C.W
Tuesday

CHAPTER-2

* Knowledge Representation

To make an intelligent system we must have to train Knowledge base of it.

Data → Raw facts & figures.

Information → Processed ~~Information~~ Data

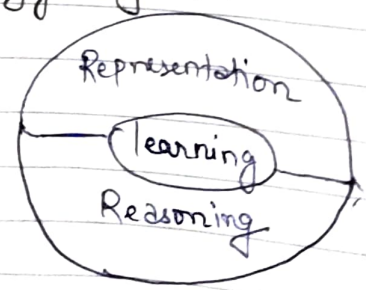
Knowledge → Processed Information

5
Swadesh
BSP
] Data ⇒ [5pen → Information
~~is having~~
Swadesh ~~is a~~] ⇒ [Swadesh is having 5pen] = Kn

According to sage, an intelligent system would have three capabilities:

→ Knowledge Representation.
→ Reasoning
→ Learning.

Morphology figure:



C.W
Wednesday

classmate
Date 11/01/23
Page 29

Roles Knowledge can play in an AI Program:

Types of Knowledge

① Essential Knowledge.

② Heuristic Knowledge.

Essential Knowledge → Describing search space
→ Goal state
→ Criteria to reach the goal state.

Heuristic Knowledge → Elimination of non-useful state
→ choosing a better intermediate state.

By further classifying the Knowledge, we can obtain:

1) Declarative Knowledge
↳ Facts
↳ Affirmations.

→ Initial state, Goal state

→ Ex. Log table

2) Procedural Knowledge.
↳ Description of performing something.

→ Procedure to reach goal state from initial state.

→ Using a log table

We can further classify into:

Domain Independent Knowledge
↳ General Knowledge
↳ Common Sense.

Domain Specific Knowledge
↳ Specified Knowledge

2½ 2½ 2 2½ 2½ 2½

↓ state space problem

Properties for Knowledge Representation Scheme

1) Acquistinal Efficiency. — It should be able to understand new problems based on knowledge base which is previously existing.

2) Representational Adequacy - It should have proper symbol & semantics to represent (portray) all the knowledge related to a problem domain.

3) Inferential Adequacy — From the pre-existing knowledge base new deductions/ inferences should be able to form.

4) Inferential Efficiency — When there are multiple states, a state may exist which may have exponential number of states ~~because there is an~~. Now we must be able to select a state which bring us to polynomial time (no of states) and hence giving optimal solution.

Knowledge Representation Scheme classification

- Logical Representation Scheme
  - ↳ First Order Predicate Logic
    - ↳ Prolog language.
- Procedural Representation Scheme
  - ↳ If-Then
    - ↳ Production System
- Network Representation Scheme
  - ↳ Semantic Net.
- Structured Representation Scheme
  - ↳ Frames
  - ↳ Scripts

Slot & Filler Structures

- Weak S&F stru.
  - ↳ Semantic Net.
  - ↳ Frames.
- Strong S&F Struct.
  - ↳ Script.

Proposition

Adarsh is a friend of PAVAN

↳ ADA – FRI – PAV → T / F

↳ T|F

↳ Advantage : It is simple

Problem: Terms may repeat.

Now, e.g.

Adarsh is a friend of pavan.

Adarsh is a friend of Shiva,

↳ ADA – FRI – PAV – ADA – FRI – SHV

Solution: First Order Predicate Logic.

** First Order Predicate Logic (FOPL)

1. Predicate Name
2. Function
3. Variable
4. Constant
5. Connectives
6. Quantifiers

Term

friend (Adarsh, Pavan).

↓ Any number of Terms

Connectives:

- ¬ → NOT
- ∧ → And
- ∨ → OR

∀ → for all

∃ → There exists

Quantifiers

- ∀ → for all
- ∃ → Existential Quantifier

Universe Quan.

Variable – Terms present within parentheses of predicate

– n number of variable

Function – special type of procedure which return value.

– many type of procedure are there.

Predicate – maps 'n' terms with their truth value.

