20CS504/PE120IT504/PE1

Hall Ticket Number:

III/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

December, 2024 — **Computer Science and Engineering**

Fifth Semester — **Artificial Intelligence**

Time: Three Hours — **Maximum:** 70 Marks

Answer question 1 compulsorily. **(14X1 = 14 Marks)**
Answer one question from each unit. **(4X14 = 56 Marks)**

			CO	BL	M
1	a)	What is artificial Intelligence?	CO1	L1	1M
	b)	Briefly explain about the Turing test?	CO1	L2	1M
	c)	Write the purpose of a heuristic function guide the path to a goal?	CO1	L1	1M
	d)	Formulate map colouring problem as constraint satisfaction problem?	CO1	L3	1M
	e)	Write any two logical propositions used in AI?	CO2	L1	1M
	f)	Represent the following statement in first-order logic? Gershwin did not write "All kings are persons."	CO2	L3	1M
	g)	When is forward chaining more suitable than backward chaining?	CO2	L4	1M
	h)	What is the purpose of an ontology in knowledge representation?	CO3	L1	1M
	i)	Define unification in FOL.	CO3	L3	1M
	j)	Using CD theory represent the proposition “john ran”?	CO3	L3	1M
	k)	Describe the Blocks World problem?	CO4	L1	1M
	l)	What is an expert system in AI?	CO4	L1	1M
	m)	Present the main components of expert system shell?	CO4	L1	1M
	n)	List the advantages of rote learning?	CO4	L1	1M

Unit-I

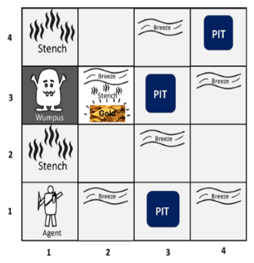
2	a)	List the key disciplines that contribute to the foundations of AI.	CO 1	L1	7M
	b)	Write Uniform cost search algorithm and explain its primary purpose in artificial intelligence.	CO 1	L2	7M

(OR)

3	a)	Explain the interaction between an agent and its environment with examples of real world applications.	CO 1	L1	7M
	b)	Illustrate how an AND-OR search tree can be used to solve the problem of an erratic vacuum world.	CO 1	L3	7M

Unit-II

4	a)	Write a short on Knowledge-Based Agent. Write a PEAS description for Wumpus World given below:	CO 2	L1	7M



	b)	Create about the steps involved in resolution technique with any example	CO 2	L3	7M

(OR)

5	a)	Construct a FOL representation for the following sentences a. Horses, cows, and pigs are mammals. b. An offspring of a horse is a horse. c. Bluebeard is a horse. d. Bluebeard is Charlie's parent. e. Offspring and parent are inverse relations. f. Every mammal has a parent.	CO 2	L3	7M
	b)	List the basic steps involved in the unification algorithm with example.	CO 2	L1	7M

P.T.O

Unit-III

6	a)	List and explain various primitive actions and represent the following sentence in conceptual dependency "***John purchased the house from mike***"	CO 3	L1	7M
	b)	Explain the importance of semantic net structure in AI and represent semantic net for the following sentence. "***Every dog has bitten every mail carrier***"	CO 3	L3	7M
		(OR)			
7	a)	Represent the following sentences using conceptual dependency representation. John ate an apple. Mary gave a book to john. Mary hit the ball with a bat. The teacher explained the concept to the students.	CO 3	L3	7M
	b)	Discuss the components of a planning system in artificial intelligence.	CO 3	L2	7M
		Unit-IV			
8	a)	Define learning in the context of artificial intelligence and explain its significance with ***winston's learning program***.	CO 4	L1	7M
	b)	Explain in brief about explanation based learning (EBL) with example.	CO 4	L3	7M
		(OR)			
9	a)	Explain the role of knowledge acquisition in the development of expert systems.	CO 4	L2	7M
	b)	Define expert system and write the importance of MCYCN to diagnose common diseases based on symptoms.	CO 4	L3	7M

III/IV B.Tech (Regular) DEGREE EXAMINATION

December, 2024 **Computer Science & Engineering**
Fifth Semester **Artificial Intelligence(20CS504)**
Time: Three Hours **Maximum : 70 Marks**

SCHEME OF EVALUATION

Prepared By : **J.KUMARARAJA**
Assistant Professor,
Dept. of CSE,
BEC, BAPATLA.

[illegible]

(Signature)

SCHEME OF EVALUATION

Allocation of Marks

UNIT – I

2. a) List the key disciplines that contribute to the foundations of AI.

[Consider Any Six/Seven Points]	**–**	**7 Marks**

2. b) Write Uniform cost search algorithm and explain its primary purpose in artificial intelligence.

Explanation	**–**	**4 Marks**
Algorithm	**–**	**3 Marks**

(or)

3. a) Explain the interaction between an agent and its environment with examples of real world applications.

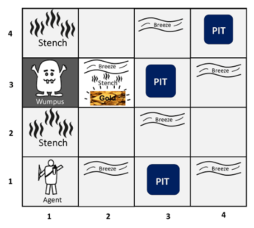
Explanation	**–**	**4 Marks**
Example	**–**	**3 Marks**

3. b) Illustrate how an AND-OR search tree can be used to solve the problem of an erratic vacuum world.

Explanation	**–**	**4 Marks**
Example	**–**	**3 Marks**

UNIT – II

4. a) Write a short on Knowledge-Based Agent. Write a PEAS description for Wumpus World given below:



Explanation	**–**	**4 Marks**
Example/PEAS Description	**–**	**3 Marks**

4. b) Create about the steps involved in resolution technique with any example

Explanation	**–**	**4 Marks**
Example	**–**	**3 Marks**

(or)

5. a) Construct a FOL representation for the following sentences

a. Horses, cows, and pigs are mammals. **b. An offspring of a horse is a horse.**
c. Bluebeard is a horse. **d. Bluebeard is Charlie's parent.**
e. Offspring and parent are inverse relations. **f. Every mammal has a parent.**

Explanation	**–**	**4 Marks**
Example	**–**	**3 Marks**

5. b) List the basic steps involved in the unification algorithm with example.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

<u>UNIT – III</u>

6. a) List and explain various primitive actions and represent the following sentence in conceptual dependency
"John purchased the house from mike"

Explanation	–	**4 Marks**
Example	–	**3 Marks**

6. b) Explain the importance of semantic net structure in AI and represent semantic net for the following sentence. "Every dog has bitten every mail carrier"

Explanation	–	**4 Marks**
Example	–	**3 Marks**

(or)

7. a) Represent the following sentences using conceptual dependency representation.
John ate an apple.
Mary gave a book to john.
Mary hit the ball with a bat.
The teacher explained the concept to the students.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

7. b) Discuss the components of a planning system in artificial intelligence.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

<u>UNIT – IV</u>

8. a) Define learning in the context of artificial intelligence and explain its significance with winston's learning program.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

8. b) Explain in brief about explanation based learning (EBL) with example.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

(or)

9. a) Explain the role of knowledge acquisition in the development of expert systems.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

9. b) Define expert system and write the importance of MCYCN to diagnose common diseases based on symptoms.

Explanation	–	**4 Marks**
Example	–	**3 Marks**

- - - - -

III/IV B.Tech (Regular) DEGREE EXAMINATION

December, 2024 — **Computer Science & Engineering**
Fifth Semester — **Artificial Intelligence(20CS504)**
Time: Three Hours — **Maximum : 70 Marks**

SCHEME OF EVALUATION

Answer Questions No. 1 compulsorily ***(14 X 1 = 14 Marks)***
Answer ONE question from each unit. ***(4 X 14 = 56 Marks)***

1. Answer all questions **(1 X 10 = 10 Marks)**

a) What is artificial Intelligence?

Definition: Artificial Intelligence is the study of how to make computers do things, which, at the moment, people do better.

b) Briefly explain about the Turing test?

The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence.

c) Write the purpose of a heuristic function guide the path to a goal?

- ✓ Every node in a search space has an evaluation function (heuristic function) associated with it. A heuristic function value h(n) on each node indicates how the node is from the goal node.
    - ✓ ***Note that Evaluation function=heuristic cost function (in case of minimization problem) OR objective function(in case of maximization).***

d) Formulate map colouring problem as constraint satisfaction problem?

A problem is solved when each variable has a value that satisfies all the constraints on the variable. A problem described this way is called a constraint satisfaction problem, or CSP.

We are given the **task of coloring each region** either **red, green, or blue** in such a way that **no neighbouring regions have the same colour**.

To formulate this as a CSP, we define the variables to be the regions

$$\boxed{X = \{WA, NT, Q, NSW, V, SA, T\}.}$$

The domain of each variable is the set Di = {red , green, blue}.

e) Write any two logical propositions used in AI?

De Morgan's Laws:

$\neg(P \land Q) \equiv \neg P \lor \neg Q$

$\neg(P \lor Q) \equiv \neg P \land \neg Q$

f) Represent the following statement in first-order logic?
Gershwin did not write "All kings are persons."

$$\forall x\ King(x) \Rightarrow Person(x)\ .$$

g) When is forward chaining more suitable than backward chaining?

Forward chaining is more suitable than backward chaining when you have a set of initial facts and need to explore all possible conclusions based on those facts.

h) What is the purpose of an ontology in knowledge representation?

The purpose of an ontology in knowledge representation is to provide a structured framework for representing knowledge.

i) Define unification in FOL.

Unification in first-order logic (FOL) is the process of finding a substitution for variables in different terms to make them identical.

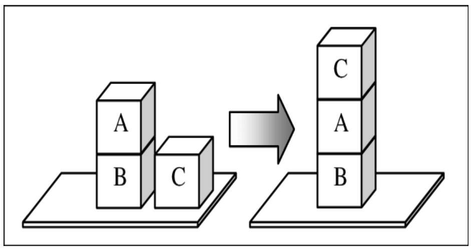
j) Using CD theory represent the proposition "john ran"?

$$\boxed{\text{John} \overset{p}{\Longleftrightarrow} \text{PTRANS}}$$

.

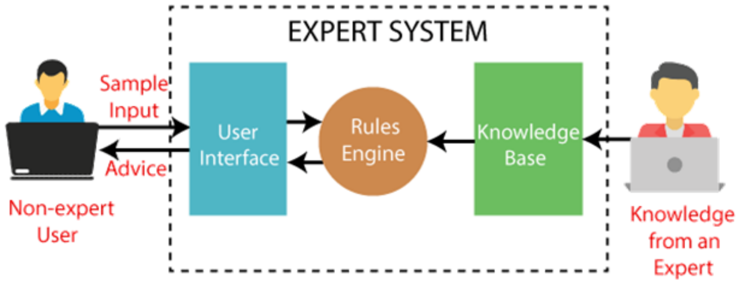
k) Describe the Blocks World problem?

The Blocks World problem is a classic artificial intelligence planning domain where an agent is tasked with arranging a set of blocks, represented as simple geometric shapes, into specific stacks by moving only one block at a time, either placing it on the table or on top of another block, with the goal of reaching a predefined desired configuration (the "goal state") from an initial arrangement (the "initial state") - essentially simulating the manipulation of physical blocks on a table with limitations on how many blocks can be moved simultaneously; it's often used to test and demonstrate planning algorithms due to its simplicity while still capturing key aspects of problem-solving with constraints.



l) What is an expert system in AI?

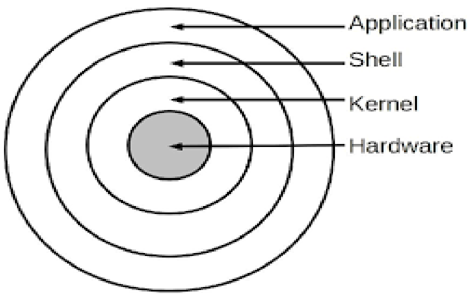
An expert system is a computer program that uses artificial intelligence (AI) to mimic the judgment and behavior of a human expert in a specific field. Expert systems are designed to complement human experts, not replace them.



m) Present the main components of expert system shell?

The shell is the layer of programming that understands and executes the commands a user enters.
In some systems, the shell is called a command interpreter

- ✓ **User Interface**
- ✓ **Inference Engine**
- ✓ **Knowledge Base**



n) List the advantages of rote learning?

Caching has been used in AI programs to produce some surprising performance improvements. Such caching is known as **rote learning**.

Rote Learning is basically *memorization.*

- Saving knowledge so it can be used again.
- Retrieval is the only problem.
- No repeated computation, inference or query is necessary.

- **Quick Memorization**: One of the primary advantages of rote learning is its efficiency in quickly memorising information. ...
- **Simplifies Complex Information**: Rote learning simplifies complex information by breaking it down into manageable chunks. ...
- **Foundation for Further Learning**

UNIT – I

2. a) List the key disciplines that contribute to the foundations of AI.

Foundations of Artificial Intelligence:

Philosophy

- ✓ e.g., foundational issues (can a machine think?), issues of knowledge and believe, mutual knowledge

Psychology and Cognitive Science

- ✓ e.g., problem-solving skills

Neuro-Science

- ✓ e.g., brain architecture

Computer Science And Engineering

- ✓ e.g., complexity theory, algorithms, logic and inference, programming languages, and system building.

