Chapter - I

* AI -> • The automation of activities that we associate with human thinking like decision making, problem solveing and learning. This relates to making computers think like humans.

or

- The study of computations that make it possible to percieve reason and act. This relates to systems that think rationally.
  (understand logically) (analysis)

or

- The art of creating machines that perform functions that require intelligence when perform by people.

- The study of how to make computers do things at which at the moment people are better. This relates to systems that act like humans.

- Branch of computer science that is concerned with the automation of intelligent behaviour.

Turing Test -> It was designed to provide a satisfactory operational defination of intelligence. Turing define intelligent behaviour as the ability to achieve human level performance in all cognitive tasks sufficient to fool an

Sr. No.

interrogator. The test he proposed is that the computer should be interrogated by human via a telatype and passes the test if the interrogator cannot tell if there is a computer or a human at the other end. For this the computer would need to possess the following capabilities:-

① Natural Language Processing (NLP) - to able it communicate successfully in english or some other language.

(2) Knowledge Representation :- It is to store information provided before or during the interrogation

③ Automated Reasoning → To use the stored information to answer questions and to draw new conclusions.

(4) Machine learning → It is to adapt to new circumstances and to detect and extrapolate patterns.

[other version of turing test]

Total Turing Test → in the total turing test a vedio signal is included so that the interrogator can test the subject's perceptual abilities as well as the opportunity for the interrogator to pass physical objects through the hatch. To pass the total turing test the computer

Sr. No.

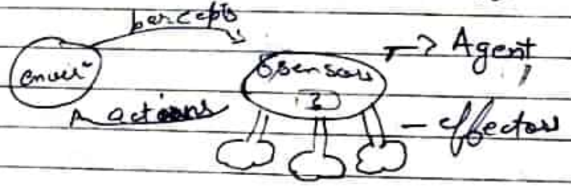
will need:-

① Computer vision to percieve objects
② Robotics To move the object.

xxx

Chapter - 2

- Agents → An agent is anything that can be viewe as perceiving its environment through sensors and acting upon that environment through effectors.



Rational A rational agent is the one that does the right thing.

Performance Measure :- It is the criteria that how successful an agent is.

Omniscience → An omniscient agent knows actual outcome of its actions and can act accordingly. Omniscience is impossible in reality.

* Rationality at any given time depends on 4 things:-
① Performance measure that defines the degree of success.

② Everything that the agent has percieved so far which is called percept sequence.

③ What the agent knows about the envirⁿ.

④ The actions that the agent can perform.

Ideal Rational Agent →

For each possible percept sequence an ideal rational agent should do whatever action is expected to maximize of its peformance measure on the basis of the evidence provided by the percept sequence or whatever the builtin knowledge the agent has.

- Autonomous Agents → If the agents actions are based completely on builtin knowledge it need pay no attendion to its percepts then we say that the agent lacks autonomy. A system is autonomous to the extende that its behaviour is determined by its own experience

actical

Prolog

↳ A goal G is True that is satisfiable or logically follows from the program iff :-

① There is a clause C in the program such that there is a clause instance i of c such that the head of i is identical to G and all the goals in the body of i are true.

structure →

eg date (25, 08, 2025).

eg seg (point(1,1), point (10, 10)].

; → or , → and

$P :- Q \overset{or}{;} R \Leftrightarrow P :- Q.$
$P :- R.$

$P :- Q, R; S, T, U \Rightarrow P :- \underbrace{(Q, R)}_{T'w} \overset{or}{;} (S, T, U).$ This

Que 2 people are relative

① 1 is predecessor of other

or ② They have a common predecessor

or ③ They have a ——— successor

Page Discriptions :-

P → Percepts
A → Actions
G → Goals
E → Environment

Ex 1 → Taxi Driver Agent

ⓐ The percepts involved → camera, speedometer, GPS, microphone,

ⓑ Actions → steer, accelerate, brake, talk to passenger

ⓒ Goals → safe, fast, legal, comfortable trip, maximise profit.

ⓓ Environment → Roads, other traffic, pedestrians, customer

Ex2 Medical Diagnosis system →

ⓐ Percepts → symptoms, findings, medical history, patient answers.
ⓑ Actions → Questions, tests, prescription.
ⓒ Goals → healthy patient, minimise costs.
ⓓ Env → Patient, hospital.

Ex 3 Interactive English Tutor

Percepts → typed words
Actions → suggestions, corrections, point exercise
Goals → maximize student score on tests.
Envi" → students,

☆
Ex 4 Part-picking Robot

Percepts → pixels of various intensity/actions identities.
Actions → pickup parts and sort into bins
Goals → Place ——— in correct bins.
Env → conveyer-belt with parts.

Properties of Environment :-

Accessible	Inaccessible
① If an agent's sensory apparatus gives an access to the complete state of envi". then we say envi" is accessible to that agent.	→ An envi" is effectively accessible if the sensors detect all aspects that are relevant to the choice of action.

