**Unit 5**

2a)I) Explain different phases in building an expert system.

Building an expert system requires a diverse range of knowledge, covering various aspects to ensure that the system can effectively emulate human expertise in a specific domain. The different types of knowledge needed for constructing an expert system include:

**1. Domain Knowledge:**

* **Definition:** Domain knowledge is the understanding of the specific subject matter or expertise that the expert system aims to emulate.
* **Role:** It forms the foundation of the expert system and includes facts, concepts, rules, and procedures relevant to the targeted domain.
* **Source:** Subject matter experts (SMEs) and domain specialists contribute domain knowledge through interviews, documentation, and existing literature.

**2. Meta-Knowledge:**

* **Definition:** Meta-knowledge is knowledge about how knowledge is organized, acquired, and applied within the expert system.
* **Role:** Helps the system manage, update, and adapt its knowledge base. It includes information about the reliability and sources of different pieces of knowledge.
* **Source:** Knowledge engineers and system designers contribute meta-knowledge during the development process.

**3. Inferential Knowledge:**

* **Definition:** Inferential knowledge involves the ability to draw conclusions, make inferences, and reason logically based on available information.
* **Role:** Enables the expert system to derive new knowledge from existing knowledge and make informed decisions.
* **Source:** Domain experts and knowledge engineers contribute inferential rules and logical structures.

**4. User Interaction and Interface Knowledge:**

* **Definition:** User interaction and interface knowledge involves understanding how users interact with the expert system and designing an effective user interface.
* **Role:** Ensures that the expert system is user-friendly and that interactions align with user expectations.
* **Source:** Human-computer interaction experts and user feedback contribute to this knowledge.

In summary, building an expert system requires a comprehensive set of knowledge types, ranging from the specific expertise within the domain to the procedural, inferential, and meta-knowledge necessary for effective system development and operation. Integrating these diverse forms of knowledge is crucial for creating intelligent and contextually aware expert systems.

ii) What is an inference engine? Describe forward and backward chaining mechanism used by an inference engine using an example.

1)It consists of inference mechanism and control strategy.

2)Inference means search through knowledge base and derive new knowledge.

3)It involves formal reasoning involving matching and unification similar to the one performed by human expert to solve problems.

4)Inference operates by using modus ponens rule.

5)Control strategy determines the order in which rules are applied.

There are mainly two types of control mechanism viz., forward chaining and backward chaining.

**Forward Chaining**

The execution cycle is

1. Match phase: Examine the rules to find one whose IF part is satisfied by the current contents of Working memory (the current state)

2. Conflict resolution phase: Out of all ‘matched’ rules, decide which rule to execute (Specificity, Recency, Fired Rules)

3. Act phase: Fire the applicable rule by adding to Working Memory the facts specified in the rule’s THEN part (changing the current state)– Repeat until there are no more rules to apply.

**Backward Chaining**

The execution cycle is

– Start with goal state.

– Check the conclusions of the rules to find all rules that can satisfy the top goal on the stack.

– Select one of these rules; the preconditions of the selected rule will be set as new goals on the goal stack.

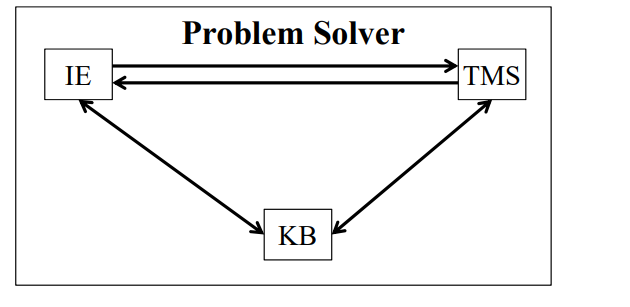
– System terminates if goal stack is empty or no more rules to try.

2b)i) Explain Justification-Based Truth Maintenance Systems with an example.

1)TMS is a structure that helps in revising beliefs and maintaining truth every time a new information is added. – Main job of TMS is to maintain consistency of the knowledge.

2)Truth maintenance system (TMS) works with inference engines for solving problems within large search spaces.

3)The TMS and inference engine both put together can solve problems where algorithmic solutions do not exist.



