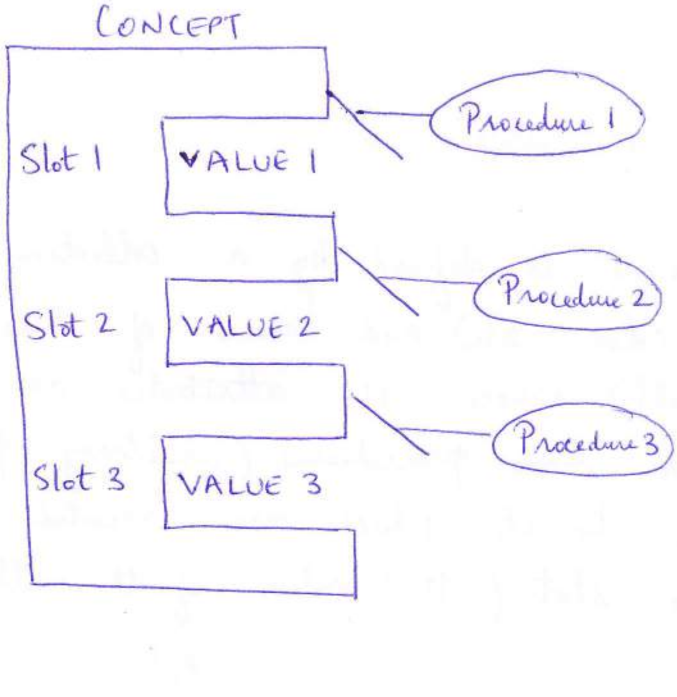
(a) If - added Procedure: Executes when new information is placed in the slot.

(b) If-removed Procedure: Executes when information is deleted from the slot.

(c) If-needed Procedure: Executes when information is needed from the slot, but the slot is empty.

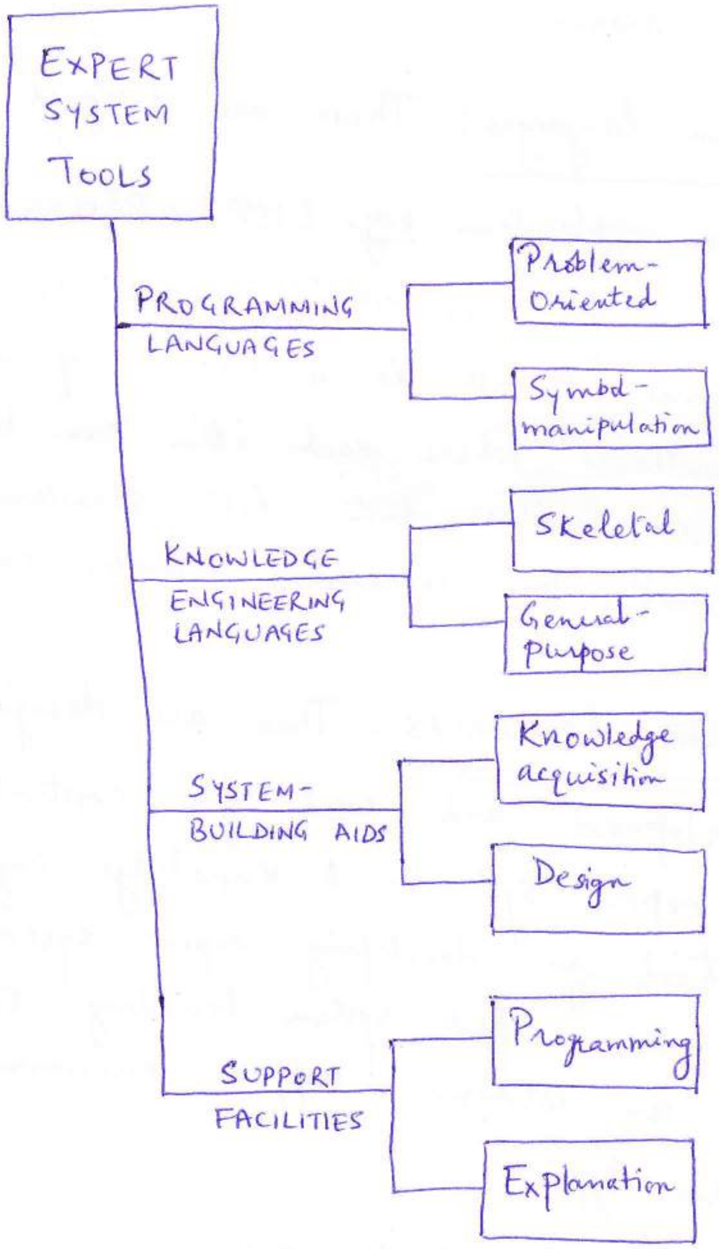
These attached procedures can monitor the assignment of information to the node, making sure that appropriate action is taken when values change. Frame systems are useful for problem domains where expectations about the form and content of the data play an important role in problem solving such as understanding speech.