Deterministic	Non-Deterministic
→ If the next state of the envi" is completely determined by the current state and the actions selected by the agents then we say the envi" is deterministic.	→

Episodic	Non - Episodic.
→ The agents experience is divided into episodes. each episode consists of agent perceiving and then acting. The quality of the action depends on the episode itself because subsequent episodes do not depend on what actions occur in previous episodes.	eg chess

~~Static~~ Dynamic	Static
→ If the environment can change while an agent is deliberating then the envir" is dynamic for the agent	, otherwise it is static.
→ If the envir" does not change with the passage of time but the agents performance score does change then the envir" is semi - dynamic.	

Discrete / Continuous

⇒ If there are a limited no of distinct clearly defined percepts and actions, we say the env is discrete. otherwise it is continuous.

03.09.2025.

Ch-3

Solving Problem By Searching

- The problem solving agent is a kind of goal bas agent that decides what to do by finding sequenc of action that lead to the desirable states. The process of looking for such a sequence is called search. A search Algorithm takes a problem as input and returns a solution in the form of an action sequence.

- The Ist step of problem solving is goal formulatio based on the current situation. 2nd step. Problem is follows goal formulation and is the process of deciding what actions and states to consider.

- Once a solution is found, the action it recommend can be carried out under the execution phase.

Sr. No.

Uninformed Search — $O(d^{+1})$

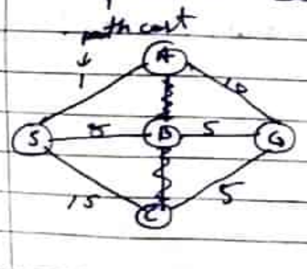
① <u>Breadth First Search</u> → It starts from 1st node and explore for the goal node from 1st then go to next level and explore all nodes at that level until it reaches goal node.

— It is done with the help of Queue.

OR

∴ In BFS the root node is expanded 1st then all nodes generated by root node are expanded next and then their successors and so on. So all the nodes at depth 'd' in the search tree are expanded from the nodes at 'd+1', So if there is a solution BFS is guranteed to finded and if there are several solutions BFS will always find the shallowest goal state 1st. BFS is complete and it is optimal provided the path cost is a non-decreasing funⁿ of the depth of the node.

② Uniformed Cost Search



$S \xrightarrow{1} A \xrightarrow{10} G \Rightarrow 11$

$S \xrightarrow{5} B \xrightarrow{5} G \Rightarrow 10$

UCS modifies the BFS by always expanding the lowest cost node on the

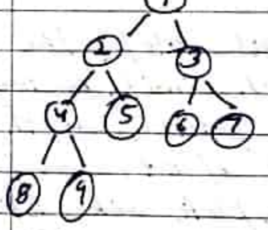
Sr. No.

fringe. [jo possible hai] measured by the path cost G(n) rather than lowest depth node.
It finds the cheapest solution, given a simple reqⁿ is meet that the cost of the path must never ↓ as we go along the path

$G(\text{successor } n) > G(n)$

eg $S \xrightarrow{15} C \xrightarrow{5} G$

③ <u>Depth First Search</u>



Search like ⇒ 1 → 2 → 4 → 8 → 9 then
5 then
3 → 6 7
let it be goal

DFS expands one of the node at the deepest level of the tree only when the search hits the dead end, does the search go back and expand node at a shallower level.

Time complexity for DFS is $O(b^m)$

DFS may be faster than BFS because it has a good chance of finding a solution after exploring only a small portion of the whole space.

H | **A* Search** → $f(n) = g(n) + h(n)$

time complexity
- worst Case → $O(b^m)$

* Greedy Search esteemated cost to the goal → H(n) and hence cut the search cost but it is neither optimal nor complete.

* Uniform Cost Search minimizes g(n) → cost of the path so far. It is optimal and complete but can be inefficient sometimes. So A* Search combines the two costs g(n) and h(n) by summing them up giving f(n) as the esteemated cost of the cheapest solution to 'n'
- This is both complete and optimal given that the fun 'H' is admisible → This implies that 'H' fun" never over esteemates the cost to reach the goal

BFS (Best First Search)

① Place a starting node on queue.
② If the queue is empty return failure and stop.
③ If the Ist element on the queue is a goal node 'G' return success and stop otherwise.

④ Remove the Ist element from the queue. expand it and compute estimated goal distances for each child. Place the children on the queue and arrange all the elements ~~tougo~~ in ascending order corresponding to goal distance from the front of the queue.

⑤ Return to step 2

A*

① Place the starting node 'S' on open.
② If Open is empty stop and return failure.
③ Remove from open node 'N' that has the smallest value of $F^*(n)$. If the node is a goal node return success and stop Otherwise.
④ Expand 'N' generating all of it's successors N' and and place 'N' on closed. For every successor N' If N' is not already on open or closed attach a back pointer to N, compute F^* N' and place it on open

⑤ Each N' that is already on open or closed should be attached to back pointers which reflect the lowest G^* of N' path. If N'

(Not > And > OR > → > ⇔)   In valid → atleast 1 false

Ch- Formalize Symbolic Logic

(Refer Search ch: basic)

Symmentics :- → [T / F] truth values possible

An interpretation for a sentence or a group of sentences is ~~a~~ assignment of a truth value to each propositional symbol.

$p \to q \Rightarrow \boxed{\sim p \vee q}$

$p \leftrightarrow q \Rightarrow (p \to q)$ and $(q \to p)$

Properties ~~of~~ :-

① satisfiable :- A statement is satisfiable if there is some interpretation for which it is ~~a~~ True. [atleast 1]

② Contradiction :- a sentence is contradictory or unsatisfiable if there is no interpretation for which it is true.