**Monotonic TMS**

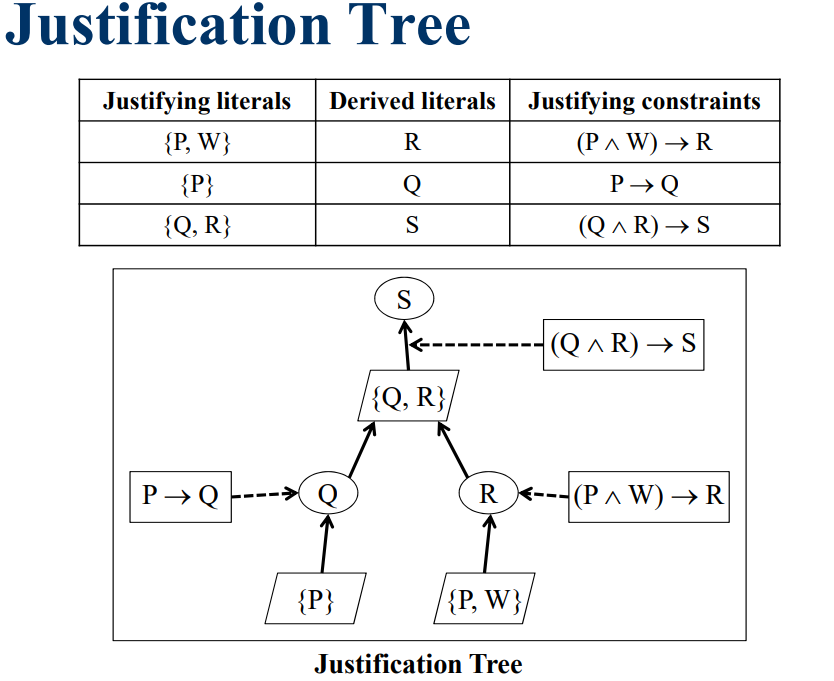
The most practical applications of monotonic systems using TMSs are qualitative simulation, fault diagnosis, and search applications.

Given a set of observations/propositions regarding some problem and a set of constraints on those propositions, a TMS can be used to ask questions about the consequences of the observations. Algorithms for monotonic systems can usually be used in non-monotonic systems as well but not vice

**Example** – Monotonic TMSν Suppose we are given the premise set ∑= {P, W} and the internal constraint set {P ◊ Q, (P Λ W) ◊ R, (Q ΛR) ◊S}.

TMS is able to derive S from these constraints and the premise set ∑.

TMS should provide the justifications of deriving S from constraints and premises.



**Contradiction Handling**

Most TMSs can inform the user that a given premise set is inconsistent with the internal constraints. – The function follows\_from(contradiction, ∑) returns Yes.

– The function justifying\_literals(contradiction, ∑) returns the set of literals resulting in the contradiction.

– The function justifying\_constraints(contradiction, ∑) returns the set of internal constraints resulting in the contradiction.

**Non-Monotonic TMS**

A non-monotonic TMS allows retraction of beliefs based on new facts.

– All birds fly (unless informed otherwise).

– All penguins are birds.

– Penguins cannot fly.

Non-monotonic TMSs can be seen as knowledge based management systems.

They maintain consistent set of beliefs for the inference engine.

The basic operation of a TMS is to attach a justification to a fact so as to maintain consistency

ii)Describe 4 applications of expert systems.