Mathematics and Physics

- ✓ e.g., statistical modeling, continuous mathematics,

Statistical Physics, and Complex Systems. Sub Areas of AI:

Game Playing

Deep Blue Chess program beat world champion Gary Kasparov

Speech Recognition

PEGASUS spoken language interface to American Airlines' EAASY SABRE reseration system, which allows users to obtain flight information and make reservations over the telephone. The 1990s has seen significant advances in speech recognition so that limited systems are now successful.

Computer Vision

Face recognition programs in use by banks, government, etc. The ALVINN system from CMU autonomously drove a van from Washington, D.C. to San Diego (all but 52 of 2,849 miles), averaging 63 mph day and night, and in all weather conditions. Handwriting recognition, electronics and manufacturing inspection, photo interpretation, baggage inspection, reverse engineering to automatically construct a 3D geometric model.

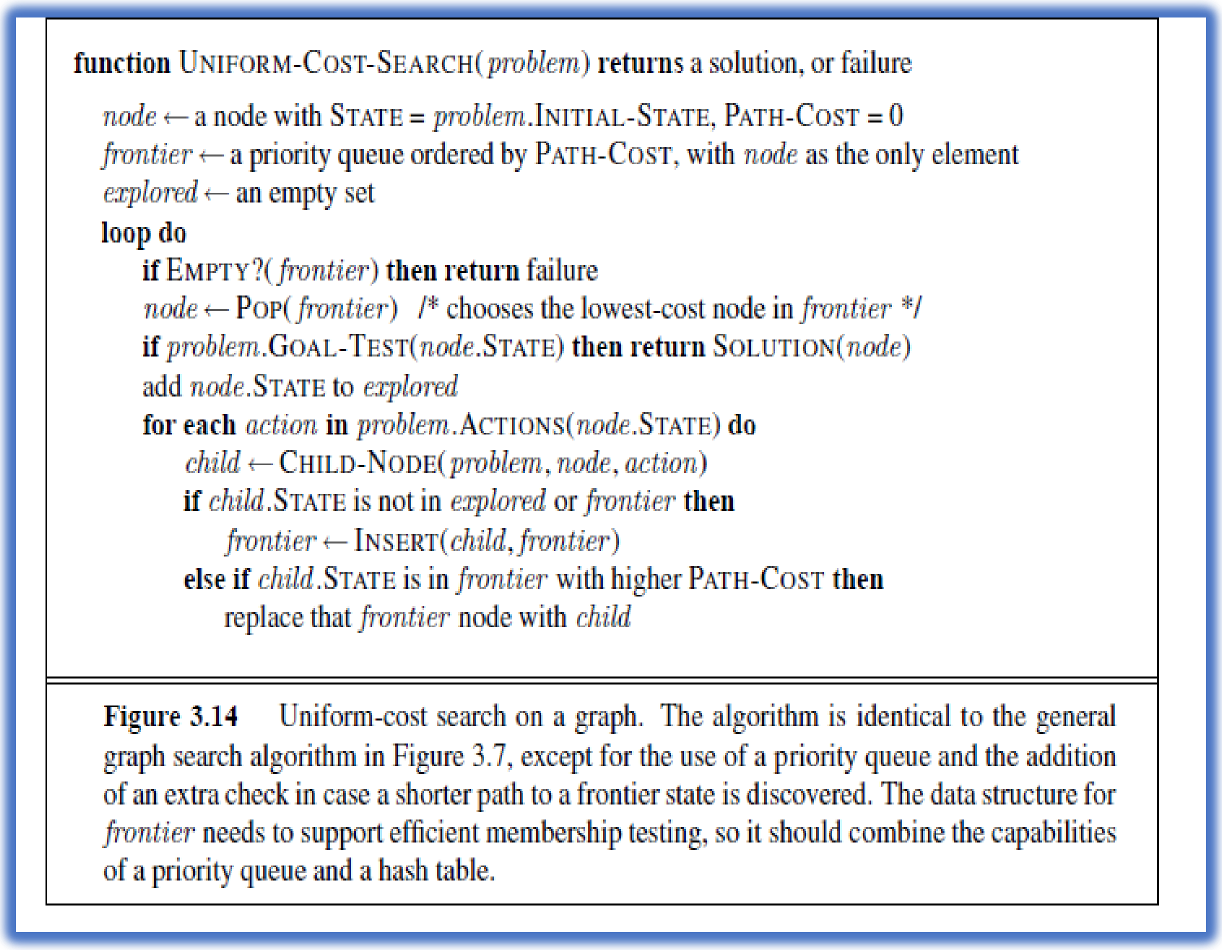
Expert Systems

Application-specific systems that rely on obtaining the knowledge of human experts in an area and programming that knowledge into a system.

Diagnostic Systems : MYCIN system for diagnosing bacterial infections of the blood and suggesting treatments. Intellipath pathology diagnosis system (AMA approved). Pathfinder medical diagnosis system, which suggests tests and makes diagnoses. Whirlpool customer assistance center.

2. b) Write Uniform cost search algorithm and explain its primary purpose in artificial intelligence.

- ➢ **Uniform cost search,** modifies breadth-first strategy by always expanding the lowest-cost node.
- ✓ **Uniform cost search is useful for finding shortest path.**
- ✓ Note that if all step costs are equal, this is identical **to breadth-first search**.
    - ✓ Unsuitable for **operators with negative cost**
- ✓ Uniform-cost search **does not care about the *number* of steps** a path has, **but only about their total cost**.
- ✓ **Instead of expanding the shallowest node, uniform-cost search expands the node n with the *lowest path cost* g(n)**.
- ✓ This is done by storing the frontier as a priority queue ordered by g. The algorithm is shown in Figure 3.14.



```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
      if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              frontier ← INSERT(child, frontier)
          else if child.STATE is in frontier with higher PATH-COST then
              replace that frontier node with child
```

Figure 3.14 Uniform-cost search on a graph. The algorithm is identical to the general graph search algorithm in Figure 3.7, except for the use of a priority queue and the addition of an extra check in case a shorter path to a frontier state is discovered. The data structure for *frontier* needs to support efficient membership testing, so it should combine the capabilities of a priority queue and a hash table.

Instead of expanding the shallowest node, **uniform-cost search expands** the node n with the *lowest path cost* g(n). This is done by storing the frontier as a priority queue ordered by g.

3. a) Explain the interaction between an agent and its environment with examples of real world applications.

- An **agent is anything** that can be viewed as **perceiving** its **environment** through **sensors** and **acting upon** that environment through **actuators.**
- A **human agent** has eyes, ears, and other **organs for sensors** and hands, legs, mouth, and **other body parts for actuators**.
- A **robotic agent** might have **cameras and infrared range finders for sensors** and various **motors for actuators**.
- A **software agent receives keystrokes, file contents, and network packets as sensory inputs** and acts on the environment by displaying on the screen, writing files, and sending network packets.

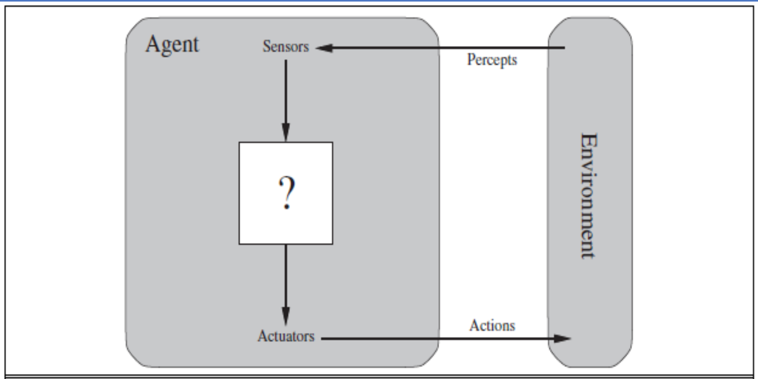


Figure 2.1 Agents interact with environments through sensors and actuators.

- ✓ We use the term **percept** to refer to the agent's perceptual **inputs** at any given instant.
- ✓ An agent's **percept sequence** is the complete history of everything the agent has ever perceived.
- ✓ An **agent's behavior** is described by the **agent function** that maps any given **percept sequence to an action**.

→ *Internally*, the agent function for an artificial agent will be implemented by an **agent program**.

- To illustrate these ideas, we use a very simple example—the **vacuum-cleaner world** shown in Figure 2.2.

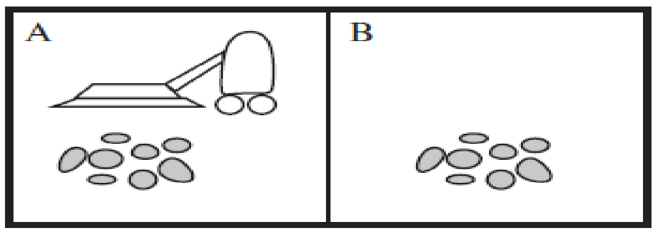
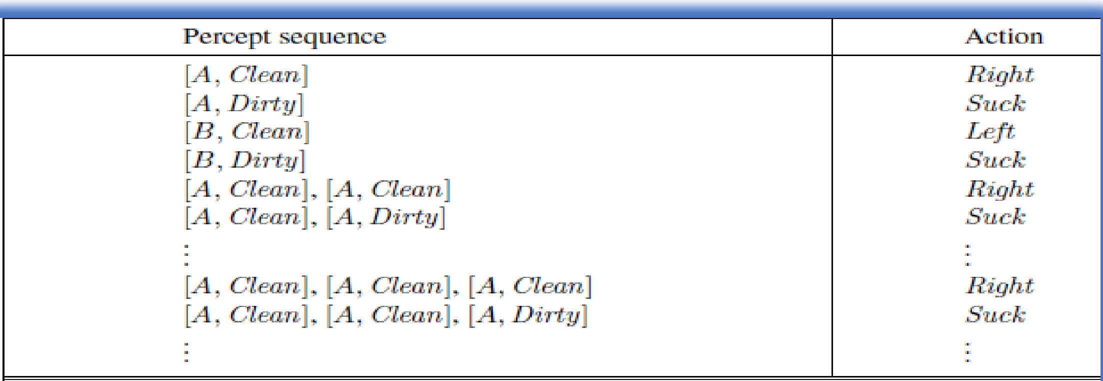


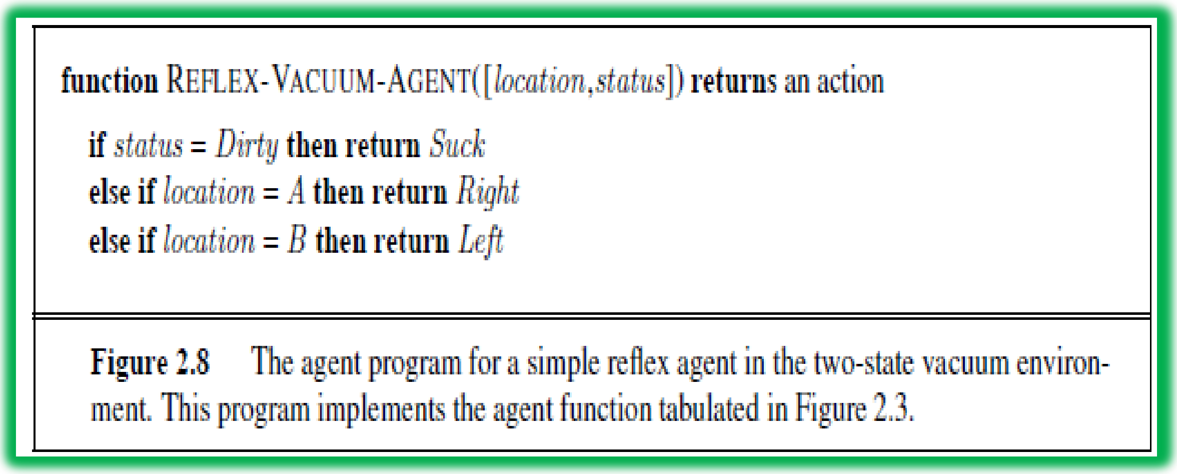
Figure 2.2 A vacuum-cleaner world with just two locations.



Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Two locations: squares A and B.
The vacuum **agent perceives which square it is in and** whether there is dirt in the square.
It can choose to move left, move right, suck up the dirt, or do nothing.
One very simple agent function is the following: if the current square is dirty, then suck;
otherwise, move to the other square.
A partial tabulation of this agent function is shown in **Figure 2.3** and an **agent program that implements it appears in Figure 2.8**

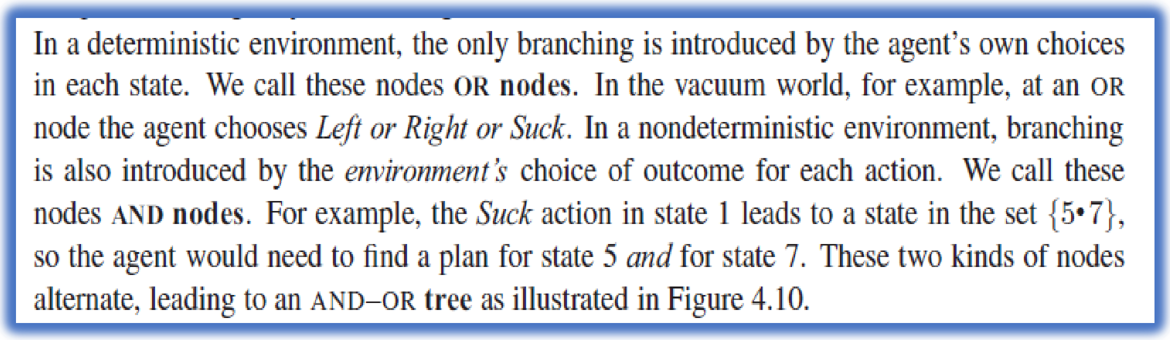


```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

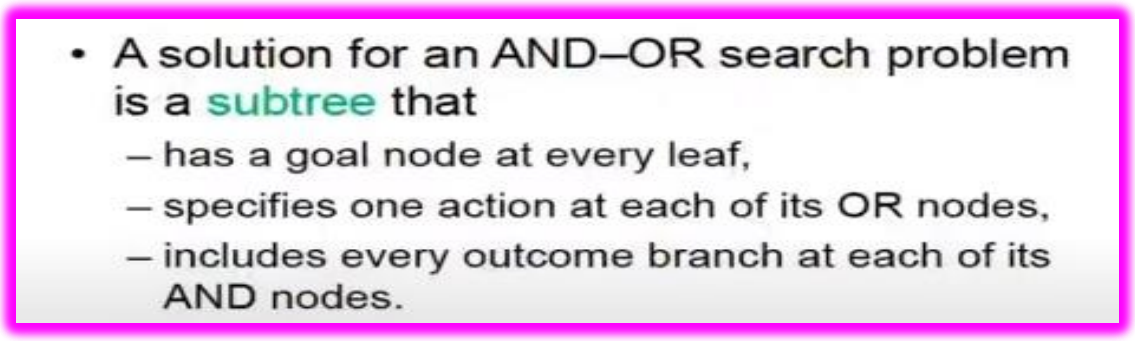
Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

3. b) Illustrate how an AND-OR search tree can be used to solve the problem of an erratic vacuum world.

- ✓ **We may sometimes encounter certain real-life problems which are extremely complicated.**
- ➢ **An effective of solving a complex problem is to reduce it to simpler parts and solve each part separately.**
- ✓ **The problem is automatically solved when we obtain solutions to all smaller, simpler problems.**
- ❖ **Problem reduction→ the structure called AND-OR graph (or tree) is useful for representing the solution of complicated problems.**

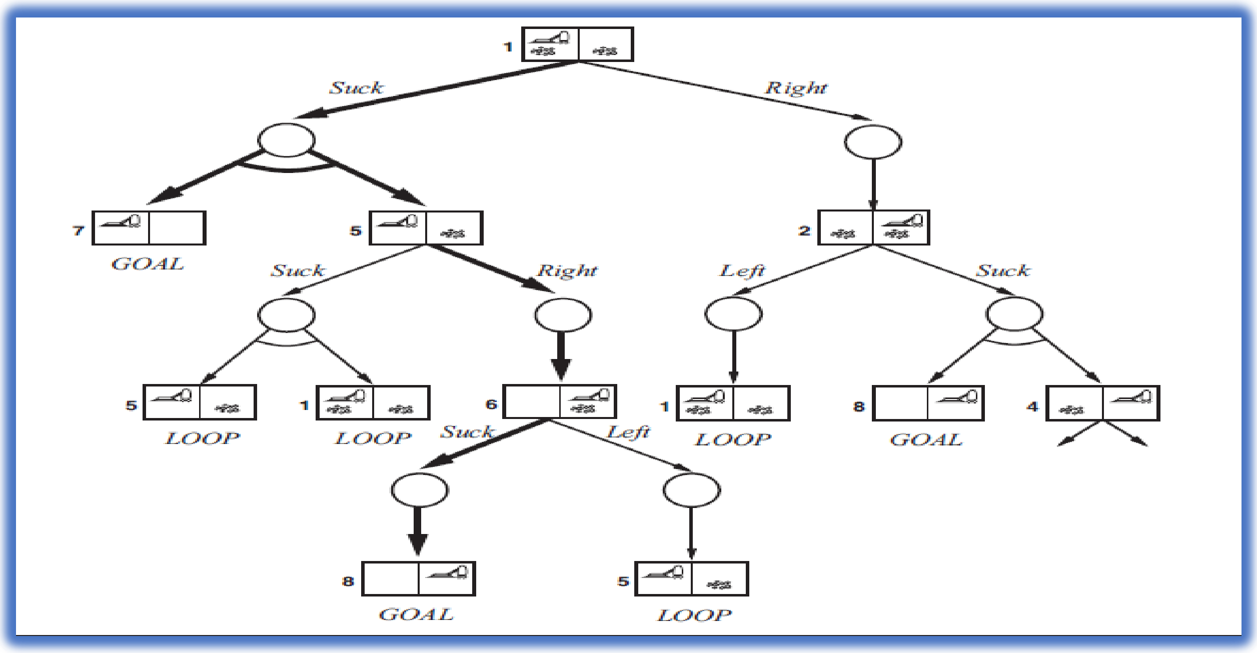


In a deterministic environment, the only branching is introduced by the agent's own choices in each state. We call these nodes OR **nodes**. In the vacuum world, for example, at an OR node the agent chooses *Left or Right or Suck*. In a nondeterministic environment, branching is also introduced by the *environment's* choice of outcome for each action. We call these nodes AND **nodes**. For example, the *Suck* action in state 1 leads to a state in the set {5•7}, so the agent would need to find a plan for state 5 *and* for state 7. These two kinds of nodes alternate, leading to an AND–OR **tree** as illustrated in Figure 4.10.



- A solution for an AND–OR search problem is a subtree that
  – has a goal node at every leaf,
  – specifies one action at each of its OR nodes,
  – includes every outcome branch at each of its AND nodes.

- A solution for an AND–OR search problem is a subtree that
  - has a goal node at every leaf,
  - specifies one action at each of its OR nodes,
  - includes every outcome branch at each of its AND nodes.

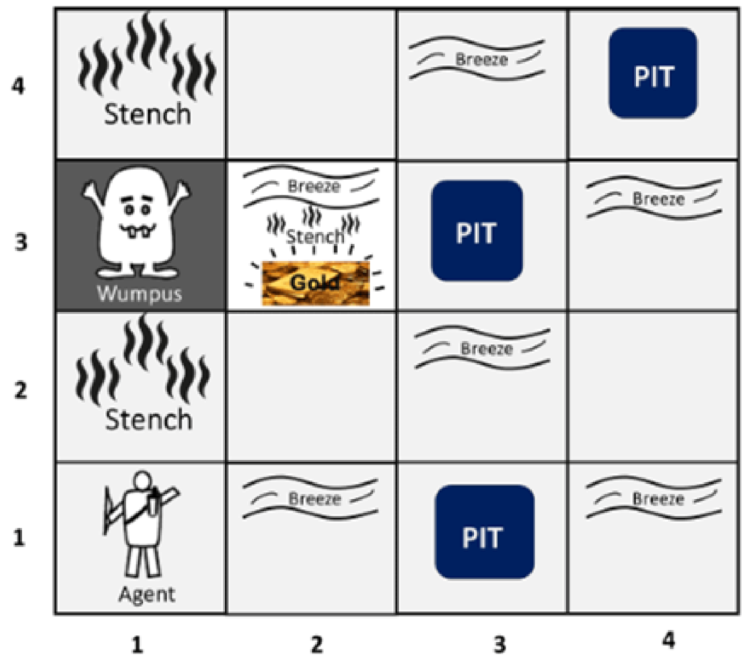


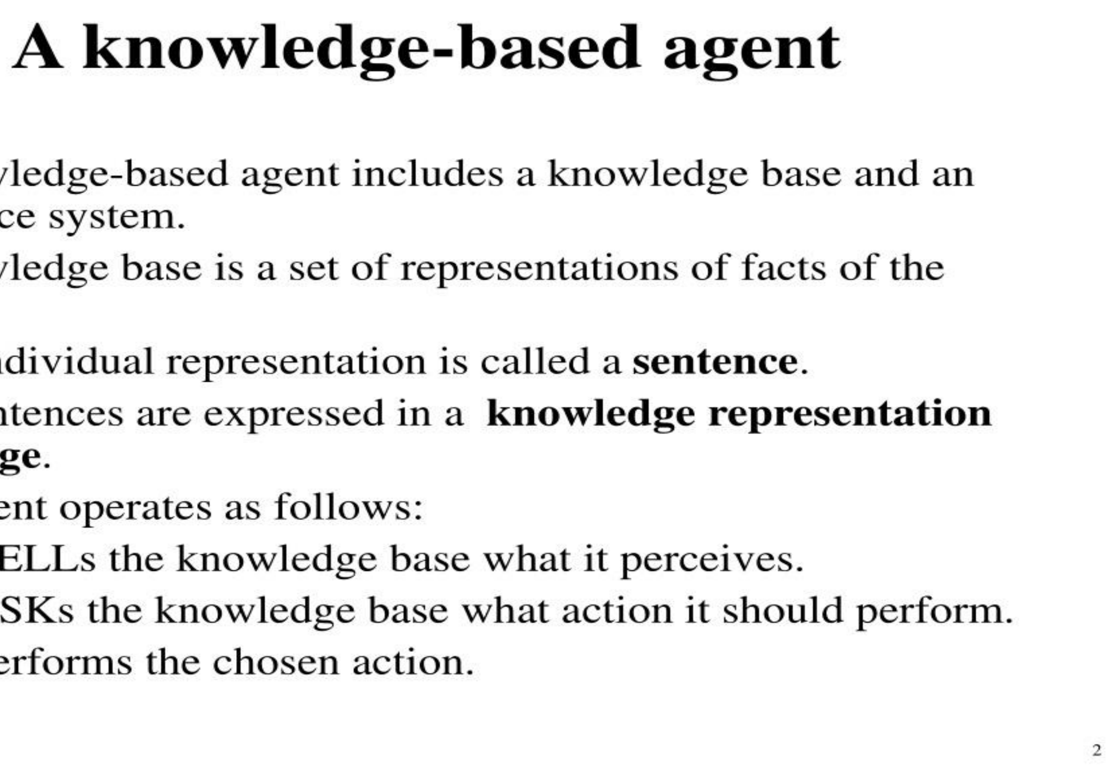
```
function AND-OR-GRAPH-SEARCH(problem) returns a conditional plan• or failure
   OR-SEARCH(problem.INITIAL-STATE, problem, [ ])

function OR-SEARCH(state, problem, path) returns a conditional plan• or failure
   if problem.GOAL-TEST(state) then return the empty plan
   if state is on path then return failure
   for each action in problem.ACTIONS(state) do
       plan ← AND-SEARCH(RESULTS(state, action), problem, [state | path])
       if plan ≠ failure then return [action | plan]
   return failure