③ Tautology / Valid :- A sentence is valid if it is true for every interpretation.

④ Equivalence :- 2 sentences are equivalent if they have the same Truth value under every interpretation.

> if 1st T then 2nd will also be T.

⑤ Logical Consequence → A sentence is ~~a~~ a logical conseq of another if it is satisfied by all interpretation which satisfies the first.

Our The sentence S is logical Cons of $S_1 - - - S_n$ iff S_1 & S_2 & - - S_n & $\sim S$ is inconsistent.

Equation :- [Discrete Ch']

①

③ Commutativity

②

① Modus Ponens

p

and $p \to q$

q

②eg Job is a father
and Job ——— → Job has a child.
Job has a child.

② Chain Rule

$p \to q$

and $q \to r$

$p \to r$

eg. Given Programmer likes lisp
→ ——— hates cobol
and ———
→ ——— like recursion
Programmer likes lisp → recursion.

③ Substitution
if s is a valid sentence, s' derive from s by consistent substitution of propositions in s is also valid.

$p \vee \sim p \Rightarrow$ True

$Q \vee \sim p \Rightarrow$ True (Valid)

④ Simplificatiation :-

$p \wedge q = p$

⑤ Conjunction

$p \wedge q \Rightarrow p \& q$

⑥ Transposition

$p \to q$ infer $\sim p \to \sim q$

Formal System

It is a set ~~system~~ of axioms 'S' and set of inference Rule 'L' from which new statements can be logically derived.

It is denoted as $<S, L>$ or K_b

Soundness

Let $<S, L>$ be formal system, we say inference procedures 'L' are sound if and only if any statements 's' that can be derived from $<S, L>$ is a logical consequence of $<S, L>$

Completeness

Let $<S, L>$ be formal system then inference procedure 'L' is complete iff any sentence 's' logically implied by $<S, L>$ can be derived

using that procedure.

First Order Predicate Logic [FOPL]

① Connectives $\Rightarrow \sim, \&, \vee, \to, \leftrightarrow$

and/conjun ↑   ↑ or

② Quantifiers ⇒ existancial ($\exists$), universal ($\forall$)

③ Constants ⇒ no capital, in begining of Alpabets. (a, b, c ...)

eg numbers

(These are fixed value term that belong to a given domain of disclose.

④ Variables ⇒ These are terms that can assume diff^n values over a given domain.

⑤ Functions ⇒ fun^n symbols denote relations defined on a 'D'. They map 'n' elements $\{n \geq 0\}$ to a single element of the domain.

⑥ Predicates :- symbols denote relations or functional mappings from the elements of a Domain 'D' to the values T or F.

(use capital letter)

Like functions predicates may have 'n' ~~terms~~ arguments written as ~~[illegible]~~ $P(t_1, t_2, \ldots t_n)$, where $t_1, \ldots t_n$ are defined over some domain.

and it's pointer was changed remove it and place it on open.

6. Return to step 2.

Conditions :-

① **Admissiability** → a heuristic search algo is admissiable if it is guaranteed to return an optimal solution when one exists.

② **Completeness** ⇒ ——————— is complete if it always terminates with a solution when one exists.

③ **Dominance** → Let A_1 and A_2 be admissiable algo's with heuristic estimation functions H_1^* and H_2^* respec. A_1 is said to be more inform than A_2 whenever $H_1^*(n) > H_2^*(n)$ $\forall n$. A_1 is also said to dominate A_2.

④ **Optimality** :- Algo A is optimal over a class of algo's if A dominates all the members of the class.

Means End Analysis :-

The problem state of MEA has an initial state, one or more goal state. A set operators 'O' with given pre cond^ns and a diff^n function that computes the diff^n b/w 2 states S_i and S_j.

A problem is solved using Means End Analysis by :-

① Comparing the current state S_i to a goal state S_g and computing the diff^n 'D_{ig}'

② An operator O_k is than selected to reduce the diff^n D_{ig}.

③ The operator O_K is applied if possible, if not the current state is saved, a sub-goal is created and MEA is applied recursively to solve the problem related to sub-goal.

④ If the sub-goal is solved, the saved state is restored and work is resumed on the original problem.

→ The Drawback of DFS is that it can get stuck along the long path

→ DFS should be avaided for Search Trees with large or infinite max. depths..

④ Depth Limited Search :-

DLS avoids the pitfalls (Kamiya) of DFS by imposing a cutoff on the max. depth of a path.

⑤ Iterative Deeping Search :-

This is a strategy that side steps the issue of choosing the best depth Limit. by trying all possible by trying all possible depth limits. It combines the benefit of DFS and BFS. It is optimal and complete like BFS and has modest memory requirements of DFS

$O(b^d)$

Iteration Deeping is preferred Search method when there is a large search space and depth of solution is not known.

⑥ Bidirectional Search :-

The idea behind BS is to simultaneously search both forward from the initial state and backward from the goal and stop when two

searches meet in the middle.