Convert to Clause form

1) Eliminate → using the fact that $a \rightarrow b$ is equivalent to $\neg a \vee b$

2) Reduce the scope of each ¬ to a single term, using the fact that $\neg(\neg p) = p$, de Morgan's Laws [which say that $\neg(a \wedge b) = \neg a \vee \neg b$ and $\neg(a \vee b) = \neg a \wedge \neg b$ and the standard correspondes between quantifiers $[\neg \forall x : P(x) = \exists x : \neg P(x) = \forall x : \neg P(x)]$

3) Standardize variables so that each quantifier binds a unique variable. Since variables are just dummy names, this process cannot affect the truth value of wff (well formed formula).

4) Move all quantifiers to the left of the formula without changing their relative order. This is possible since there is no conflict among variable names.

5) Eliminate existential quantifiers.

6) Drop the Prefix.

7) Convert the matrix into a conjunction of disjuncts.

8) Create a separate clause corresponding to each conjunct.

9) Standardize apart the variables in the set of clauses generated in step 8.

In Book → Man (Marcus)
In Prolog → man(Marcus).

1) Marcus was a man.
   ↳ man(Marcus)
2) Marcus was a Pompeian
   ↳ pompeian(Marcus)
3) All pompeians were romans.
   ∀x : pompeian(x) ⟶ romans(x).
4) Caesar was a ruler :
   ruler(caesar).
5) All Romans were either loyal to Caesar or hated him.
   ∀x : roman(x) ⟶ loyalto(x, Caesar) ∨ hate(x, caesar)
6) Everyone is loyal to someone.
   ∀x : ∃y : loyalto(x, y)
7) People only try to assassinate rulers they are not loyal to.
   ∀x : ∀y : man(x) ∧ ruler(y) ∧ tryassassination(x,y) → ¬ loyalto(x,y)
8) Marcus tried to assassinate caesar.
   tryassassinate (Marcus, Caesar)

⇒ man (Marcus)
pompeian (Marcus)
∀x : pompeian (x) ⟶ romans(x)
ruler (caesar)
∀x : roman (x) ⟶ loyalto (x, caesar) ∨ hate (x, caesar)
∀x : ∃y : loyalto (x, y)
∀x : ∀y : man(x) ∧ ruler(y) ∧ tryassassinate (x,y) ⟶ ¬ loyalto(x, y)
tryassassinate (Marcus, Caesar)

How to convert to casual form

First Order : Quantification happens at level of variables.
Second Order : Quantification of function variable.
Third Order : Quantification of variable present within function of a function.
and so on.

Step I: Applying Rule 1,

1. marcus(man) [No change]
2. pompeian(marcus) [No change]
3. $\forall x: \neg$ pompeian(x) $\vee$ romans(x)
4. ruler(caesar) [No change]
5. $\forall x: \neg$ roman(x) $\vee$ loyalto(x, caesar) $\vee$ hate(x, caesar)
6. $\forall x: \exists y:$ loyalto(x, y) [No change]
7. $\forall x: \forall y: \neg$(man(x) $\wedge$ ruler(y) $\wedge$ tryassassination(x, y)) $\vee \neg$ loyalto(x, y)
8. tryassassinate(marcus, caesar) [No change]

Step II: Applying Rule 2,

1. marcus(man) [N.C]
2. Pompeian(marcus) [N.C]
3. $\forall x: \neg$ pompeian(x) $\vee$ romans(x) [N.C]
4. ruler(caesar) [N.C]
5. $\forall x: \neg$ roman(x) $\vee$ loyalto(x, caesar) $\vee$ hate(x, caesar) [N.C]
6. $\forall x: \exists y:$ loyalto(x, y) [N.C]
7. ~~$\forall x: \forall y: \neg$(man(x) $\wedge$ ruler(y) $\wedge$ tryassassination(x, y))~~
7. $\forall x: \forall y: \neg$ man(x) $\vee \neg$ ruler(y) $\vee$ tryassassination(x, y) $\vee \neg$ loyalto(x, y)
8. tryassassinate(marcus, caesar) [N.C]