A node is organized as shown:



## NATURE OF EXPERT SYSTEM TOOLS :

Expert system tools are programming systems that simplify the job of constructing an expert system. They range from very high-level programming languages to low-level support facilities. The expert system tools are divided into four major categories :



(1) PROGRAMMING LANGUAGES : These are used for expert system development. It is an artificial language developed to control and direct the operation of a computer.

These are of two types :

(a) Problem - oriented languages : These are designed for particular classes of problems. e.g. FORTRAN & PASCAL. FORTRAN has features for performing algebraic calculations and is applicable to scientific, mathematical and statistical problem areas.

(b) Symbol - manipulation languages : These are designed for artificial intelligence applications. e.g. LISP & PROLOG. LISP has mechanisms for manipulating symbols in the form of list structures. A list is a collection of items enclosed by parantheses where each item can be either a symbol or another list. List structures are useful building blocks for representing complex concepts.

(2) KNOWLEDGE ENGINEERING LANGUAGES : These are designed for expert system development and used for constructing and debugging expert systems. A Knowledge engineering language is a tool for developing expert systems and consists of an expert system building language integrated into an extensive support environment.

These are of two types :

(a) Skeletal Systems : A skeletal Knowledge engineering languag is a stripped - down expert system with its domain - specific Knowledge removed, leaving only the inference diagnosing and treating bacterial infections became the SKeletal System EMYCIN i.e. empty MYCIN. SKeletal Systems provide structure and built-in facilities that make system development easy and fast. They lack generality and flexibility. They apply to a restricted class of problems and reduce expert-system builder's design options.

(b) General-purpose Systems: A general-purpose Knowledge engineering language can handle many different problem areas and types. It provides more control over data access and search but may be more difficult to use. These languages vary a great deal in the extent of their generality and flexibility.

(3) SYSTEM-BUILDING AIDS: The system-building aids consist of programs that help acquire and represent the domain expert's Knowledge and programs that help design the expert system under construction. These programs address very difficult tasks. These are of two types:

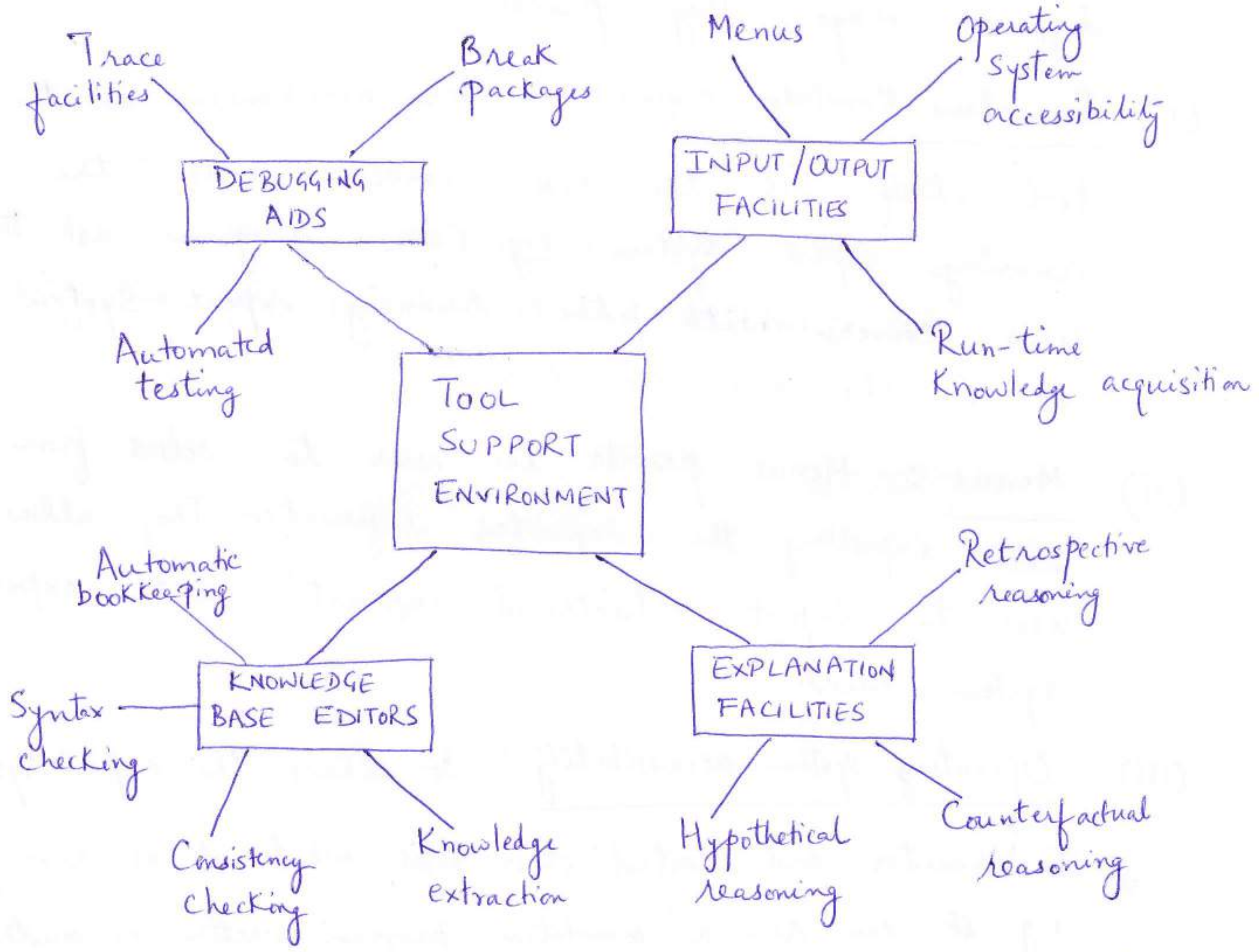
(a) Design-aids: This software tool helps the Knowledge engineer design and build expert system.
e.g. AGE System: AGE provides the user with a set of components which can be assembled to form portions of an expert System. Each component which is a collection of INTERLISP functions supports an expert system framework such as forward chaining,

The term blackboard refers to a central database used by systems to coordinate and control the operation of independent groups of rules called Knowledge sources. The Knowledge sources communicate by writing messages on the blackboard and reading messages from other Knowledge sources. Knowledge engineers have used AGE to design and build HANNIBAL, an expert system that performs situation assessment by interpreting enemy radio communication data.

(b) Knowledge acquisition aids: This system-building aid helps transfer Knowledge from a domain expert to a Knowledge base. e.g. TEIRESIAS system: This system acquires new rules about the problem domain through an interaction that allows users to state rules in a restricted subset of English. The system analyzes the rules, makes suggestions regarding their completeness and consistency and helps the user debug them. TEIRESIAS is a tool for exploring new ideas in Knowledge acquisition and database maintenance.

(4) SUPPORT FACILITIES : The support facilities consist of tools for helping with programming such as debugging aids and Knowledge base editors and tools that enhance the capabilities of the finished system such as built-in input / output and explanation mechanisms.