$O(b^{d/2})$

Heuristic Funⁿ

- One of the simplest Best First Search strategy is to min. the estimated cost to reach that goal. i.e the node whose state is judged to be closed to the goal state is always expanded first.
For most problems the cost of reaching the goal from a particular state can be estimated but cannot be determined exactly.

- A funⁿ that calculates such cost esteemated is called the heuristic fun.

- H(n) = esteemated cost of the cheapest path from the state at node 'n' to a goal state. A BFS that uses 'H' to select the next node to expand is called greedy search.

- H(n) = 0 when 'n' is the goal node.

- Greedy Search is susceptible to false start.
resembles DFS as it prefers to follow singal path all the way to the goal. but back tracks when hits a dead end.

- It is not optimal and it is incomplete because it can start down an infinite path and never try other possibilities.

Sr. No.

Well defined problems and Solution:-

- A problem is a collection of information that the agent will use to decide what to do.

Initial State:-

- The initial state is the state that the agent knows itself to be in.

- The set of possible actions available to the agent. Above are the basic elements of a problem defination.

The term operator : denotes the description of an action in terms of which state will be reached by carrying out the action in a particular state. It is also represented sometimes as a successor function 'S'. So, given a particular state 'x', S(x) returns the set of states reachable from 'x' by any singal action.

-> Together these define "the State Space" of the problem. It is a set of all possible state reachable from the initial state by any sequence of actions.

-> Path:- Path in the state space is any sequence of actions leading from 1 state to another.

-> Goal ~~state~~ Test :- which the agent can apply to a singal state description to determine if it a goal state.

It is possible that one solⁿ is prefferable

Sr. No. represented by 'G'

to another even though both reach the goal. A path cost function :- is a function that assigns a cost to a path. It could be considered that the cost of the path is the sum of costs of individual actions along the path.

=> Together the initial State, operator set, goal ~~test~~ test and path cost fun. define a problem.

The output of certain a search algo. is a solution that is a path from the initial state to a state that satisfies the goal test.

The effectiveness of search can be measured in atleast 3 ways:-

1. Does it find the solution at all.
2. Is it a good solution i.e. one with a low path cost.
3. What is the search cost associated with the time and memory required to find a solution. The total cost of the search is the sum of the path cost and the search cost.

Sr. No.

eg

S_1 - All employees earning 50K or more per yr pay taxes.

S_2 :- Some employees are sick today

S_3 - No employee earns more than the president.

Sol Pred- $E(x) \rightarrow x$ is employee

funⁿ $i(x) \rightarrow$ returns income of x

Pred $T(x) \rightarrow$ pays taxes

Pred $GE(i(x), 50k) \rightarrow i(x) \geq 50k$

$S(x) \rightarrow$ x is sick today

$P(x) \rightarrow$ x is president

① $\forall x \left(E(x) \,\&\, GE(i(x), 50000) \right\} \rightarrow T(x))$

② $\exists x \left[E(x) \rightarrow S(x) \right]$

③ $\forall x \forall y \left[E(x) \,\&\, P(y) \rightarrow \sim GE(i(x), i(y)) \right]$

Well formed. formulas (wff)

→ An atomic formula is a well formed formula

→ If P and Q are well formed formulas then $\sim P$, $P \wedge Q$, $P \vee Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ $\forall x, p(x)$. and $\exists x, p(x)$ are well formed formulas.

→ These are formed only by applying the above rules a finite no of times.

Sr. No

Ground Atom :- A predicate with no terms is called ground atom.

{ syntax error hai tho hi hai wff)

eg $\sim p \vee Q$ ✗ $\rightarrow (\sim p) \vee Q$ ✓

or

$(\sim p \vee Q$ ✗ $\Rightarrow (\sim p \vee Q)$ ✓

or $P \rightarrow Q$ ✓ $(P \rightarrow Q$ ✗

$(P \rightarrow Q)$

⇒ $\forall x \{ F(x) \vee G \} = \forall F(x) \vee G$

⇒ $\sim [\forall(x) (F(x))] = \exists x \sim F(x)$

⇒ $\sim (\exists(x) (F(x))] = \forall x \sim F(x)$

⇒ $\forall x [F(x) \& G(x)) = \forall x F(x) \,\&\, \forall x (G(x))$

⇒ Conjunctive Normal form (CNF)

$F_1 \,\&\, F_2 \,\&\, F_3 \,-\,-\, \&\, F_n$

$(P \vee Q) \vee (P \rightarrow Q) \,\&\, (P \leftrightarrow Q)$

⇒ Disjunctive Normal Form (DNF)

$F_1 \vee F_2 \vee F_3 \,-\,-\, \vee F_n$

$(P \,\&\, Q) \vee (P \vee Q)$ this to

$\{ (P \vee P \vee Q) \,\&\, (Q \vee P \vee Q)$ cnf

Real/ Sr No from book

C> # Conceptual Dependency [CD]