Step III: Applying Rule 3,

1. No change.
2. (N.C)
3. $\forall x_1: \neg$ pompeian(x_1) $\vee$ romans(x_1)
4. ruler(caesar) (N.C)
5. $\forall x_2: \neg$ roman(x_2) $\vee$ loyalto(x_2, caesar) $\vee$ hate(x_2, caesar)
6. $\forall x_3: \exists y:$ loyalto(x_3, y)

(7) $\forall x_4 : \forall y_1 : \neg man(x_4) \vee \neg ruler(y_1) \vee tryassassination(x_4, y_1) \vee \neg loyalto(x_4, y_1)$

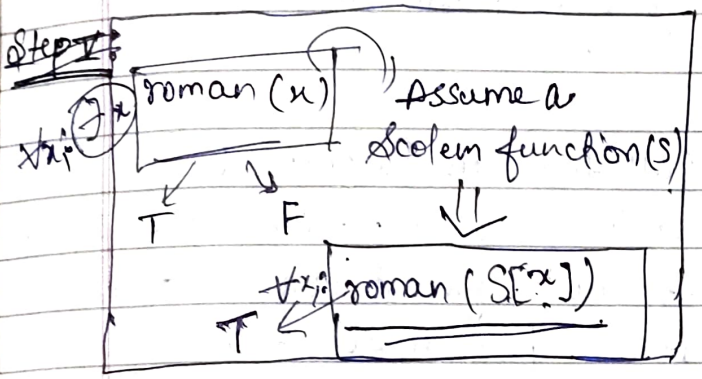
(8) N.C.

Step IV: Applying Rule 4:

N.C. in any (WFF).

To remove existential qualifier, we use ~~Skolemisation~~ Skolemisation.

In skolemisation, we assume that there exist a function (called skolem function) that always return the value true for a predicate.



Step V: Applying Rule 5:

(1) N.C.

(2) N.C.

(3) N.C.

(4) N.C.

(5) N.C.

(6) $\forall_3$: loyalto. $(x_3 : fl)$

(7) N.C.

(8) N.C.

Step VI: Applying Rule 6.

(1), (2). → N.C.

(3) $\neg pompeian(x_1) \vee romans(x_1)$

(4) → N.C.

(5) $\neg roman(x_2) \vee loyalto(x_2, caesar) \vee hate(x_2, caesar)$

(6) loyalto $(x_3 : fl)$

(7) $\neg man(x_4) \vee \neg ruler(y_1) \vee tryassassination(x_4, y_1) \vee loyalto(x_4, y_1)$

(8) N.C.

RESOLUTION

1) Convert all the statements of F to clause form.
2) Negate P and convert the RESULT to clause form. Add it to the set of clauses obtained in 1.
3) Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.

(a) Select two clauses. Call these the Parent Clauses.
(b) Solve them Together. The Resolvent will be the disjunction of all the literals of both parent clauses with appropriate substitutions performed and with the following exceptions:

If there is one pair of literals T_1 and $\neg T_2$ such that one of the parent clauses contains T_2 and the other contains T_1 and if T_1 and T_2 are unifiable, then neither T_1 nor T_2 should appear in the resolvent. We call T_1 and T_2 complementary literals. Use the substitution produced by the unification to create the resolvent.
If there is more than one pair of complementary literals only one pair should be omitted from the resolvent.

(c) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available in the procedures.