Components of a support environment for expert system tools



The support facilities consists of four types of tools:

(a) Debugging Aids : These consist of :-

(i) Trace facilities : Tracing provides the user with a trace or display of system operation by listing the names (or numbers) of all rules fired or showing the names of all subroutines called.

(ii) Break packages : It is a mechanism for telling the program where to stop so that the programmer can examine the values of variables at that point.

(iii) Automated Testing : It lets the user automatically test a program on a large number of benchmark problems to uncover errors or inconsistencies in the solutions.

(b) I/O Facilities : Different tools deal with I/O in different ways. They provide :

(i) Run-time Knowledge acquisition : Here mechanisms in the tool itself let the user converse with the running expert system. e.g. EMYCIN programs ask the user for needed information whenever they can't find it in the Knowledge base.

(ii) Menus : Menus provide the user to select from when inputting the requested information. They allow user to input volunteered information as the expert system runs.

(iii) Operating system accessibility : It allows the expert system to monitor and control other jobs while it is running. e.g. it can run a simulation program written in another language to obtain needed information. ROSIE has special commands that let a ROSIE expert system talk to the local operating system just as if it were a user on the system.

(c) Explanation Facilities : Almost all expert systems can explain to users how they reach particular conclusions. There are

(i) Retrospective reasoning: It explains how the system reached a particular state. e.g. The user may wish to know why the system needed the answer to the question it just asked or how the system arrived at a certain conclusion.

(ii) Hypothetical reasoning: The system explains what would have happened differently if a particular fact or rule had been different.

(iii) Counterfactual reasoning: The system explains why an expected conclusion was not reached.

(d) Knowledge Base Editors: Most expert system tools provide a mechanism for editing the knowledge base. There is a standard text editor for modifying rules and data by hand. The tools include four facilities in their support environment:

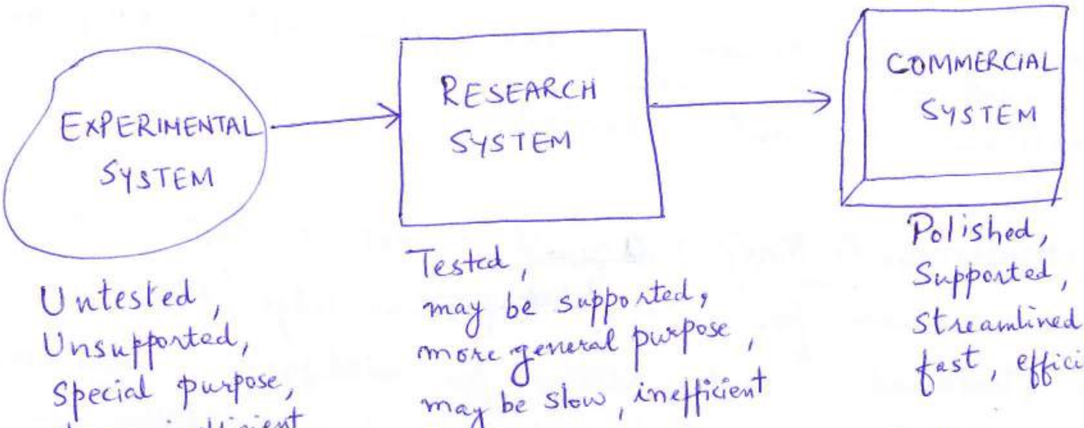
(i) Automatic bookkeeping: The editor monitors the changes made by the user and records pertinent information about the event. e.g. EMYCIN editor.