↳ Entities

Picture Producer (PP) | Picture Aider

↳ Actions

1) ATRANS → transfer of an abstract entity
2) ATTEND → focusing attend on a
3) CONC → Thinking about something
4) EXPEL → expelt karna
5) GRASP → holding
6) INGEST → Eating / Consuming
7) MBUILD → Building on some existing info
8) MOVE → moving
MTRANS → transfer of a mental entity / thoughts
PROPEL → push
PTRANS → physical transfer from 1 location to another
SPEAK → speak

eg Bird flew

p → Past tense

$Bird \overset{p}{\Longleftrightarrow} PTRANS$

Joe is a student

$Joe \Longleftrightarrow student$

Joe pushed the door

$Joe \overset{p}{\Longleftrightarrow} PROPEL \overset{o}{\longleftarrow} door$

Past tense   Object

Pg 141 book

Sr. No

Joe gave Sue a flower

$Joe \overset{p}{\Longleftrightarrow} ATRANS \overset{r}{\longleftarrow} [\ ]$

recieving act

$\uparrow o$ → object

flower

Joe ate some soup

$Joe \overset{p}{\Longleftrightarrow} INGEST \overset{i}{\longleftarrow} Joe$ → instrumental

$\uparrow o$ soup

do

$\uparrow o$ spoon

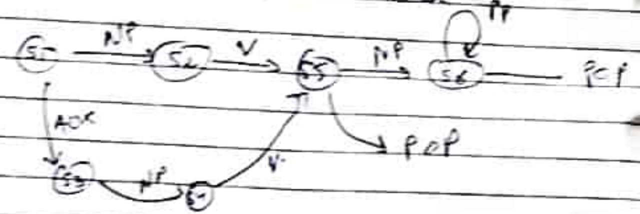
P → Past
Nil → Present (write nothing)
f → future
? → interrogative
/ → not
K → Continuous tense eg is going
r → recepent / receiving act
d → direction
o → object
T → time
loc → location
i → instrument / tool

Script → 6 marks (Read Book)

Recursive Transition Network

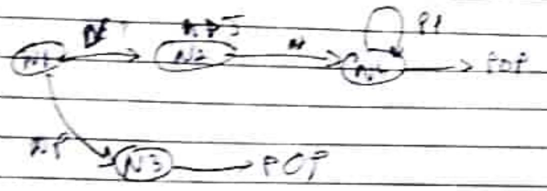
S →

Top Level RTN



Noun phrase

N



Prep Phrase RTN

~~variable~~ → current node . (CN)
position of code (POS)

CND → Current Node
POS → Position in the sentence
RLIST → Desti: pointers to return / return address

Explain properly in Exam

Constraint Satisfaction Problem

X_3 X_2 X_1 ← carry

Que →   TWO
           TWO
\+
FOUR

$0 \le T, W, O, F, U, R \le 9$
$F \ne 0, T \ne 0$
All diff (T, W, O, F, U, R)
$0 \le X_1, X_2, X_3 \le 1$

Sol

T	W	O	F	U	R	X_1	X_2	X_3
~~0~~	0	~~0~~	~~0~~	0	[illegible]	0	0	~~0~~
~~1~~	1	~~1~~	1	1	~~1~~			
~~2~~	2	2	2	2	2	1	1	1
~~3~~	3	3	3	3	3			
~~4~~	4	4	4	4	4			
5	5	5	5	5	5			
6	6	6	6	6	6			
7	7	7	7	7	7			
8	8	8	8	8	8			
9	9	9	9	9	9			

10th place

$O + O = R + 10X_1$ — ①

$W + W + X_1 = U + 10X_2$ — ②

$T + T + X_2 = O + 10$ — ③

Let T = 5
$10 + X_2 = O + 10$
$O = X_2$
if $X_2 = 0 \rightarrow O = 0$
$X_2 = 1 \rightarrow O = 1$

T = 6
$12 + X_2 = O + 10$
$2 + X_2 = O$
$X_2 = 0 \Rightarrow O = 2$
$X_2 = 1 \Rightarrow O = 3$

Chomsky Hierarchy

- Type 0 are most general with ~~rules~~ from $xyz \rightarrow xwz$. here y cannot be the empty string.

- Type 1 → these are context-sensitive grammers They have the restriction that length of the string on right hand side of the rule must be atleast as long as string LHS. Also in production of the form $ayz \rightarrow xwz$ y must a single non-terminal symbol and w are non-empty string.

- type 2 → context free grammer. It is characterised by rules with the general form <sym> written as sym1, sym2 -- till k and LHS is single non-terminal symbol.

- Type 3 → called a finite state or regular grammer whose rules are characterized by the form

$A \rightarrow aA, \; A \rightarrow B, \; A \rightarrow a$

- Languages generated by these grammers are termed type 0, type 1, type 2, type 3

⇒ Transformational Grammer

$S \rightarrow M + P$ ↳ proposition (sub, obj, verb)

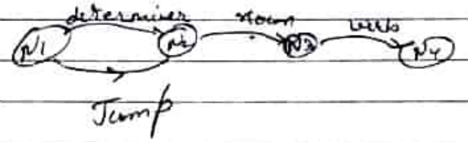
modality
(type of senten eg interogat, exclamatory)

→ Systematic Gram.

→ Segmentic Grammer →

Input → Parser → O/P token
↕
lexemes

Transition Network



Deterministic And Non - Deterministic Pars

Sr. No.

NLP

↳ Grammer 'G' is defined as $\langle v_{Nt}, v_t, S, p \rangle$ where

v_{nt} → set of non-terminal symbols

v_t → —— terminal ——

→ starting symbol

p → finite set of productions or rewrite rules.