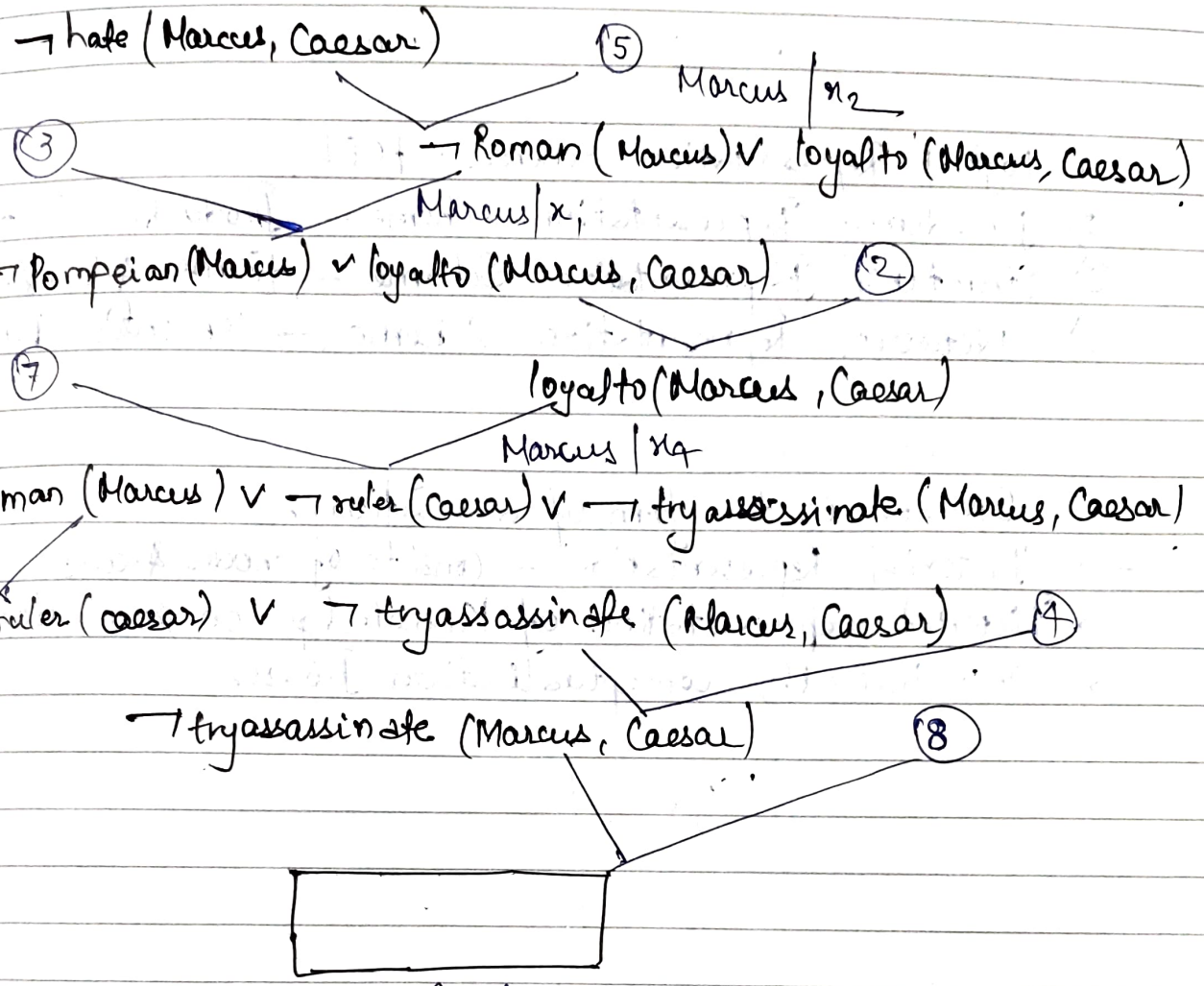


Figure : A Resolution Proof.

Why we create Knowledge Base?

↳ To get interpretation of our query or (inferencing)

Assumption that Reverse is true is called "Refutation"

To proof something: two algorithms work parallely

① Resolution

② Unification.

Two tree in ~~AI~~ programming: ① Search Tree

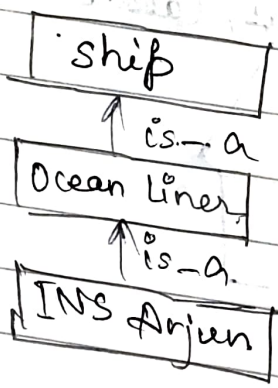
② Proof Tree

* Four Knowledge Representation Scheme

1) Logical Representation Scheme – FOPL
2) Procedural Representation Scheme – Production System
3) Structural Representation Scheme – Frames & Scripts
4) Network Representation Scheme – Semantic Net

Semantic NET

→ Operates in the form of association.
→ Pictorial Representation – consists of nodes & arcs
→ Pictorial representation of Knowledge on some domain
→ It is basically conceptualisation Process.