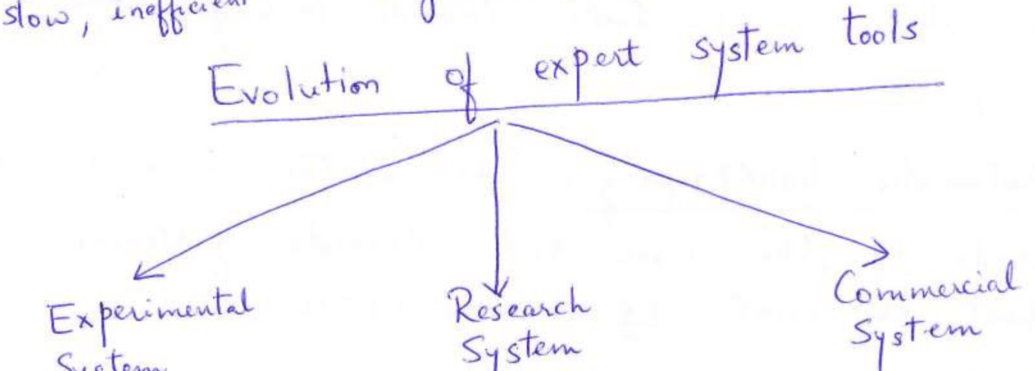
(ii) Syntax checking: The editor uses knowledge about the grammatical structure of the expert system language to help the user input rules with the correct spelling and format.

(iii) Consistency checking: The system checks the semantics or meanings of the rules and data being entered to see if they conflict with existing knowledge in the system. When a conflict occurs, the editor helps the user resolve the conflict by explaining what caused it and describing ways

(iv) Knowledge extraction: The editor helps the user enter new Knowledge into the system. It combines syntax and consistency checking. It would shorten system development time and training time for new system users.

## STAGES IN THE DEVELOPMENT OF EXPERT SYSTEM TOOLS


EXPERIMENTAL SYSTEM
RESEARCH SYSTEM
COMMERCIAL SYSTEM
Untested, Unsupported, Special purpose, slow, inefficient
Tested, may be supported, more general purpose, may be slow, inefficient
Polished, Supported, streamlined, fast, efficient



(1) Experimental System: Expert system tools start in research environment as an experimental system created for a specific task. The developer applies it to that task but seldom tests it on other problems. Such tools are slow, inefficient and use more computer time and memory.

(2) Research System: Expert system tool may reach the next stage of development and emerge as a research system. Such tool have been extensively tested. Such tools may be relatively slow and inefficient.

(3) Commercial System: Few expert system tools have reached the stage of a commercial system. These tools are polished, streamlined, well- t meet the user's needs and

## Representation and Programming Methods Supported by Expert System Tools:

An expert system tool may support one or more methods for representing & organizing knowledge:

| | Method | Description | Tool |
|---|---|---|---|
| (1) | Rule-based | Uses IF-THEN rules to perform forward or backward chaining. | EMYCIN |
| (2) | Frame-based | Uses frame hierarchies for inheritance and procedural attachment e.g. semantic nets & frames. | SRL |
| (3) | Procedure-oriented | Uses nested subroutines to organize and control program execution. | LISP |
| (4) | Object-oriented | Uses items called objects that communicate with one another via messages | SMALL-TALK |
| (5) | Logic-based | Uses predicate calculus to structure the program and guide execution. | PROLOG |
| (6) | Access-Oriented | Uses probes that trigger new computations when data are changed or read. | LOOPS |

(1) & (2) Rules, Semantic Nets and Frames: The three common methods for knowledge representation are rules, semantic nets and frames.