- A 'language L' can be considered as a set of strings of finite or infinite length where a string is constructed by concatenating basic atomic elements called symbols.

- The finite set 'v' of symbols of the language is called alphabet or vocabulary. Among all possible strings that can be generated from v are those that are well-formed the sentences.

- Well-formed sentences are constructed using a set of rules called grammer. A grammer 'G' is a format specification of the sentence structures that are allowable in the language and the language generated by the grammer is denoted by L (G)

- The alphabet 'V' is the union of the disjoint set v_{nt} and v_t which includes the empty string e,

Sr. No.

- The non terminals can be decomposed further whereas the terminals cannot be decomposed. A general production rule from 'p' has the form xyz → xwz where xyz and w are strings from v.

eg

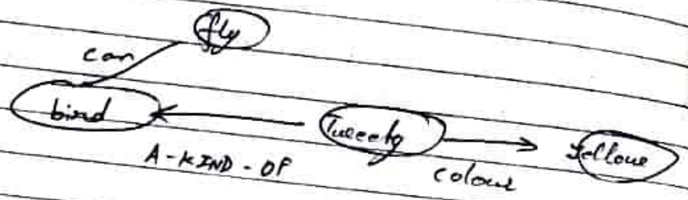
eg

Sr. No.

CHAPTER - 7

Knowledge Representation

↳ Associative Network



☆ ↳ FRAMES

```
(<frame-name> (<slot1>(<facet1> <val1> <val
               <slot2> (<facet2> <val1> <val2>)
                    :
                    :
                    ))
```

⇒ A slot may or may not have a facet and can have multiple facets.

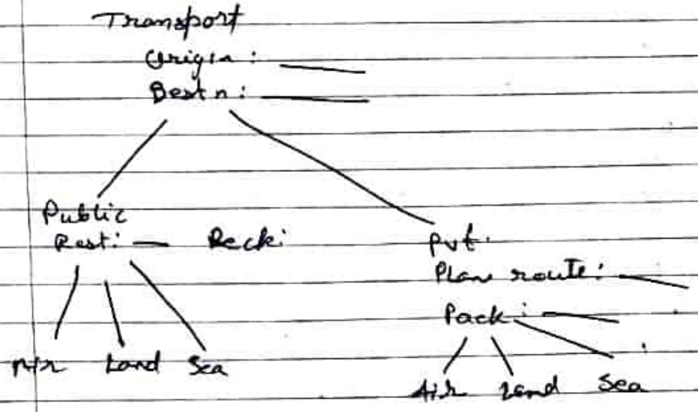
eg

```
(mayank (COURSE (VALUE Comp Sc))
        (AGE    (VALUE  19))
        ( FATHER (VALUE  Mr. Rajesh))
        (LOCATION (H.No (VALUE 42))
                  STREET (VALUE B.Mata Colony))
                  CITY(VALUE  Faridabad))
        )
```

Sr. No

```
              A-Kind-of
              ↑
eg ⇒ (Ford   (AKO (VALUE Car))
             (COLOR (VALUE Red))
            (MODEL (VALUE   4-door))    get
            (GAS-MIL (DEFAULT fget))    from
            (RANGE (VALUE if needed))   some
            (WEIGHT (VALUE 2600))       where
                                        else
```

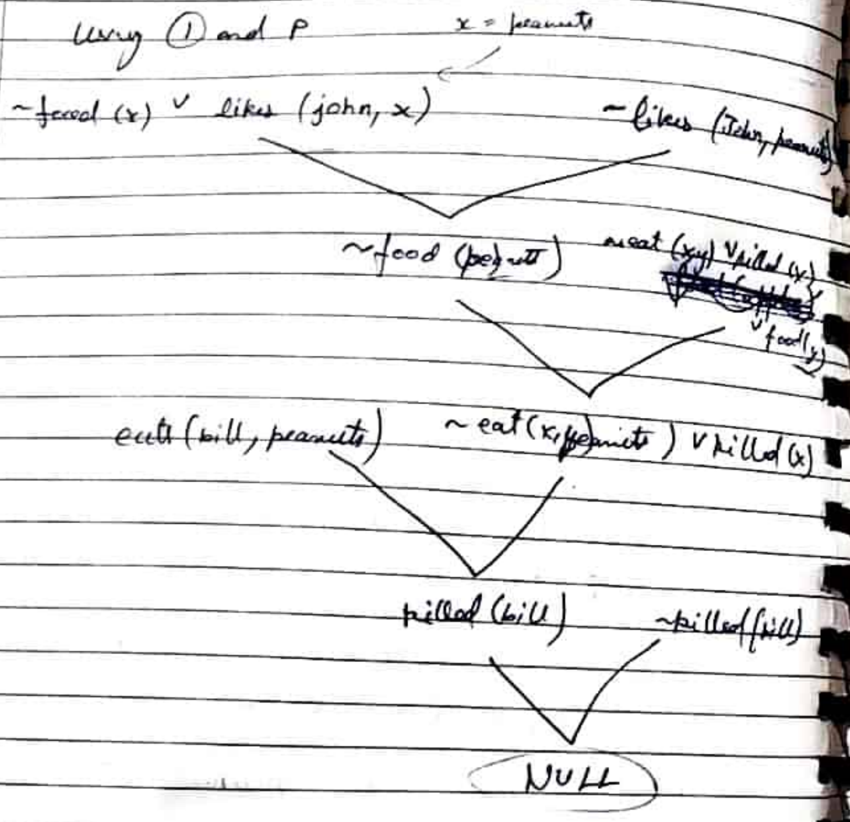
eg FRAMES



Standard Names

① fget → gets value from another frame
② fslot frame → gives names of slots
③ ffacet → →
④ fput fslotname ffacetname
⑤ fremove