**Planning and scheduling**

– Creating plans for applying a series of chemical reactions

– Manufacturing process planning

– Airline flight scheduling

**Design and manufacturing**

– Gene cloning

– Integrated circuits layout designing

– Creation of complex organic modules

**Prediction**

– Weather prediction for rains, storms and tornado

– Prediction of crops and GDP

– Stock market

**Advisory services**

– Financial planning

– Tax planning

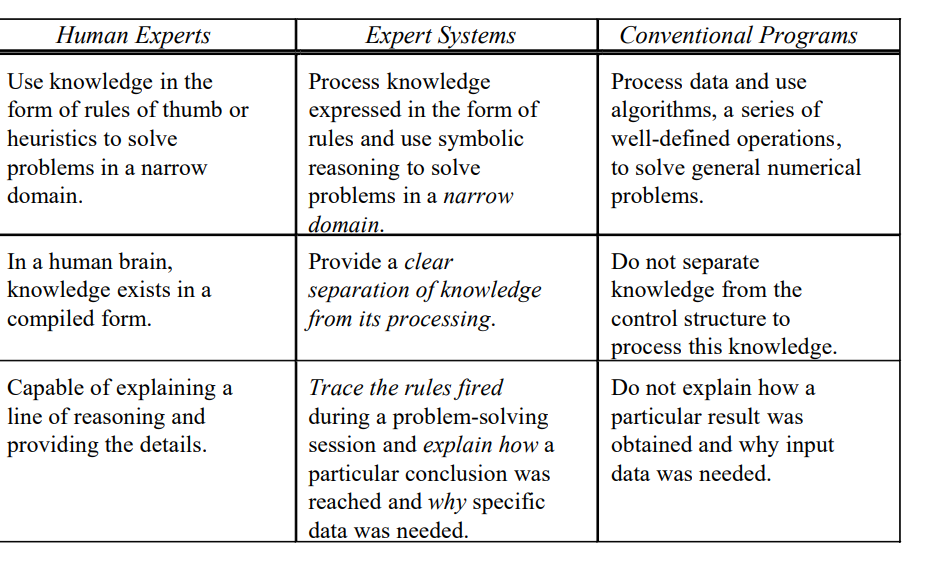
– Legal advice

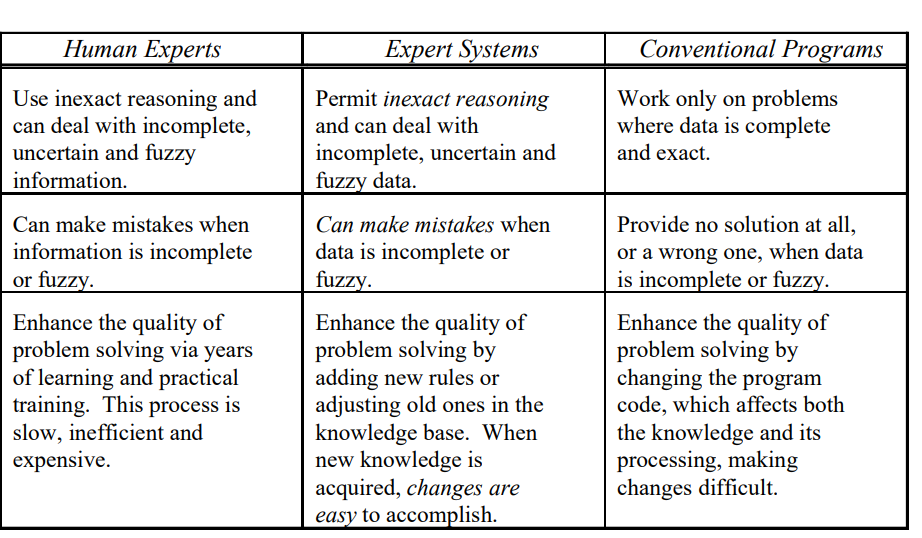
**Instruction/tutoring (GUIDON)**

-Tutoring to suit different students

– Student progress evaluation

3a)i)Compare and contrast Expert and Traditional Systems.





ii)Explain forward chaining and backward chaining in expert systems using the following example.

Rules:

R1: if A then B

R2: if C then E

R3: if A and C then F

R4: if B and E then D

Facts: A, C

Goal: D

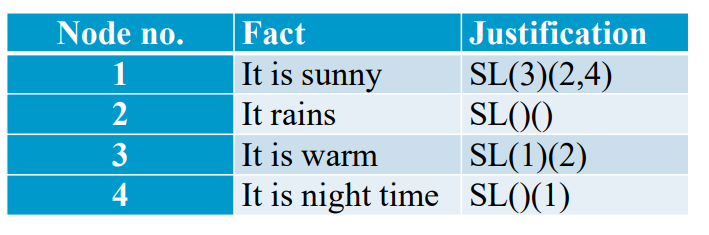
3b)i)Explain support-list and conditional proof in non-monotonic Truth Maintenance Systems.

**Support list (SL):**

It is defined as “SL(IN-node)(OUT-node)”, where INnode is a list of all IN-nodes (propositions) that support the considered node as true.

– Here IN means that the belief is true.

– OUT-node is a list of all OUT nodes for the considered node to be true. OUT means that belief is not true.



**Conditional Proof**

A belief may be justified on the basis of several other beliefs, by the conditional proof of one belief relative to other beliefs, or by the lack of belief in some fact.

Truth maintenance processing is required when new justifications change previously existing beliefs.

In such cases, the status of all beliefs depending on the changed beliefs must be re-determined. Dependency-directed backtracking is a powerful technique based on TMSs.

– It employs the recorded dependencies to precisely locate the hypotheses responsible for inconsistency using the conditional proof mechanism.

ii)Describe 4 expert system shells and tools.

**1. Rule-Based Shells:**

* **CLIPS (C Language Integrated Production System):**
  + **Description:** A popular open-source expert system tool that supports the development of rule-based systems. It has a powerful rule syntax and is widely used in various domains.
  + **Language:** CLIPS Language.
* **JESS (Java Expert System Shell):**
  + **Description:** A rule engine for the Java platform, based on the CLIPS syntax. It allows seamless integration with Java applications and supports complex rule-based reasoning.
  + **Language:** Java.
* **OPS5:**
  + **Description:** One of the earliest expert system shells, known for its efficiency in rule execution. It uses a production system model and has been influential in the development of subsequent shells.
  + **Language:** LISP-based.

**2. Frame-Based Shells:**

* **Kappa-PC:**
  + **Description:** A frame-based expert system shell that supports the development of knowledge-based systems. It uses frames to represent objects and their attributes, allowing for efficient knowledge representation.
  + **Language:** Prolog.

**3. Object-Oriented Shells:**

* **OPS83:**
  + **Description:** An extension of OPS5 that introduces object-oriented programming concepts. It allows the representation of knowledge using objects, classes, and inheritance.
  + **Language:** LISP-based.

**4. Hybrid Shells:**

* **CLIPSFuzzy:**
  + **Description:** An extension of CLIPS that incorporates fuzzy logic capabilities. It enables the representation of uncertain or imprecise information in the knowledge base.
  + **Language:** CLIPS Language.
* **Prolog+CG:**
  + **Description:** A hybrid system that combines Prolog for rule-based reasoning with conceptual graphs for knowledge representation. It supports both deductive and abductive reasoning.
  + **Language:** Prolog.