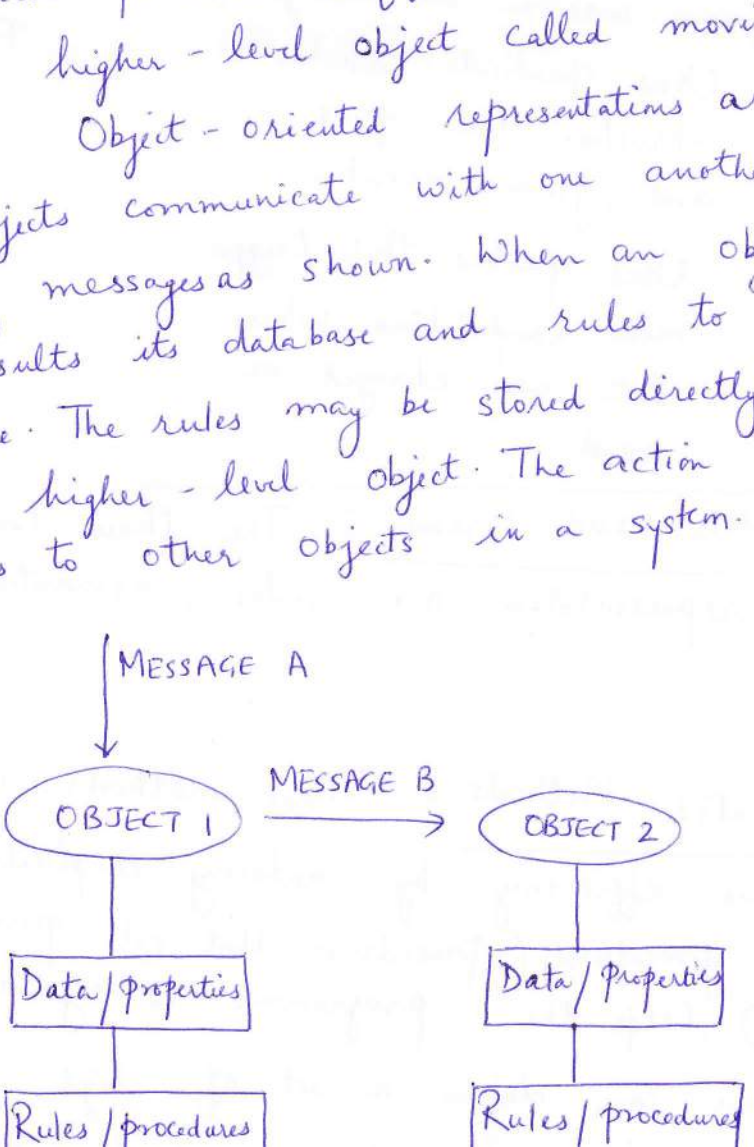
(3) Procedure-oriented Methods: These methods use subroutines that increases efficiency by reducing duplication in the code. Nested procedures (procedures that call procedures that call procedures) help the programmer to organize the program. The programmer can define a set of high-level procedures for describing actions the program should take and write

the program in a clear, concise way. Procedures provide great flexibility but makes explanation more difficult.

(4) Object-oriented Methods: These methods use Objects (called actors) that represent entities capable of exhibiting behavio e.g. In SWIRL which is an object-oriented air battle simulation system, the objects are penetrators (offensive aircraft), AWACS (air-borne) radars, ground radars, missile installations, missiles, filter centers (that interpret radar reports), fighters (defensive aircraft), fighter bases, command centers and ter

Each object has distinct properties associated with it and is situated in a network hierarchy that lets it inherit properties of higher-level objects. e.g. In SWIRL the objects penetrator, fighter, missile and AWACS all link to the higher-level object called moving object.

Object-oriented representations are unique in that all objects communicate with one another by sending and receiving messages as shown. When an object receives a messag it consults its database and rules to decide what action to take. The rules may be stored directly with the object or in a higher-level object. The action involves sending new messages to other objects in a system.



The objects and message passing provide a way to specify concurrent, asynchronous operations and helps to simulate many unrelated processes occurring at the same time.

(5) Logic-based Methods: These methods use predicate logic to control the analysis of a set of declarative clauses. Each clause has the form:

consequent :- antecedent-1, antecedent-2, ...... antecedent-n

where the antecedents are predicates that can be tested for their truth value and the consequent is a predicate that is true if its antecedents can be proved true. A logic program takes a goal and compares it with the consequents of the stored clauses. When it finds a match, it tries to prove the goal by considering the antecedents of the matched consequent as subgoals; when all the subgoals are shown to be true, the goal itself is proved. e.g. In PROLOG, the search is rigidly controlled by the interpreter the mechanism that analyzes and processes the clauses. PROLOG provides sophisticated pattern matching, has built relational data base facilities and efficient compilers.

(6) Access-oriented Methods: These methods use demons, procedure invoked when data are changed or read, to monitor programs or external devices. The demons act like problems connected to particular values of variables in a program. These methods make it easy to construct sophisticated visual displays for monitoring variables in a program. e.g. LOOPS programming language provide access-oriented methods in the form of graphical gauges. Gauges in LOOPS are defined as classes and driven by active values, for probles connected to the variables of a LOOPS program.