C.W
Tuesday

"In the Knowledge, Lies the Power"

Date 31/01/22

CHAPTER-3
EXPERT SYSTEMS

* Expert System is a Knowledge ~~Expensive~~ Intensive Application. that solves problem in specific domain.

Difference b/w Expert Systems & Human.

1) No age of Expert System, human beings are mortal.
2) Expert Systems are accessible around the clock.
3) AI Expert Systems provide documentation completely.
4) Consistent response by AI expert systems.
5) Humans have five Sense Organs to change physical parameters but Expert Systems can have multiple physical measures

Features of Expert System

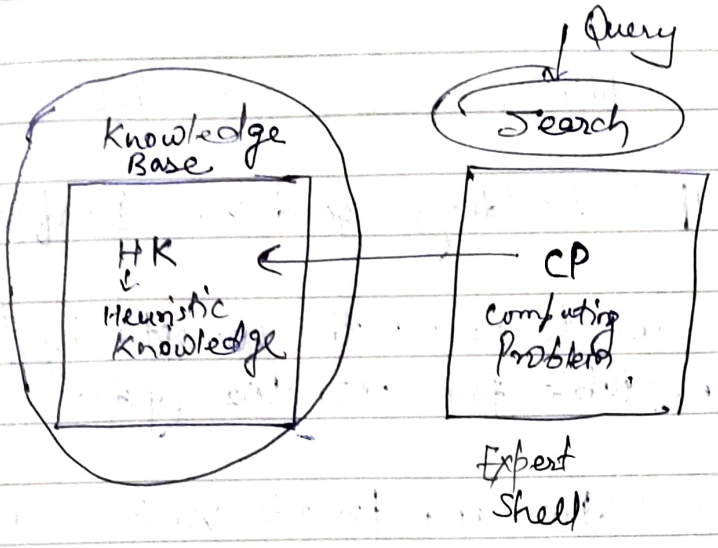
- Knowledge Based Problem Solving
- Intensity of problem solving greater than or equivalent to Human Beings.
- Embibing Heuristic model in Expert System (so that it can access minute pieces of information about domain).

Note: In AI System, majorly two modules:
- Knowledge Base
- Search Engine

In Expert System, five major components.

C.W
Wednesday

Date 01/02/23
Page 43



- Symbolic Processing
- A feature required that can add or remove new information in the Knowledge base.
- It can filter and handle data inconsistencies, irrelevancy of data.
- Finance
- ~~Indivi~~ It should solve all queries of individuals in related to domain.
- Meta Knowledge.
- Good User Interface.

* Tools Available for development of Expert System.

Four categories of tools:

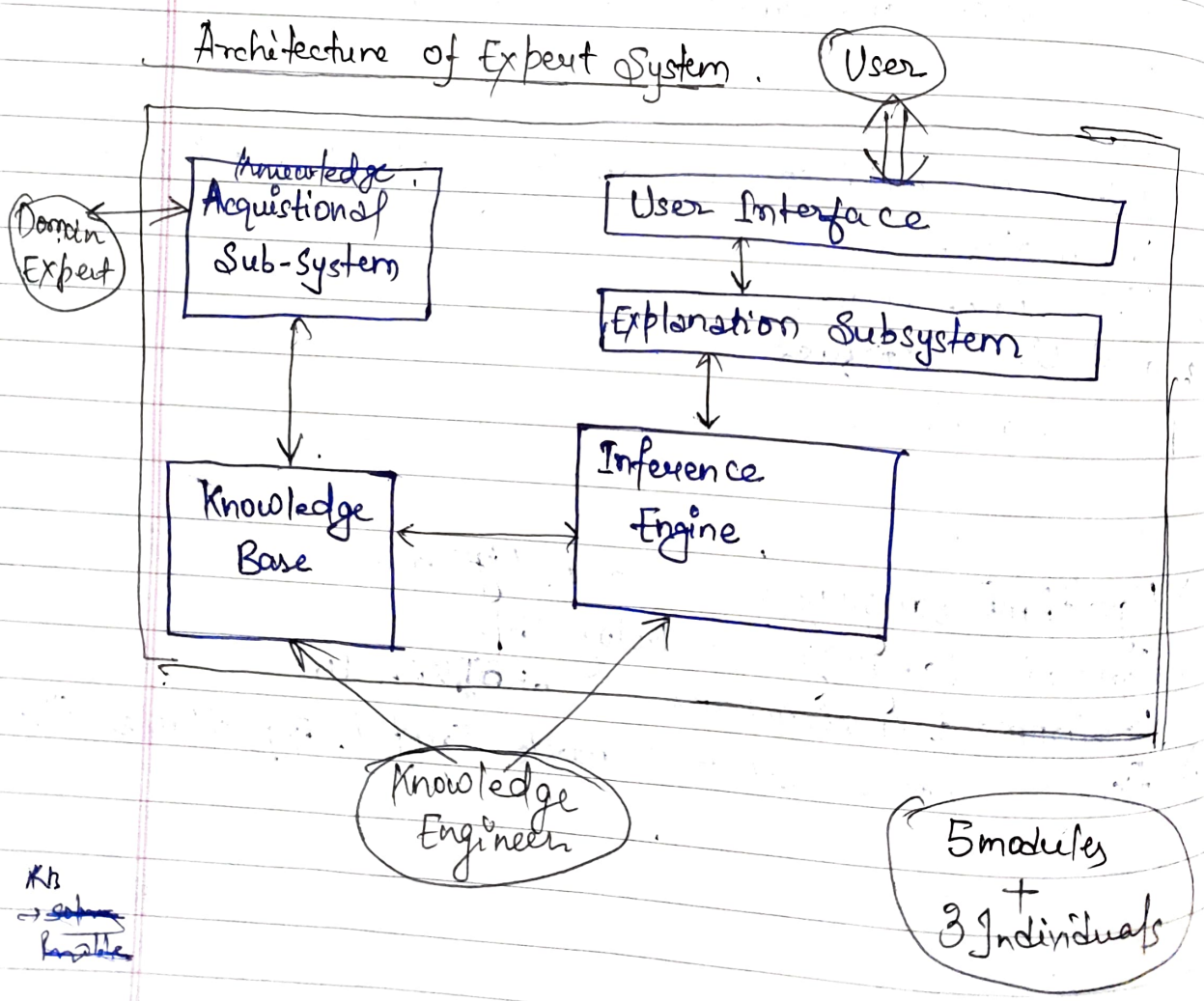
1) Algorithmic Languages - C, C++, Python
2) Symbolic Languages - Prolog, Lisp.
3) Development Environment - ARCTIKEE, LOOPS
4) Expert System Shell - Leonardo, Krystal, ExpertRule, XI+

Note: Expert System Shell is inference engine or control interpreter

Features of Tools which we need to select AI:

- Tools should help in building good User Interface.
- Tools should be able to convert the conceptualized idea into the implementable form onto the system.
- Tools should be scalable, and provide various support development of Expert System.
- An Expert System Shell should be good to search relevant information from Knowledge Base.

Architecture of Expert System.



KBS

Starting from Any Module,

Knowledge Base
- Expert System is incomplete without it.
- Constructed using one of the Knowledge Representation Scheme.
- It should be then further developed with any one of the tool.
- It can store information about a particular knowledge domain.
- It is constructed by Knowledge Engineer by consulting various domain experts.

Inference Engine
- Control program resides in it.
- User problem is solved by search procedure of expert system inference engine.

Explanation Engine
- Keeps track of procedure performed in Inference Engine.
- It helps in providing explanation of the solution obtained by Expert System.

User Interface
- Natural Language Processing may be present
- Provides liberty to user to interact with user interface in better way.

Knowledge Acquisition Sub-system
- Worked upon by Domain Expert by recognizing if a new knowledge comes.
- If new knowledge comes then domain expert consults with Knowledge Engineer.
- Domain Expert may not be a computer literate person.
- Converts the knowledge required into Expert System understandable form.

Advantages of Expert System.

Steps in formation of Expert System.

(1) Big Knowledge Base : Archive of Knowledge, disseminate Knowledge.