function AND-SEARCH(states, problem, path) returns a conditional plan• or failure
   for each $s_i$ in states do
       $plan_i$ ← OR-SEARCH($s_i$, problem, path)
       if $plan_i$ = failure then return failure
   return [if $s_1$ then $plan_1$ else if $s_2$ then $plan_2$ else •••if $s_{n-1}$ then $plan_{n-1}$ else $plan_n$]
```

Figure 4.11 An algorithm for searching AND–OR graphs generated by nondeterministic environments. It returns a conditional plan that reaches a goal state in all circumstances. (The notation [• | •] refers to the list formed by adding object • to the front of list •.)

UNIT – II

4. a) Write a short on Knowledge-Based Agent. Write a PEAS description for Wumpus World given below:





A knowledge-based agent

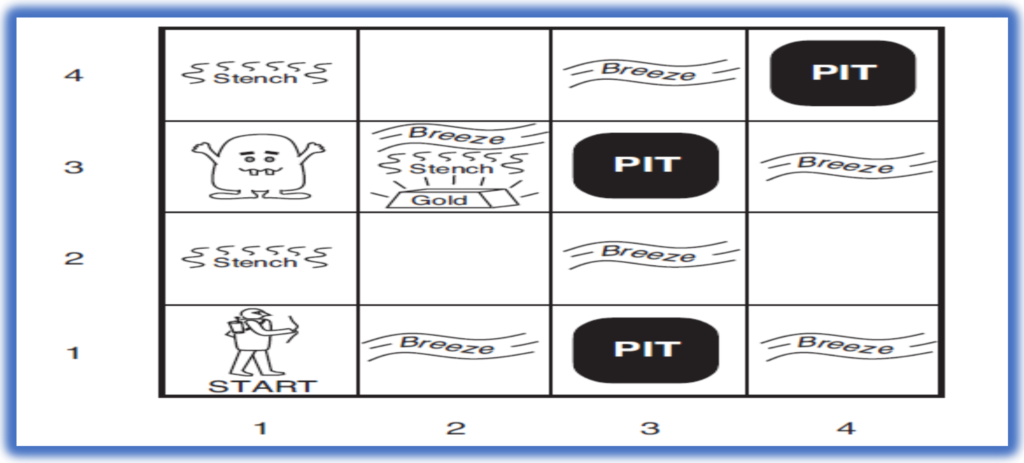
- A knowledge-based agent includes a knowledge base and an inference system.
- A knowledge base is a set of representations of facts of the world.
- Each individual representation is called a **sentence**.
- The sentences are expressed in a **knowledge representation language**.
- The agent operates as follows:
  1. It TELLs the knowledge base what it perceives.
  2. It ASKs the knowledge base what action it should perform.
  3. It performs the chosen action.

2

- **knowledge-based agents** in → **wumpus world** environment.
- The **wumpus world** is a cave consisting of **rooms connected by passageways**.
- Lurking somewhere in the cave is the **terrible Wumpus**, a **beast** that **eats anyone who enters its room.**

The wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in).

Goal → finding a heap of gold



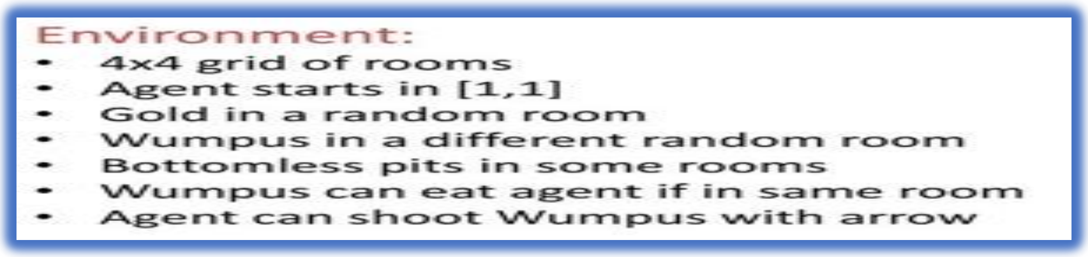
PEAS description for Wumpus world

Performance measure: +1000 for climbing out of the cave with the gold,

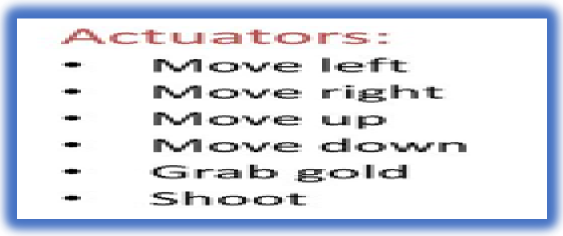
–1000 for falling into a pit or being eaten by the Wumpus

–1 for each action taken and –10 for using up the arrow.

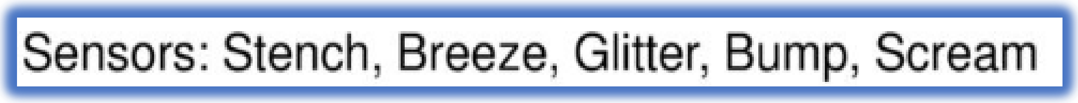
The **game ends either when the agent dies or when the agent climbs out of the cave**.



Environment:
- 4x4 grid of rooms
- Agent starts in [1,1]
- Gold in a random room
- Wumpus in a different random room
- Bottomless pits in some rooms
- Wumpus can eat agent if in same room
- Agent can shoot Wumpus with arrow



Actuators:
- Move left
- Move right
- Move up
- Move down
- Grab gold
- Shoot



Sensors: Stench, Breeze, Glitter, Bump, Scream

In the squares directly **adjacent to a pit**, the agent will **perceive a *Breeze***.

– In the square where the gold is, the agent will perceive a *Glitter*.

– When an agent walks into a wall, it will perceive a *Bump*.

When the wumpus is killed, it emits a woeful *Scream* that can be perceived anywhere in the cave.

4. b) Create about the steps involved in resolution technique with any example

Resolution is a method of theorem proof that involves constructing refutation proofs, or proofs by contradictions. It was created in 1965 by a mathematician named John Alan Robinson.

When several statements are supplied and we need to prove a conclusion from those claims, we employ resolution. In proofs by resolutions, unification is a crucial idea.

Resolution is a single inference rule that can work on either the conjunctive normal form or the clausal form efficiently.

Clause: A clause is a disjunction of literals (an atomic sentence). It's sometimes referred to as a unit clause.

Conjunctive Normal Form (CNF): Conjunctive normal form (CNF) is a sentence that is represented as a conjunction of clauses.

The resolution inference rule:

The propositional rule is just a lifted version of the resolution rule for first-order logic. If two clauses include complementary literals that are expected to be standardized apart so that they share no variables, resolution can resolve them.

$$\frac{l_1 \vee \ldots\ldots \vee l_k, \qquad m_1 \vee \ldots\ldots \vee m_n}{\mathrm{SUBST}(\theta, l_1 \vee \ldots\ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots\ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots\ldots \vee m_n)}$$

Where l_i and mj are complementary literals, there is a resolution in FOL.
Because it only resolves perfectly, this rule is also known as the binary resolution rule .

Example:

We can determine two clauses which are given below:

[Animal (g(x) V Loves (f(x), x)] and [¬ Loves(a, b) V ¬Kills(a, b)]

Two complimentary literals are: Loves (f(x), x) and ¬ Loves (a, b)

These literals could be unified with unifier θ= [a/f(x), and b/x] , and it will bring about a resolvent clause:

[Animal (g(x) V ¬ Kills(f(x), x)].

Steps for Resolution:

1. Conversion of facts into first-order logic
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification)

To better comprehend all of the preceding phases, we shall use resolution as an example.

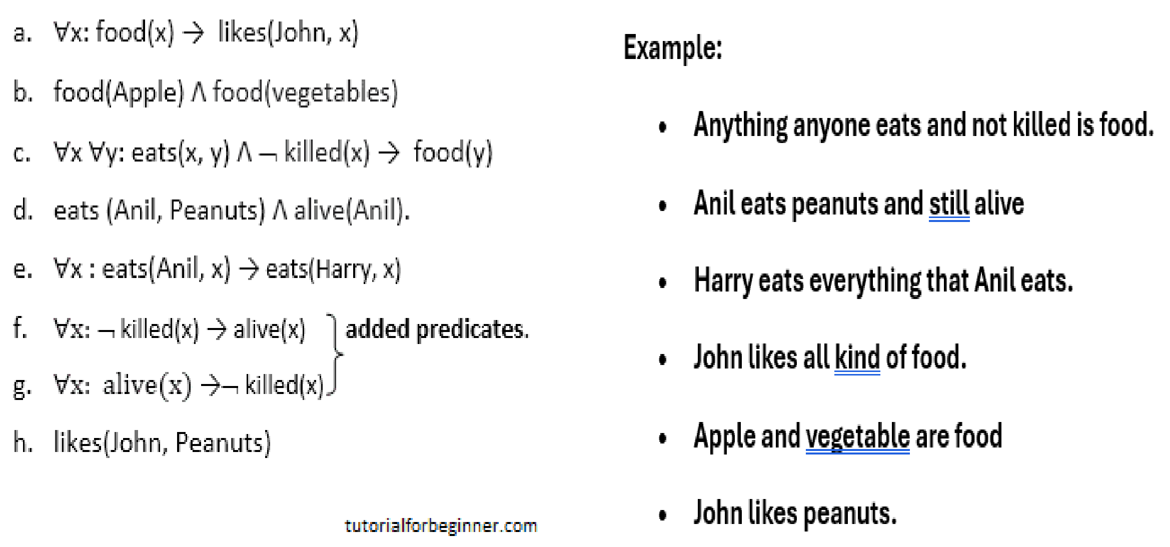
Example:

- Anything anyone eats and not killed is food.
- Anil eats peanuts and still alive
- Harry eats everything that Anil eats.
- John likes all kind of food.
- Apple and vegetable are food
- John likes peanuts.

Prove by resolution that:

Step-1: Conversion of Facts into FOL

We'll start by converting all of the given propositions to first-order logic.



a. ∀x: food(x) → likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y: eats(x, y) Λ ¬ killed(x) → food(y)

d. eats (Anil, Peanuts) Λ alive(Anil).

e. ∀x : eats(Anil, x) → eats(Harry, x)

f. ∀x: ¬ killed(x) → alive(x)

g. ∀x: alive(x) →¬ killed(x)

added predicates.

h. likes(John, Peanuts)

tutorialforbeginner.com

Example:

- **Anything anyone eats and not killed is food.**
- **Anil eats peanuts and still alive**
- **Harry eats everything that Anil eats.**
- **John likes all kind of food.**
- **Apple and vegetable are food**
- **John likes peanuts.**

Step-2: Conversion of FOL into CNF

Converting FOL to CNF is essential in first-order logic resolution because CNF makes resolution proofs easier.

✓ Eliminate all implication (→) and rewrite:

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x¬ [¬ killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

✓ Move negation (¬)inwards and rewrite

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x ¬killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

- ✓ Rename variables or standardize variables

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x ¬killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

- ✓ Eliminate existential instantiation quantifier by elimination.
  - We will eliminate existential quantifiers in this step, which is referred to as Skolemization. However, because there is no existential quantifier in this example problem, all of the assertions in this phase will be the same

- ✓ Drop Universal quantifiers.

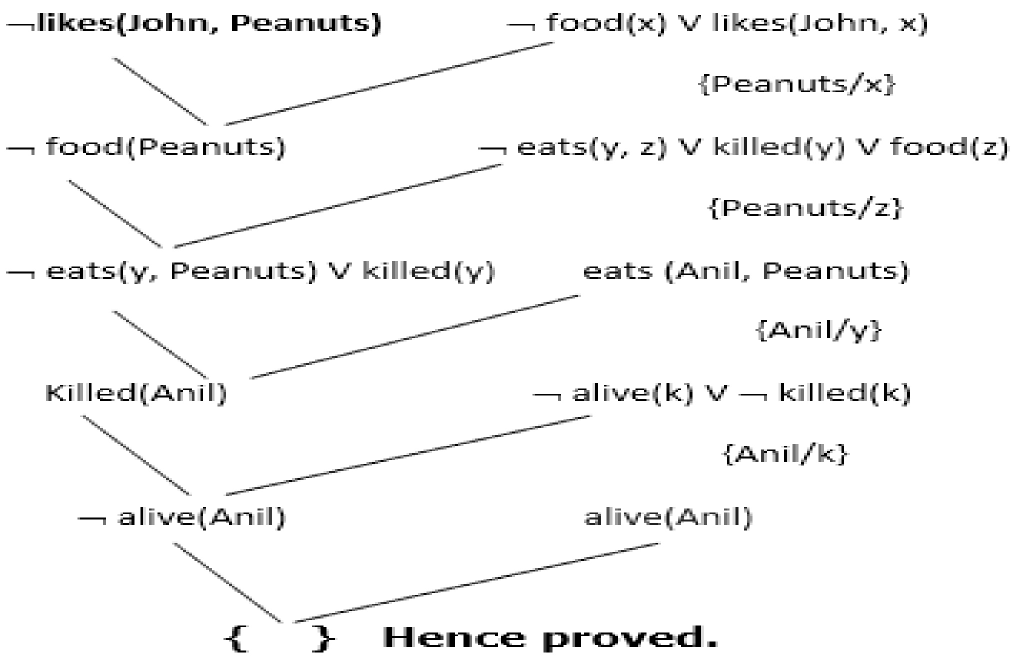
  - We'll remove all universal quantifiers ∀ in this phase because none of the statements are implicitly quantified, therefore we don't need them

1. ¬ food(x) V likes(John, x)
2. food(Apple)
3. food(vegetables)
4. ¬ eats(y, z) V killed(y) V food(z)
5. eats (Anil, Peanuts)
6. alive(Anil)
7. ¬ eats(Anil, w) V eats(Harry, w)
8. killed(g) V alive(g)
9. ¬ alive(k) V ¬ killed(k)
10. likes(John, Peanuts).
    [Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two independent statements.]

- ✓ Distribute conjunction ∧ over disjunction V`. This step will not make any change in this problem.

Step 3: Reverse the statement that needs to be proven.

We will use negation to write the conclusion assertions in this statement, which will be written as "likes" (John, Peanuts)



As a result, the conclusion's negation has been demonstrated to constitute a total contradiction with the given collection of truths.

Step 4: Create a graph of resolution:

In this stage, we'll use a resolution tree and substitution to solve the problem. It will be as follows for the afore said problem:

Explanation of Resolution graph:

- **Step 1:** ¬likes(John, Peanuts) , and likes(John, x) get resolved(canceled) by substitution of {Peanuts/x}, and we are left with ¬ food(Peanuts)
- **Step 2:** ¬ food(Peanuts) , and food(z) get resolved (canceled) by substitution of { Peanuts/z}, and we are left with¬ eats(y, Peanuts) V killed(y) .
- **Step 3:** ¬ eats(y, Peanuts) and eats (Anil, Peanuts) get resolved by substitution {Anil/y}, and we are left with Killed(Anil).
- **Step 4:** Killed(Anil) and ¬ killed(k) get resolve by substitution {Anil/k}, and we are left with ¬ alive(Anil) .
- **Step 5:** ¬ alive(Anil) and alive(Anil) get resolve.

5. a) Construct a FOL representation for the following sentences

a. Horses, cows, and pigs are mammals. **b. An offspring of a horse is a horse.**
c. Bluebeard is a horse. **d. Bluebeard is Charlie's parent.**
e. Offspring and parent are inverse relations. **f. Every mammal has a parent.**

a) Horses, cows, and pigs are mammals.

(If x is a horse, cow, or pig, then it is a mammal)
(If x is a horse, then it is a mammal)
(If x is a cow, then it is a mammal)
(If x is a pig, then it is a mammal)

Symbolic representation:

Horse(x) ⇒ Mammal(x), Cow(x) ⇒ Mammal(x), Pig(x) ⇒ Mammal(x)

b) An offspring of a horse is a horse.

(If x is a horse and x has an offspring y, then y is a horse)

Symbolic representation:

Horse(x) ^ Offspring(x,y) ⇒ Horse(y)

c) Bluebeard is a horse.

(If x is a horse and x's name is Bluebeard, then Bluebeard is a horse)

Symbolic representation:

Horse(x) ^ Name(x,Bluebeard) ⇒ Horse(Bluebeard)

d) Bluebeard is Charlie's parent.

Symbolic representation:

ParentOf(Bluebeard,Charlie)

e) Offspring and parent are inverse relations.

(x is the parent of y if and only if y is the offspring of x)

Symbolic representation:

ParentOf(x,y) ⇔ Offspring(y,x)

f) Every mammal has a parent.

Symbolic representation:

Mammal(x) ⇒HasParent(x)

5. b) List the basic steps involved in the unification algorithm with example.

- Unification is the process of finding a substitute that makes two separate logical atomic expressions identical. The substitution process is necessary for unification.
- It accepts two literals as input and uses substitution to make them identical.
- Let Ψ_1 and Ψ_2 be two atomic sentences, and be a unifier such that $\boldsymbol{\Psi_{1\sigma} = \Psi_{2\sigma}}$, then **UNIFY($\Psi_1$, Ψ_2)**can be written.
- **Example: Find the MGU for Unify{King(x), King(John)}**
  Let Ψ_1 = King(x), Ψ_2 = King(John),

Substitution θ = {John/x} is a unifier for these atoms, and both equations will be equivalent if this substitution is used.

- For unification, the UNIFY algorithm is employed, which takes two atomic statements and returns a unifier for each of them (If any exist).
- All first-order inference techniques rely heavily on unification.
- If the expressions do not match, the result is failure.
- The replacement variables are referred to as MGU (Most General Unifier).

Conditions for Unification

The following are some fundamental requirements for unification:

- Atoms or expressions with various predicate symbols can never be united.
- Both phrases must have the same number of arguments.
- If two comparable variables appear in the same expression, unification will fail.

Unification Algorithm: : Unify(Ψ_1, Ψ_2)

Step. 1: If Ψ_1 **or** Ψ_2 **is** a variable **or** constant, **then**:

- a) If Ψ_1 **or** Ψ_2 are identical, **then return** NIL.
- b) Else **if** Ψ_1 **is** a variable,
  - a. **then if** Ψ_1 occurs **in** Ψ_2, **then return** FAILURE
  - b. Else **return** { (Ψ_2/ Ψ_1)}.
- c) Else **if** Ψ_2 **is** a variable,
  - a. If Ψ_2 occurs **in** Ψ_1 **then return** FAILURE,
  - b. Else **return** {(Ψ_1 / Ψ_2)}.
- d) Else **return** FAILURE.

Step.2: If the initial Predicate symbol **in** Ψ_1 **and** Ψ_2 are **not** same, **then return** FAILURE.

Step. 3: IF Ψ_1 **and** Ψ_2 have a different number **of** arguments, **then return** FAILURE.

Step. 4: Set Substitution **set**(SUBST) to NIL.

Step. 5: For i=1 to the number **of** elements **in** Ψ_1.

- a) Call Unify **function with** the ith element **of** Ψ_1 **and** ith element **of** Ψ2, **and** put the result **into** S.
- b) If S = failure **then** returns Failure
- c) If S $\neq$ NIL **then do**,