(1) Experimental Systems : Experimental systems exist as Knowledge engineering languages and system-building aids. The two experimental system-building aids that help to construct and refine expert systems are ROGET and SEEK.

(a) ROGET :

- It helps a domain expert design a Knowledge base for a diagnosis-type expert system.
- It queries the domain expert, asking questions that identify types of subproblems that the expert system must solve, the results or solutions the system must produce, the evidence or data required to solve the problem and the relationships between the data or facts of a case and its solution.
- It helps the expert to define and organize the rule building blocks, the primitive concepts or pieces of Knowledge to be used in rule formation.

(b) SEEK :

- It helps the domain expert or Knowledge engineer to refine tules during the development of a diagnostic-type expert system.
- The rules are represented in EXPERT language but expressed in a tabular format that divides the findings associated with a conclusion into two categories i.e. major and minor and the levels of confidence associated with a conclusion into three categories i.e. definite, probable and possible.
- It suggests ways to generalize or specialize rules based on use of Metarules i.e. rules about rules.

(2) Research Systems : Few research systems are as under :

(a) AL/X :

- It uses a rule - based representation combined with a semantic net that links rule components.
- It makes inferences based on a combination backward and forward chaining.
- It has been used by Intelligent Terminals Ltd. to explore the problem of diagnosing causes of automatic shut downs on oil production lines.

(b) LOOPS :

- It is used to create an expert assistant for designers of integrated digital circuits.
- It provides user with a great deal of flexibility since it is based on object - oriented, rule - based, procedure oriented and access - oriented representation methods.

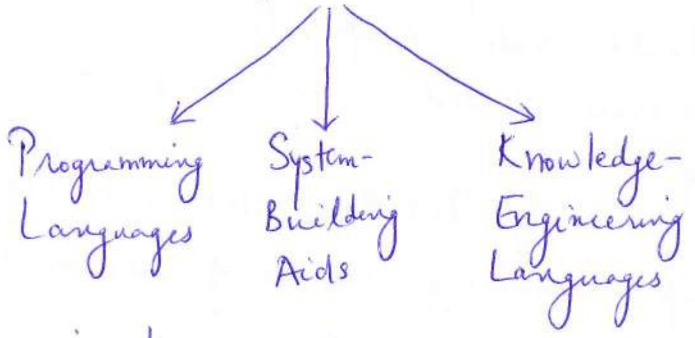
(c) MRS :

- It represents metaknowledge i.e. Knowledge about how an expert system reasons with its domain Knowledge.
- It is a Knowledge engineering language for rule - based and logic - based representation methods.
- It incorporates a flexible control scheme utilizing forward and backward chaining.
- It assists in diagnosing faults in computer hardware systems.

(d) SRL :

- It is a frame - based Knowledge engineering language developed for exploring issues of inheritance in frame systems.
- It provides the user with much flexibility in defining a representation system e.g. The user can define new relations and their inheritance semantics, new slots and new search specifications.
- It helps schedule jobs in a steam turbine blade point.

(3) Commercial Systems : Commercial Systems are divided into three categories :



Programming Languages

System-Building Aids

Knowledge-Engineering Languages

(a) Programming Languages :

Commercial versions of LISP and PROLOG are available. Other programming languages available are INTERLISP-D, SMALLTALK-80 , ZETALISP.

(b) System-Building Aids :

Commercial expert-system building aids are TIMM, RULEMASTER and EXPERT-EASE.

- These are designed to assist in Knowledge acquisition.
- The user defines the problem in terms of all possible decisions that can be made and the names and values of factors to consider in arriving at a decision.
- The system queries the user for examples describing conditions leading to each decision.
- From the examples, the system infers a procedure for solving the problem.

(c) Knowledge Engineering Languages :

Commercial Knowledge engineering languages are ART , KES, M.1 , OPS5 , DUCK.

- These are complete tools for expert system development, combining powerful languages with sophisticated support environments.