Using ① and P



Unification Algorithm

Step 1 -> Set $k = 0$, $\sigma_k = \epsilon$ ← empty set

Step 2 -> If the set $S\sigma_k$ is singleton then stop. Output σ_k as MGU of S otherwise find the disagreement set D_k of $S\sigma_k$

when 2 express. diffⁿ

eg $P(f(a), x, y)$, $P(z, b, y)$ => $D_k = \{f(a), z, x, b\}$

Step 3 -> If there is a variable v and term t in D_k such that v does not occur in t. Put $\sigma_{k+1} = \sigma_k \{t / v\}$ and set $k = k+1$ and return to step 2. otherwise stop and output S is not unifiable.

eg MGU -> $\{f(a)/z, b/x\}$ ← replace with the

Example 1 $\overset{C_1}{P}$ $\overset{C_2}{(P \wedge Q) \rightarrow R}$ $\overset{C_3}{(S \vee T) \rightarrow Q}$ $\overset{C_4}{T}$

Prove P is True.

Sol ① clausal form $\Rightarrow$ $\boxed{P}$, $\sim(P \wedge Q) \vee R$, $\sim(S \vee T) \vee Q$

$\boxed{\sim P \vee \sim Q \vee R}$ $(\sim S \wedge \sim T) \vee Q$

$\downarrow$

conjunctions.

$(\sim S \vee Q) \wedge (\sim T \vee Q)$

$\downarrow$

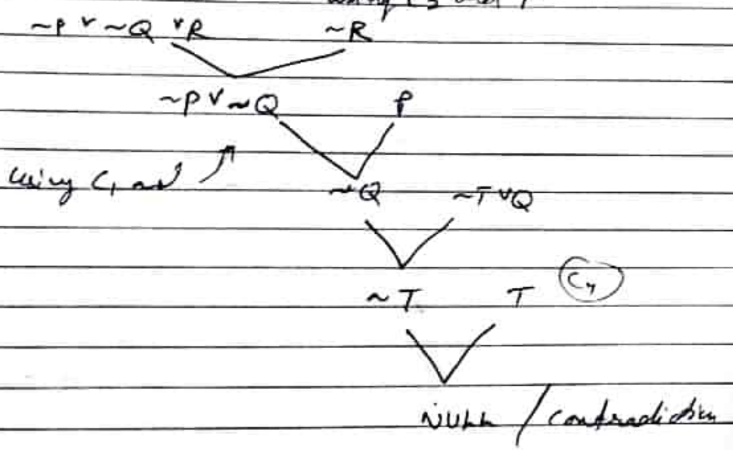
$\boxed{\sim S \vee Q}$

$\boxed{\sim T \vee Q}$

$\boxed{T}$

② $\boxed{\sim R}$ (P)

③ 2 clauses and resolve $\Rightarrow$ using C_2 and P



Example 2 $S_1 \rightarrow$ John likes all kinds of food.

$\forall x\ food(x) \rightarrow likes(John, x)$

$S_2 \rightarrow$ Apples are ~~food~~ fruit / food

food (apples)

$S_3 \rightarrow$ Chicken is food

food (chicken)

$S_4 \rightarrow$ Anything anyone is eat and is not killed is food.

$\forall x \forall y\ eats(x, y) \wedge \sim killed(x) \rightarrow food(y)$

$S_5 \rightarrow$ Bill eats peanuts and is still alive

eats (bill, peanuts) $\wedge$ alive (bill)

$S_6 \rightarrow$ Sue eats every thing Bill eats.

$\forall x\ eat(Bill, x) \rightarrow eats(Sue, x)$

Prove $\rightarrow$ John likes peanuts $\rightarrow$ likes(John, peanuts).

Sol^n ① $\rightarrow$ $\sim food(x) \vee likes(John, x)$

④ $\sim [eats(x, y) \wedge \sim killed(x)] \vee food(y)$

$[\sim eats(x, y) \vee killed(x)] \vee food(y)$

⑥ $\sim [eats(Bill, x)] \vee [eats(Sue, x)]$

P $\rightarrow$ $\sim [likes(John, peanuts)]$

⑤ eats (bill, peanuts)

alive (bill) $\rightarrow$ $\sim$ killed (bill)

Unification (MGU)

-> A substitution is defined as a set of pairs V_i, T_i where v_i are distinct variables and T_i are terms not containing the v_i. T_i replace or are substitute it for the corresponding V_i in any expression for which the substitution is applied.

Substitution ; t_i, v_i

$\beta = \{a/x, g(b)/y\}$ — Jaha x hai yaha a — y — g(b)

$C = P(x,y) \vee Q(x, f(y))$

$C' = C\beta = P(a, g(b)) \vee Q(a, f(g(b)))$

-> Any substitution that makes two or more expressions equal is called unifier of the expression.