(2) Transparency : It does not only tell solution but also. when and how of solution. (Human Window)

(3) ~~Data~~ New Knowledge can easily be entered into it directly by domain expert.

(4) Consistent : Response time is consistent.

(5) Accessbility : Accessible solution and ~~master~~ expert system

(6) Creation & Duplication : Takes considerably less amount of time.

Expert System plays three roles:
(1) Problem Solver.
(2) Archive of Knowledge.
(3) Human Window

(7) Merging of different ~~come~~ technologies, so we may face some problems

(8) Human beings prioritise task over each other. Expert system are designed in a similar way to prioritise tasks.

Limitations of Expert System

1) Many industries still don't use expert system, it may be due to lack of domain experts.
   e.g. Metallurgy, Medical Diagnosis, etc.

Note: Level of Granularity : Brittleness.
based upon

2) Level of features of granularity level of knowledge. We can solve problem (More knowledge more feature or vice versa).

3) Limitations of Knowledge Representation System are imposed upon Expert System.

4) Lack of Meta Knowledge.

5) Lack of Flexibility

6) Lack of Understability - Level of understability depends on how it has been programmed.

7) Return on Investment Analysis must be made before creating an Expert System.

8) Validation

9) Knowledge Representation Scheme are fragile at their boundaries It may help to validate the contradictory ~~to answer~~ knowledge not.

C.W
Tuesday

07/02/23

10) Lack of conversation in Expert system due to presence or absence of Knowledge Acquisition Skills.

11) Lack of Common Sense.

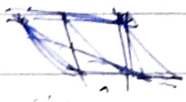
Expert System	Software System
1) Created by Knowledge Engineer	Created by Software Engineer
2) It is AI application	It is not AI application
3) It works on Knowledge Base	It works on database.
4) It works on Heuristic	It works on Algorithms.
5) It works on Inference Engine.	It has translators like Compilers & interpreters.
6) Created Using Special development tools.	Created using Programming Languages

Knowledge Base	Data Base
1) It stores knowledge (higher level of abstraction)	It stores raw facts & figures i.e data.
2) Here operation are performed at level of class.	Here operations are performed on objects themselves.
3) It contains information in the form of knowledge Representation scheme.	Here data is stored in form of Relational & Non Relational database.

4) Here knowledge Base is used for decision making, planning etc.

Here database is used for basic Operations. (Addition, Subtraction, etc)

* Life Cycle Of Expert System



Steps:

1) Identification :
   - Requirement from Domain Experts
   - Requirement ~~from~~ of resources
   - Feasibility check or ROI study (ROI → Return on Investment)

2) Selection of Tools :
   - Types of tools prescribed
   - Selection of properties.
   ~~tools based on~~

3) Designing of solution Structure
   - Fixing a level of Granularity
   - Selection of Knowledge Representation Scheme.
   - Prototype Creation (first of its type)
   - Large Scale Implementation.

4) Full Scale Implementation
   - Easy parts implemented ~~first~~
   - Hard parts implemented later

5) Deployment : Testing is also done here parallely (Testing happens in each phase).

6) Maintenance : • Maintenance Person.

Father of Expert System
↳ Edward Albert Feigenbaum
↳ Regarded due to creation of DENDRALS.

Applications of Expert Systems

① Control & Monitoring System
- PRANT → is one control & monitoring system. | Control is objective of monitoring.
  ↳ It is used on assembly line for monitoring the amou of components.
- REACTOR → Used in nuclear reactors.

② Debugging System
- To identify malfunction of codes in:
  - Code
  - Electronics
  - Blue Box – Identifies malfunctioning of compon in machine
  - ~~Design~~.

③ Designing
- VLSI Design – Have many IC components.
- WAR affairs – Using map locations, etc.

④ Diagnosis
- Most used area.

⑤ Instruction
- Teaching skills, etc.

⑥ Computer Aided Design (CAD)
⑦ Administering Anaesthesia
⑧ Boiler
⑨ Sensor Interpretation of Physical Environment
⑩ Planning System
⑪ Prediction System – Weather forecasting, Economy fore