```
        a. Apply S to the remainder **of** both L1 **and** L2.
        b. SUBST= APPEND(S, SUBST).
    Step.6: Return SUBST.
```

Example:

Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}

S_0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}
SUBST θ={b/Z}

S_1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}
SUBST θ={f(Y) /X}

S_2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}
SUBST θ= {g(b) /Y}

S_2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} **Unified Successfully**.
And Unifier = { b/Z, f(Y) /X , g(b) /Y}.

6. **a) List and explain various primitive actions and represent the following sentence in conceptual dependency**

   "*John purchased the house from mike*"

It is an another knowledge representation technique in which we can represent any kind of knowledge.

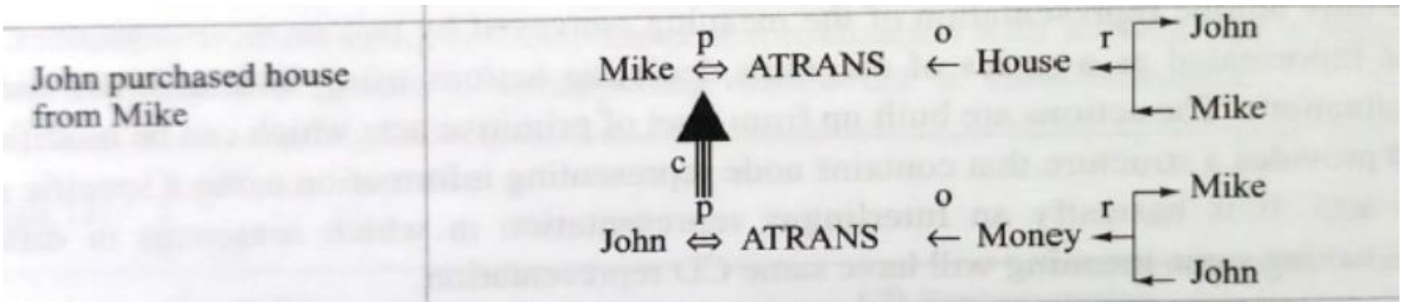
Conceptual Dependency (CD) is a theory of how to represent the kind of knowledge about events that is usually contained in natural language sentences.

- ✓ it is developed by Roger Schank at Stanford University in 1969.
- ✓ It mainly focuses on the concept and meaning rather than on syntax or structure.
- ✓ CD representations use elven primitive acts as the basis for most activities.

Conceptual Primitives Actions

- ✓ **ATRANS** Transfer of an abstract relationship (e.g., give)
- ✓ **PTRANS** Transfer of the physical location of an object (e.g., go or walk)
- ✓ **PROPEL** Application of physical force to an object (e.g., push, pull)
- ✓ **MOVE** Movement of a body part by its owner (e.g., kick, throw, hit)
- ✓ **GRASP** Grasping of an object by an actor (e.g., catch, clutch)
- ✓ **INGEST** Ingestion of an object by an animal (e.g., eat, drink)
- ✓ **EXPEL** Expulsion of something from the body of an animal (e.g., cry, sweat)
- ✓ **MTRANS** Transfer of mental information (e.g., tell, read, speak, sing)
- ✓ **MBUILD** Building new information out of old (e.g., decide, describe, answer)
- ✓ **SPEAK** Production of sounds (e.g., say, tell and sing)
- ✓ **ATTEND** Focusing of a sense organ toward a. stimulus (e.g., listen, watch, see and look)

John purchased house from Mike

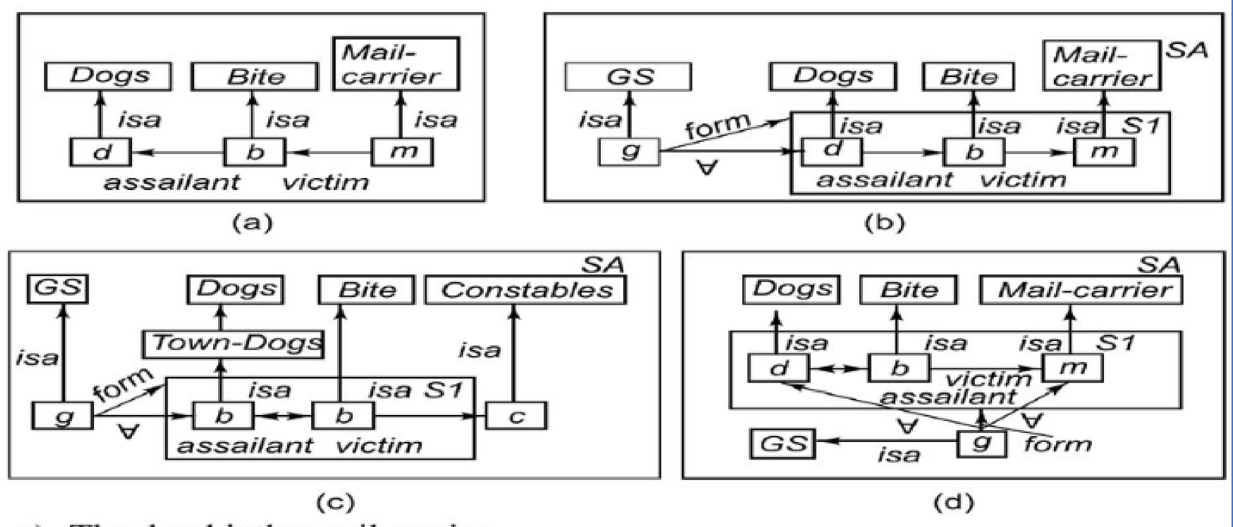


6. b) Explain the importance of semantic net structure in AI and represent semantic net for the following sentence.

“Every dog has bitten every mail carrier”

Representing knowledge in semantic networks

♣ A semantic network represents knowledge in the form of a graph in which the nodes represent objects, situations, or events, and the arcs represent the relationships between them.
♣ originally developed by M. R. Quillian as a model for human memory labeled, directed graph o nodes represent objects, concepts, or situations o labels indicate the name nodes can be instances (individual objects) or classes (generic nodes)
♣ links represent relationships o the relationships contain the structural information of the knowledge to be represented o the label indicates the type of the relationship Nodes and Arcs Arcs define binary relationships that hold between objects denoted by the nodes.



a) The dog bit the mail carrier.
b) Every dog has bitten a mail carrier.
c) Every dog in town has bitten the constable.
d) Every dog has bitten every mail carrier.

Note: Consider d) Figure

7. . **a) Represent the following sentences using conceptual dependency representation.**

John ate an apple.
Mary gave a book to john.
Mary hit the ball with a bat.
The teacher explained the concept to the students.

Conceptual Dependency (CD) representation for the given sentences:

1. **John ate an apple:**

- **Actor:** John (PP)
- **Action:** INGEST (ACT)
- **Object:** apple (O)

[Image: "John" -> "INGEST" -> "apple"]

2. **Mary gave a book to John:**

- **Actor:** Mary (PP)
- **Action:** ATRANS (ACT) - Transfer of possession
- **Object:** book (O)
- **Recipient:** John (R)

[Image: "Mary" -> "ATRANs" -> "book" -> "John" (R)]

3. Mary hit the ball with a bat:

- **Actor:** Mary (PP)
- **Action:** PROPEL (ACT) - Physical force applied
- **Object:** ball (O)
- **Instrument:** bat (I)
-

[Image: "Mary" -> "PROPEL" -> "ball" (O) -> "bat" (I)]

4. The teacher explained the concept to the students:

- **Actor:** Teacher (PP)
- **Action:** MTRANS (ACT) - Transfer of mental information
- **Object:** concept (O)
- **Recipient:** students (R)
-

[Image: "Teacher" -> "MTRANS" -> "concept" (O) -> "students" (R)]

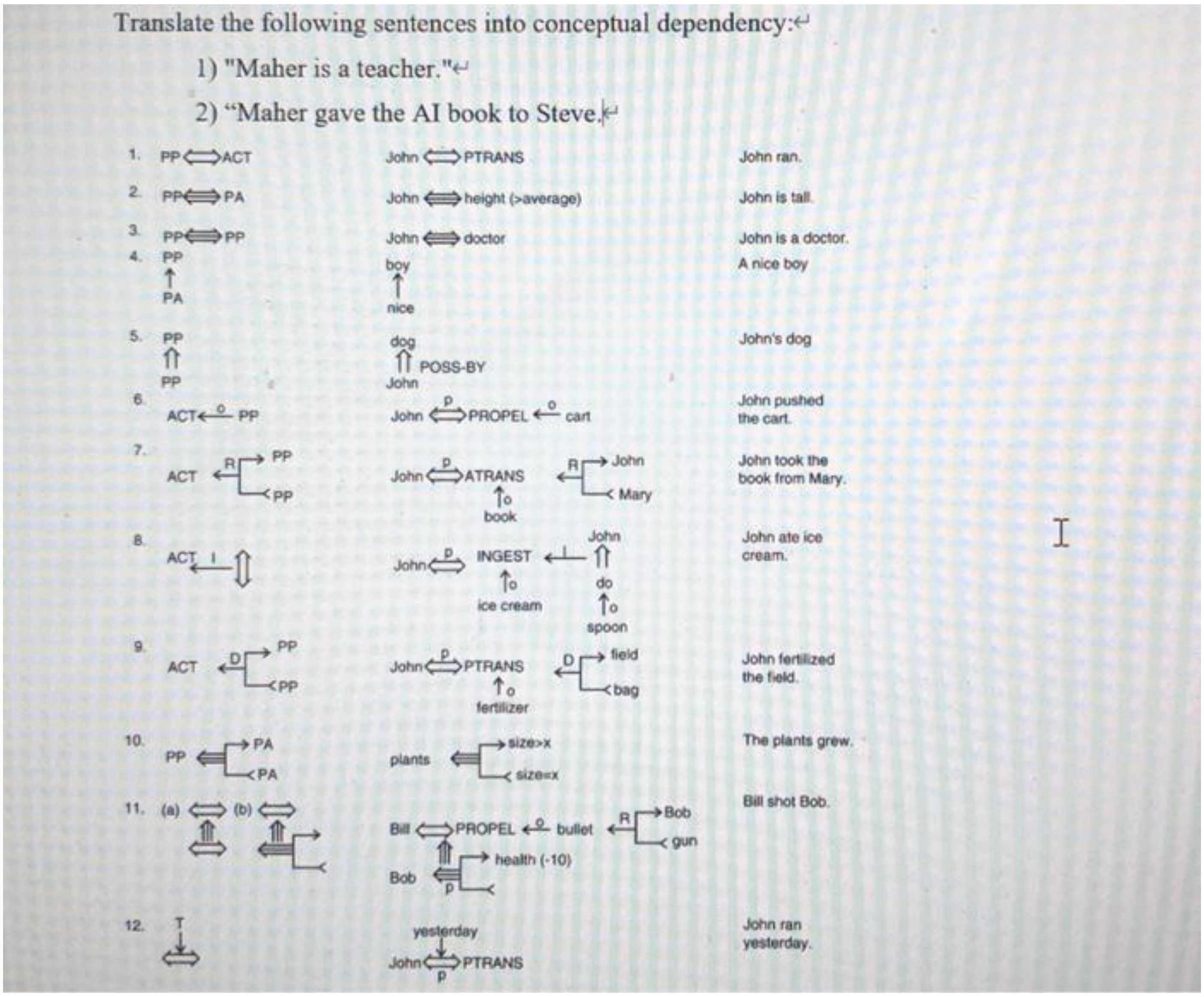
Key points about CD representation:

- **Primitive acts:** The core actions like "INGEST", "ATRANs", "PROPEL", "MTRANS" are called primitive acts.

- **Arrows:** Arrows indicate the direction of the action.

- **Labels:** "PP" for the performer (actor), "O" for the object, "R" for the recipient, "I" for the instrument.

Sample:



Translate the following sentences into conceptual dependency:

1) "Maher is a teacher."

2) "Maher gave the AI book to Steve.

1.	PP ⇔ ACT	John ⇔ PTRANS	John ran.
2.	PP ⇔ PA	John ⇔ height (>average)	John is tall.
3.	PP ⇔ PP	John ⇔ doctor	John is a doctor.
4.	PP ↑ PA	boy ↑ nice	A nice boy
5.	PP ⇑ PP	dog ⇑ POSS-BY John	John's dog
6.	ACT ←o PP	John ⇔p PROPEL ←o cart	John pushed the cart.
7.	ACT ←R (→ PP, ← PP)	John ⇔p ATRANS ↑o book ←R (→ John, ← Mary)	John took the book from Mary.
8.	ACT ←I ⇕	John ⇔p INGEST ↑o ice cream ←I John ⇑ do ↑o spoon	John ate ice cream.
9.	ACT ←D (→ PP, ← PP)	John ⇔p PTRANS ↑o fertilizer ←D (→ field, ← bag)	John fertilized the field.
10.	PP ⇐ (→ PA, ← PA)	plants ⇐ (→ size>x, ← size=x)	The plants grew.
11.	(a) ⇔ ⇑ ⇔ (b) ⇔ ⇑ ⇐	Bill ⇔ PROPEL ←o bullet ←R (→ Bob, ← gun) ⇑ Bob ⇐p (→ health (-10))	Bill shot Bob.
12.	T ↓ ⇔	yesterday ↓ John ⇔p PTRANS	John ran yesterday.

7. b) Discuss the components of a planning system in artificial intelligence.

Simple problem solving tasks basically involve the following tasks:

- Choose the **best rule** based upon heuristics.
- **Apply this rule** to create a new state.
- Detect when a **solution** is found.
- Detect **dead ends** so that they can be avoided.

Find out when a **near-perfect solution** is found.

1. <u>Choice of best rule</u>:

- finding the **differences between the current states and the goal states** and then **Choosing the rules** that reduce these differences most effectively.
- Means end analysis good example of this.

If we wish to **travel by car to visit a friend**

- The first thing to do is to **fill up the car with fuel.**
- If we do not have a car then **we need to acquire one.**
- The **largest difference must be tackled first**.

2. Rule application:

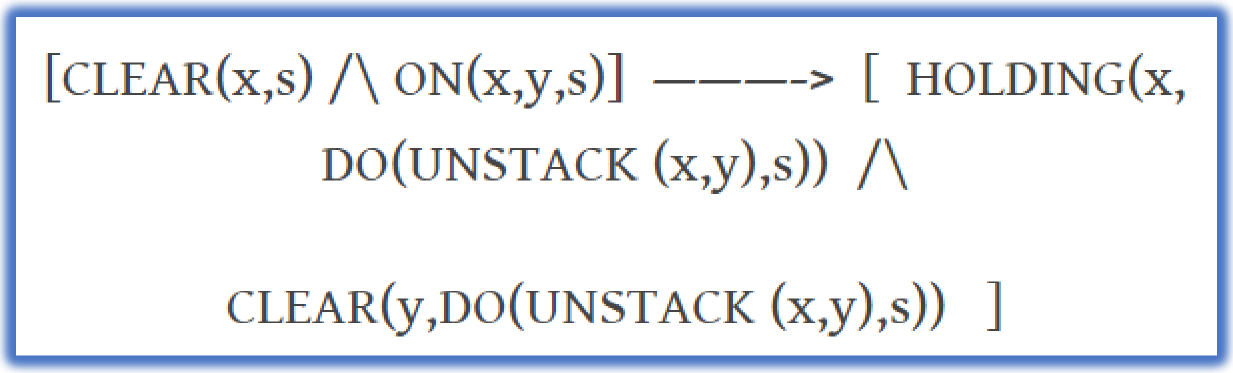
- A number of approaches to this task have been used.

Green's Approach (1969)

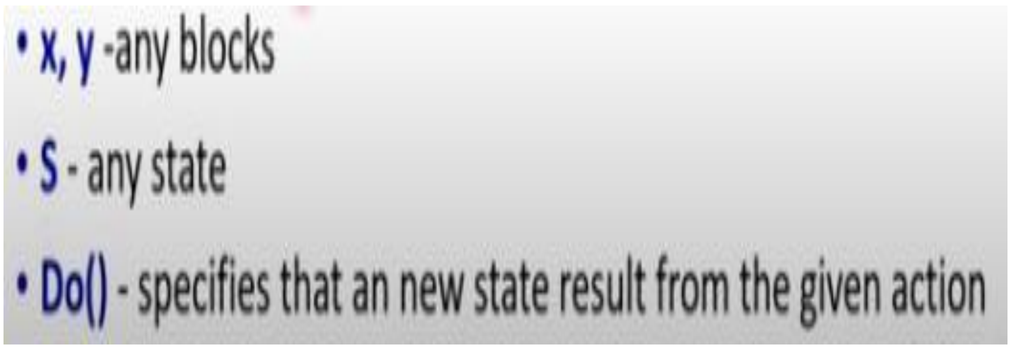
Basically this states that *we note the changes to a state produced by the application of a rule*.

- Consider the problem of having **two blocks *A* and *B* stacked on each other** (*A* on top).
- Then we may have an **initial state** S_0
- which could be described as:
- **ON(A, B, S_0) ^ ONTABLE(B, S_0) ^ CLEAR(A, S_0)**

If we now wish to **UNSTACK(*A*, *B*).** We express the operation as follows:



$$[\text{CLEAR}(x,s) \wedge \text{ON}(x,y,s)] \longrightarrow [\ \text{HOLDING}(x, \text{DO}(\text{UNSTACK}(x,y),s)) \wedge \text{CLEAR}(y,\text{DO}(\text{UNSTACK}(x,y),s))\ ]$$



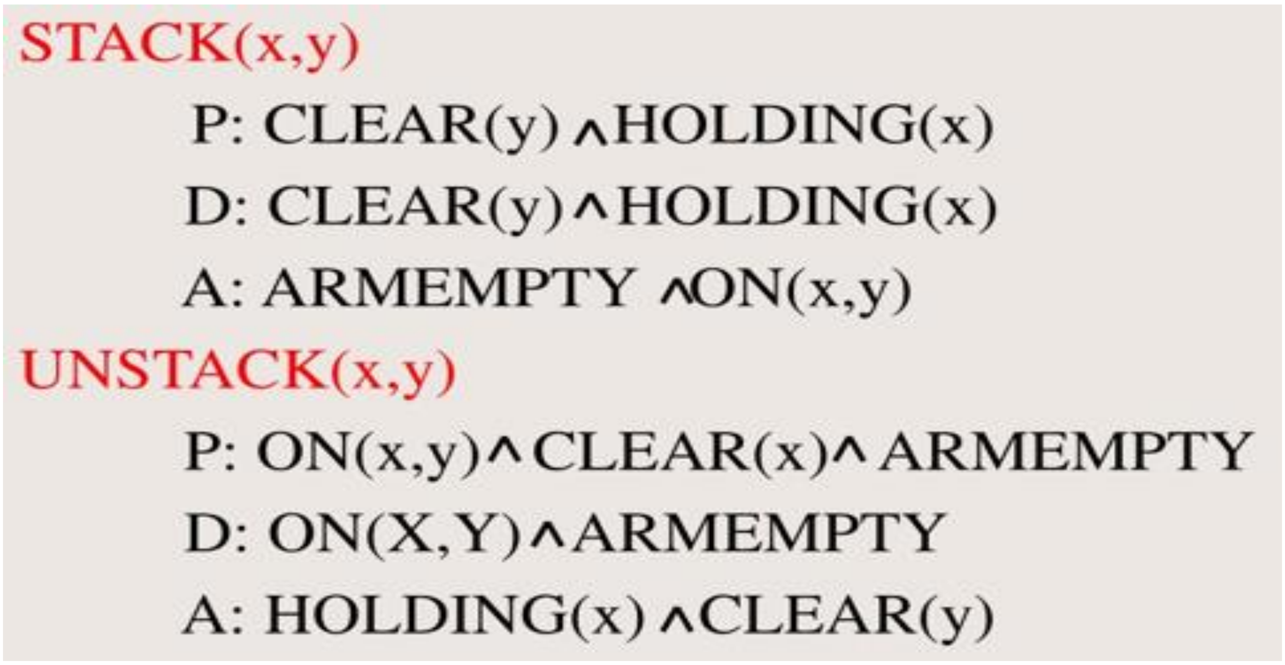
- x, y -any blocks
- S - any state
- Do() - specifies that an new state result from the given action

another approach:

STRIPS (1971.) (Stanford Research institute planning system)

→ **A mechanism that does not require a large number of explicit frame axiom.**

- Basically each operator has **three lists of predicates** associated with it:
  - A list of things that become TRUE called **ADD.**
  - A list of things that become FALSE called **DELETE**.
  - A set of **prerequisites** that must be true before the operator can be applied.
- Anything not in these lists is assumed to be unaffected by the operation.
- This method initial implementation of STRIPS -- has been *extended* to include other forms of reasoning/planning


STACK(x,y)

P: CLEAR(y) ∧HOLDING(x)

D: CLEAR(y)∧HOLDING(x)

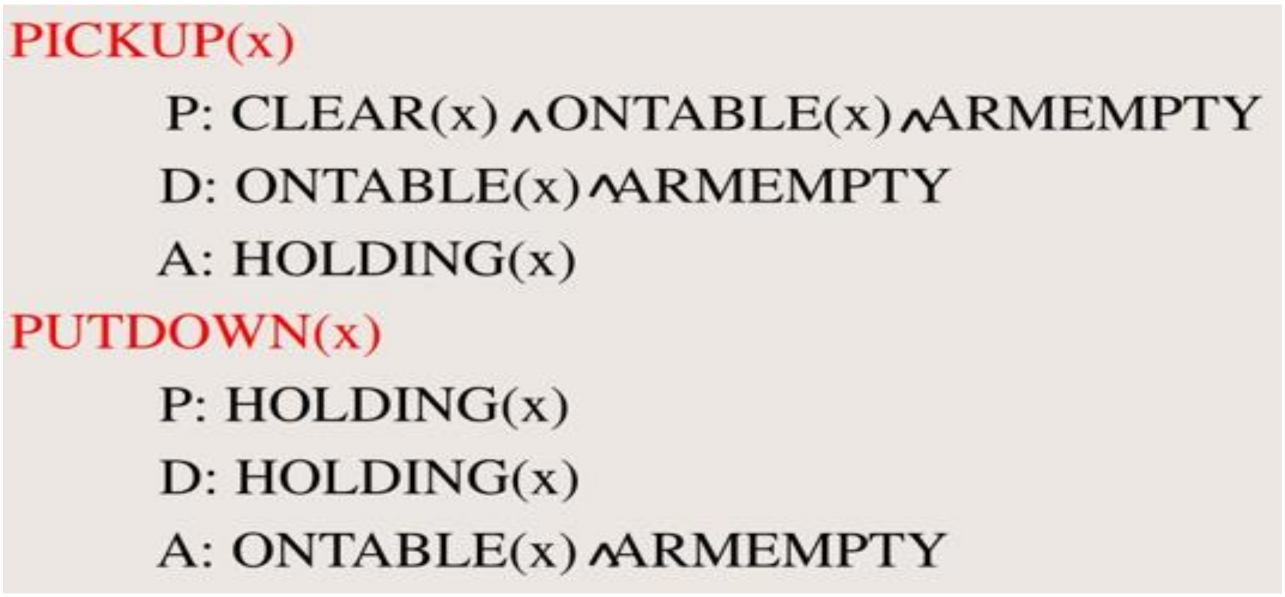
A: ARMEMPTY ∧ON(x,y)

UNSTACK(x,y)

P: ON(x,y)∧CLEAR(x)∧ ARMEMPTY

D: ON(X,Y)∧ARMEMPTY

A: HOLDING(x) ∧CLEAR(y)


PICKUP(x)

P: CLEAR(x) ∧ONTABLE(x) ∧ARMEMPTY

D: ONTABLE(x)∧ARMEMPTY

A: HOLDING(x)

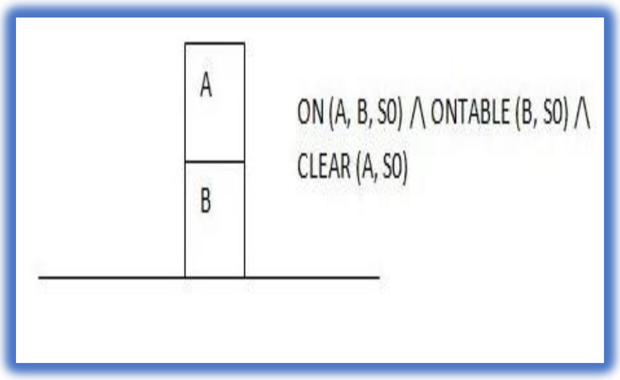
PUTDOWN(x)

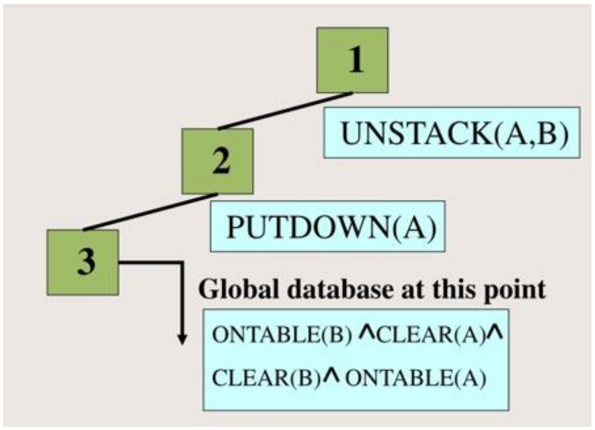
P: HOLDING(x)

D: HOLDING(x)

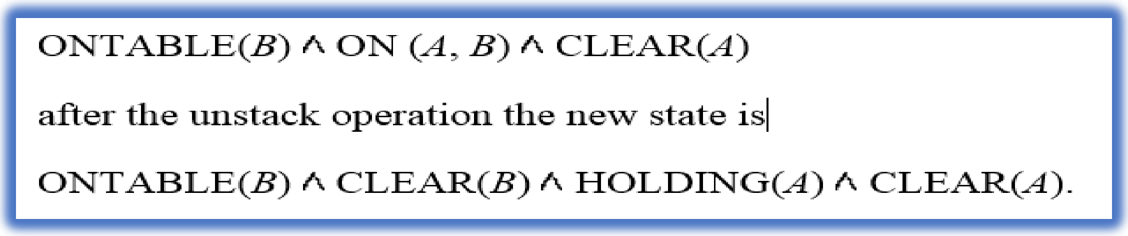
A: ONTABLE(x) ∧ARMEMPTY

Figure → STRIPS-style operators for the blocks world





For example if we start with the situation shown in **figure → 8,** we would describe it as



ONTABLE(B) ∧ ON (A, B) ∧ CLEAR(A)

after the unstack operation the new state is

ONTABLE(B) ∧ CLEAR(B) ∧ HOLDING(A) ∧ CLEAR(A).

3. Detecting a solution

- A planning system has succeeded in finding a solution to a **problem** when it has found a sequence of operators that transform the **initial state into the goal state**.

4. Detecting dead Ends

As a planning system is searching for a sequence of operators to solve a particular problem, it must be able to **detect when it is exploring a path that can never lead to a solution.**
The **same reasoning mechanisms** that can be used to detect a solution can often be used for detecting a dead end.
If the search process is reasoning forward from the initial state, it can **prune any path** that leads to a state from **which the goal state cannot be reached**.

8. a) Define learning in the context of artificial intelligence and explain its significance with winston's learning program.

Learning is an important area in AI, perhaps more so than planning.

- Problems are hard -- harder than planning.
- Recognized Solutions are not as common as planning.
- A goal of AI is to enable computers that can be taught rather than programmed.

***Learning* is an area of AI that focusses on processes of self-improvement.**
Information processes that improve their performance or enlarge their knowledge bases are **said to learn**.

- Winston (1975) described a Blocks World Learning program. This program operated in a simple blocks domain.
- The goal is to construct **representation of the definition of concepts** in the blocks domain.

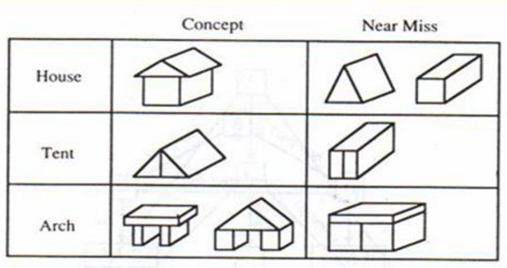


Figure 17.2: Some Blocks World Concepts

For example : Concepts such as House

It learned the concepts **House, Tent, and Arch** shown in figure → 17.2.
The figure also shows an example of a near miss for each concept. A **near miss** is an object that is **not an instance** of the concept in question but that is **very similar to such instances.**

<u>The basic approach of Winston's program</u>

There are three basic steps to the problem of concept formulation:

1. Select **one know instance** of the concept. Call this the ***concept definition.***

2. Examine definitions of **other known instance** of the concept. ***Generalise* the definition** to include them.

3. Examine **descriptions of *near misses***. *Restrict* the definition to ***exclude* these**.
   Both steps 2 and 3 rely on comparison and both similarities and differences need to be identified.

The c-note link from node C describes the difference found by the comparison routine.
This description says simply that node **C must be either a Brick or a Wedge.**

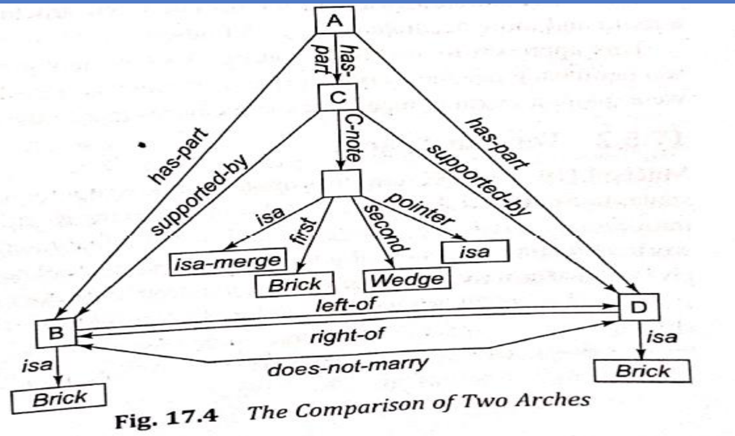


Fig. 17.4 *The Comparison of Two Arches*

The c-note link from node C describes the difference found by the comparison routine.
This description says simply that node **C must be either a Brick or a Wedge.**

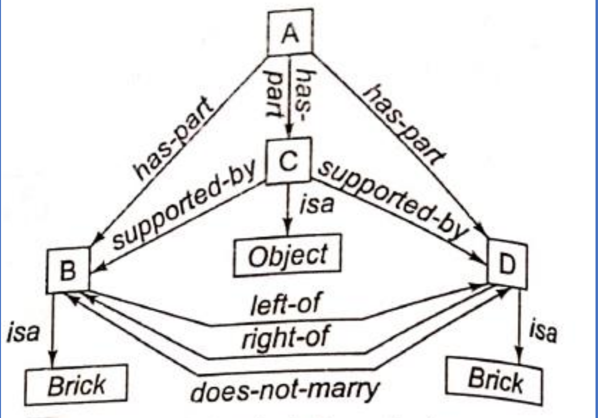


Fig. 17.5 *The Arch Description after Two Examples*

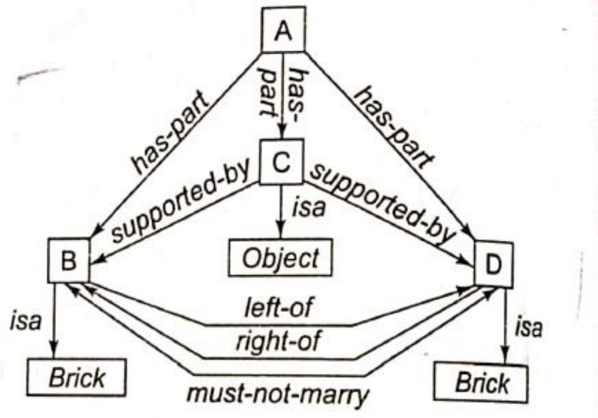


Fig. 17.6 *The Arch Description after a Near Miss*

8. b) Explain in brief about explanation based learning (EBL) with example.

Explanation-Based Learning (EBL)In terms of Machine Learning, it is an algorithm that aims to understand why an example is part of a particular concept to make generalizations or form concepts from training examples. For example, EBL uses a domain theory and creates a program that learns to play chess.

An EBL accepts 4 kinds of input:

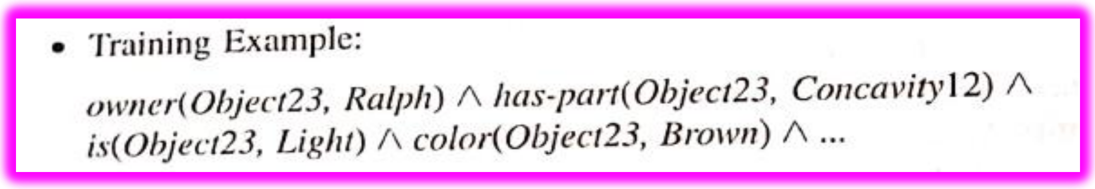
A training example-- what the learning *sees* in the world.
A goal concept-- a high level description of what the program is supposed to learn.
A operational criterion-- a description of which concepts are usable.
A domain theory-- a set of rules that describe relationships between objects and actions in a domain

From **input** EBL computes a generalization of the training example that is sufficient not only to describe the goal concept but also satisfies **the operational criterion**.

EBL has two steps:

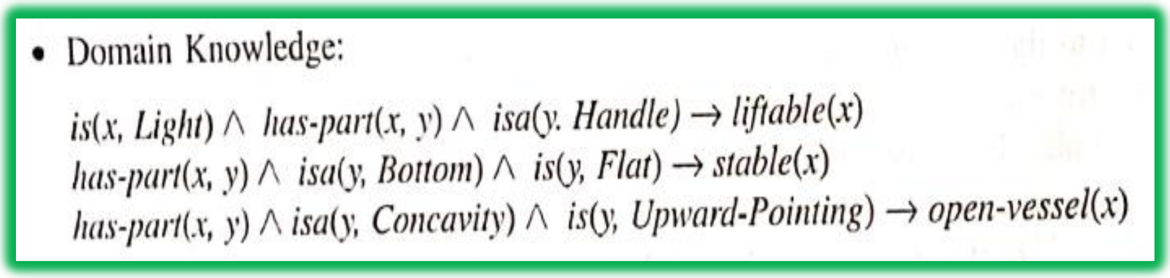
Explanation-- the **domain theory** is used to prune away all unimportant aspects of the training example with respect to the **goal concept.**
Generalization-- the explanation is **generalized as far possible** while still describing the **goal concept.**

Consider the problem of learning the concept Cup.
We want to be able to generalize from a single example of a **Cup**
Suppose the example is :



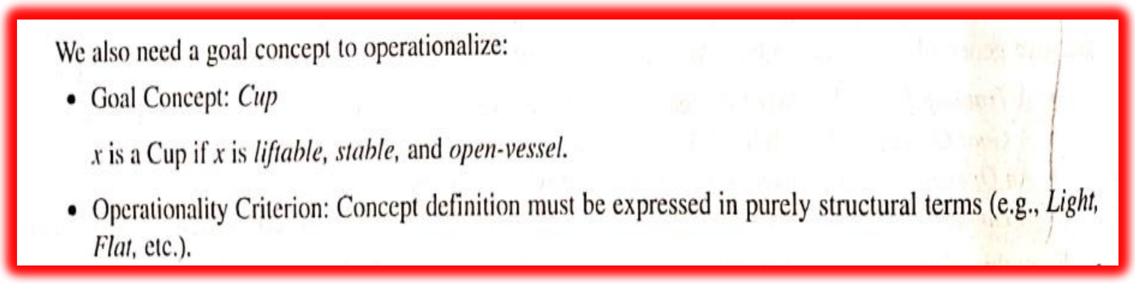
- Training Example:

  $owner(Object23, Ralph) \wedge has\text{-}part(Object23, Concavity12) \wedge is(Object23, Light) \wedge color(Object23, Brown) \wedge \ldots$



- Domain Knowledge:

  $is(x, Light) \wedge has\text{-}part(x, y) \wedge isa(y. Handle) \rightarrow liftable(x)$
  $has\text{-}part(x, y) \wedge isa(y, Bottom) \wedge is(y, Flat) \rightarrow stable(x)$
  $has\text{-}part(x, y) \wedge isa(y, Concavity) \wedge is(y, Upward\text{-}Pointing) \rightarrow open\text{-}vessel(x)$



We also need a goal concept to operationalize:

- Goal Concept: *Cup*

  x is a Cup if *x* is *liftable*, *stable*, and *open-vessel*.

- Operationality Criterion: Concept definition must be expressed in purely structural terms (e.g., *Light*, *Flat*, etc.).

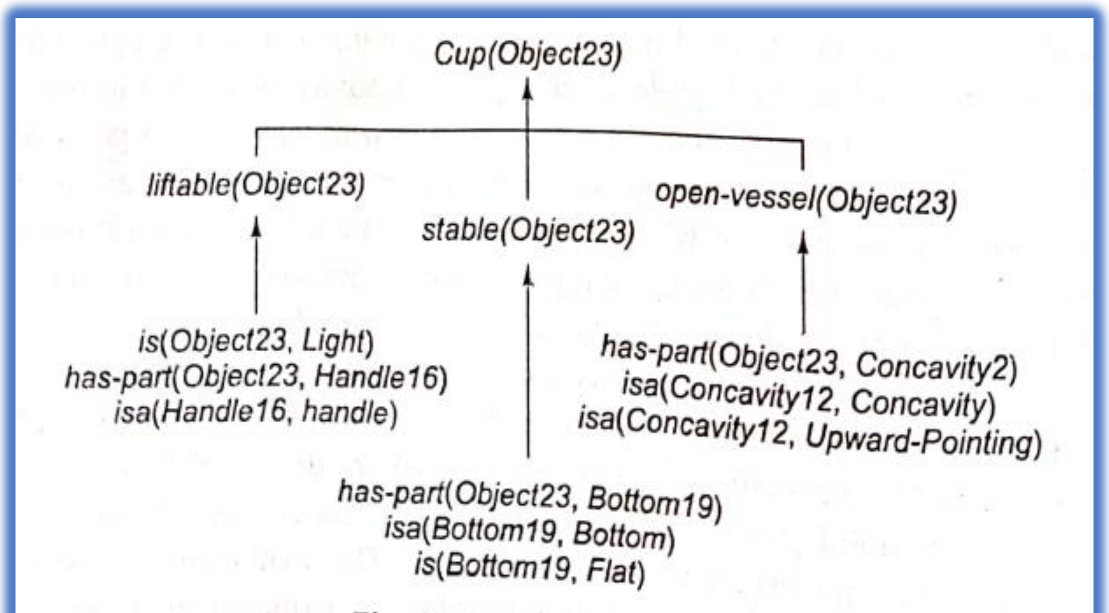
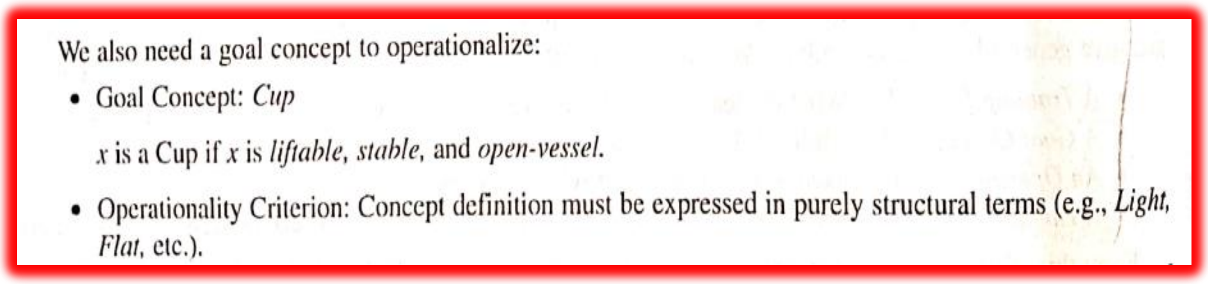


Fig. 17.15 *An Explanation*


We also need a goal concept to operationalize:

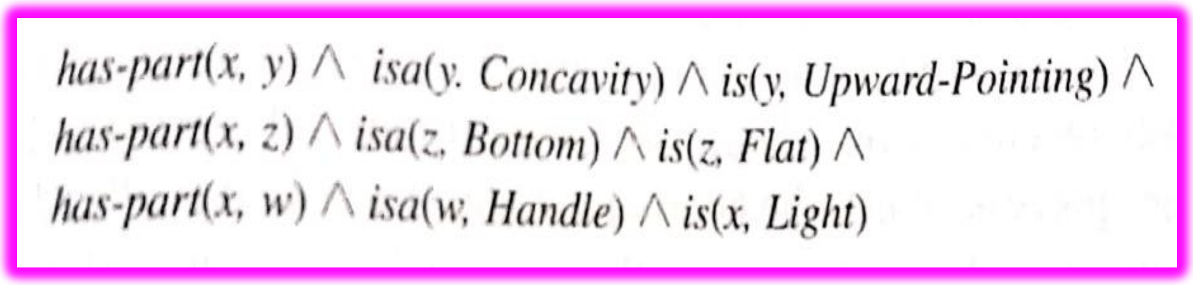
- Goal Concept: *Cup*

  x is a Cup if *x* is *liftable*, *stable*, and *open-vessel*.

- Operationality Criterion: Concept definition must be expressed in purely structural terms (e.g., *Light*, *Flat*, etc.).

➢ Given a training example and a functional description, we want to build a **general structural description of a cup.** The first step is to explain why **object23 is cup**

Generalization: we gather all the assumptions and replace **constants with variables,** we get the following description of a cup


$$has\text{-}part(x, y) \wedge isa(y, Concavity) \wedge is(y, Upward\text{-}Pointing) \wedge$$
$$has\text{-}part(x, z) \wedge isa(z, Bottom) \wedge is(z, Flat) \wedge$$
$$has\text{-}part(x, w) \wedge isa(w, Handle) \wedge is(x, Light)$$

9. a) Explain the role of knowledge acquisition in the development of expert systems.

From human experts, books, documents, sensors or computer files and converting it into a form that can be stored and manipulated by the computer for purposes of problem solving.

It occurs throughout the entire development process. Knowledge Acquisition Process

i). Identification: Identify the problem including data, criteria for solutions to meet, available resources, etc.

ii). Conceptualization: Determine the key concepts and relationships by characterizing the data, flow of information, the domain structure, etc.

iii). Formalization: Understand the underlying search space, uncertainty issues, etc. iv). Implementation: Translate acquired knowledge into the program. v). Testing: Validate and verify.

The most useful knowledge acquisition programs are those that are restricted to a particular problem-solving paradigm, e.g., diagnosis or design.

For example :- if the **paradigm is diagnosis**, then the program can structure its **knowledge base around symptoms, hypotheses, and causes.**

We now examine two knowledge acquisition systems

Knowledge Acquisition Methods

Knowledge acquisition methods constitute tools used for the process of modeling knowledge.

There are two basic strategies in knowledge engineering:

i) Starting from general and overall concepts, gradually leading the expert to elicit details of a

topic.

ii) Starting from the details of specific cases and helping the expert establish and derive

general concepts from the specific examples.

Therefore, the methods of eliciting knowledge can be categorized into two broad groups:

• Top-down (or deductive) methods

• Bottom-up (or inductive) methods

These **programs** provide support for the following activities:

- Entering knowledge.
- Maintaining knowledge base consistency.
- Ensuring knowledge base completeness

The most useful knowledge acquisition programs are those that are restricted to a particular problem-solving paradigm, e.g., diagnosis or design.

For example :- if the **paradigm is diagnosis**, then the program can structure its **knowledge base around symptoms, hypotheses, and causes.**

We now examine two knowledge acquisition systems in detail.

1) **MOLE**
2) **SALT**

MOLE [Eshelmam 1988] is a knowledge acquisition system for heuristic classification problems, such as diagnosing diseases.

An expert system produced by MOLE accepts input data, comes up with a set of candidate explanations or classifications that cover (or explain) the data, then uses differentiating knowledge to determine which one is best.

The **process is iterative**, since explanations must themselves be justified, until ultimate causes are ascertained.

MOLE interacts with a domain expert to produce a knowledge base that a system called MOLE-p (for MOLE-performance) uses to solve problems.

Initial knowledge base construction.

Refinement of the knowledge base.

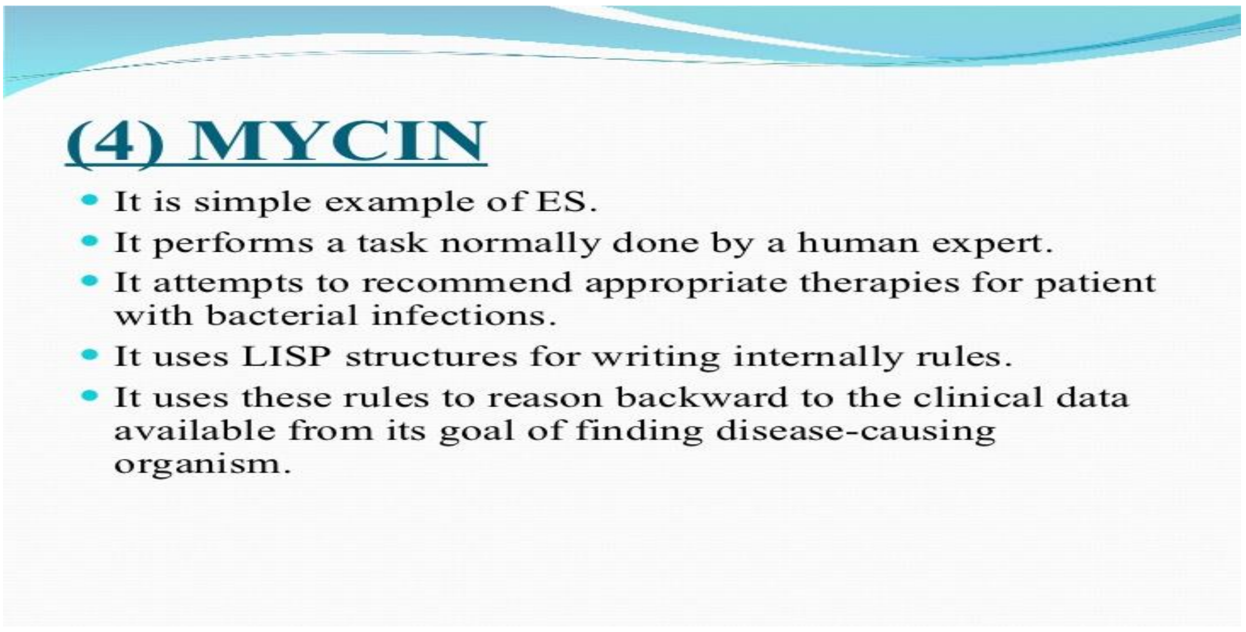
9. b) Define expert system and write the importance of MCYCN to diagnose common diseases based on symptoms.

A computer program that exhibits intelligent behaviour is an expert system (ES).

or

An ES is basically a software program that tries to perform tasks similar to human experts in a specific domain of the problem.
Expert systems may also be referred to as knowledge-based expert systems.

Expert systems represent their knowledge and expertise as data and rules within the computer systems; these rules and data can be called upon whenever needed to solve problems.



(4) MYCIN

- It is simple example of ES.
- It performs a task normally done by a human expert.
- It attempts to recommend appropriate therapies for patient with bacterial infections.
- It uses LISP structures for writing internally rules.
- It uses these rules to reason backward to the clinical data available from its goal of finding disease-causing organism.

- - - - -