-> Given two expressions that are unifiable such as expression C_1 and C_2 with a unifier β where $C_2 = C_1\beta$ we say that β is most general unifier if any other unifier α is ~~any~~ instant of β.

eg) $P(u,b,v)$, $P(a,x,y)$

$\alpha = \{u/a, b/x, v/y\}$ -> by subst this we can get

$\therefore$ This is unifiable and α is unifier.

Resolution Principle (5marks)

Step 1 :- Convert all the propositions into clausal form.

Step 2 :- Negative P ~~and~~ and convert the result to clausal form added to the set of clauses obtained in step 1.

Step 3 :- Repeat until either a contradiction is found or no progress can be made.

ⓐ Select 2 clauses and call them parent clauses.

ⓑ And resolve them together. The resulting clause called resolvent will be the disjunction of all of the literals of both of the parent clauses with the following ~~exception~~ exceptions:-

(i) If there are any pairs of literals L and ~L such that one of the parent clauses contains L and the other contains ~~not of~~ ~L then select 1 such pair and eliminate both L and ~L from the resolvent.

ⓒ If the resolvent is the empty clause then a contradiction has been found. If not then add it to the set of clauses available to the procedure.

Clausal Form Steps :-

Step1 Eliminate all implication and equivalent symbol

Step2 Move negation symbol into individual atoms

Step3 Rename variables if necessary so that all remaining variables, quantifiers have diff' variable assignments.

Step4 - Replace existentially quantified variables with special fun" and eliminate the corresponding quantifiers, this process is called $\boxed{\text{Skolemnization}}$.

Step5 Drop all universal quantifiers and put the remaining expression into CNF

Step6 Drop all conjunction symbols, writing each clause previously connected by the conjunction on a separate line.

——— × ——— × ——— × ———

Skolemnization :-

$\exists u \forall v \forall x \exists y \; P(f(u), v, x, y) \rightarrow Q(u, v, y)$

fun where u is replaced by a const

const a → $\forall v \forall x \exists y \; P(f(a), v, x, y) \rightarrow Q(\overset{\text{const}}{\bar{a}}, v, y)$

(it will become a fun of ∀ infront of it

$\forall v \forall x \; P(f(a), v, x, g(v,x)) \rightarrow Q(a, v, g(v,x))$

there [illegible] be no ∃ in answer

Q. $\exists x \forall y \, (\forall z \, P(f(x), y, z)) \rightarrow (\exists u \, Q(x,u) \; \& \; \exists v \, R(y, v)))$

Reference

① $\exists x \forall y \, (\exists z \sim P(f(x), y, z) \vee (\exists u \, Q(x,y) \; \& \; \exists v \, R(y,v)))$

② $\exists x \forall y \, (\exists x \sim P(f(x), y, z)) \vee (\text{———})$

③ x replace by const   replace by fun of y

④ $\forall y \, [(\sim P(f(a), y, g(y))) \vee (Q(a, h(y))) \; \& \; R(y, l(y))]$

⑤ $\forall y \, [(\sim P(f(a), y, g(y))) \vee (Q(a, h(y))) \; \& \; (\sim P(f(a), y, g(y))) \vee R(y, l(y))]$

⑥ $(\sim P(f(a), y, g(y))) \vee (Q(a, h(y)))$

$\sim P(f(a), y, g(y))) \vee (R(y, l(y)))$

Chomsky hierarchy :-

Type-0 :- that is most general with rule from $xyz \to xwz$, here y cannot be the empty string.

Type ① :- this are context sensitive grammars. they have be restriction that length of the string on the right hand side of the rule must be atleast as long as a string on the left hand side. also in production of the from $xyz \to xwz$ y must be a single non-terminal symbol and w be an non-empty string.

Type-② Grammar :- Context free grammars ⇒ it is characterize ~~with~~ rule with general forms from where k is greater or equal to 1 and RHS is single non-terminal symbol. ⇒ $\left\{ \begin{array}{l} S \Rightarrow aB \\ S \to aB \end{array} \right\}$

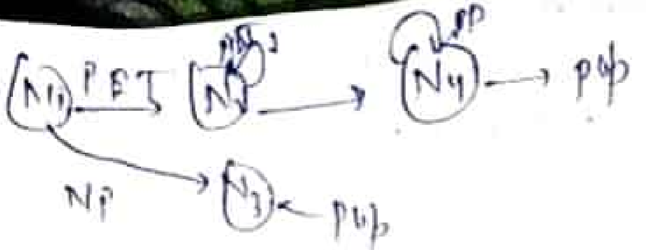
Type ③ Call the finite states or regular grammar. whose rules are characterized by the forms

$A \to aA$

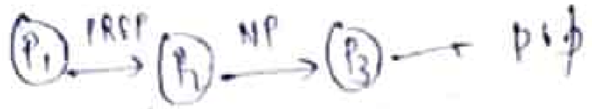
$A \to B$

$A \to a$

NP :
Noun phrase N



PP :
Prep phrase ~~the~~ ab.



* Constraint satisfaction problem :-

T W O
\+ T W O
F O U R

constraint - {T, W, O, F, U, R} ∈ {0–9}

F ≠ 0 , T ≠ 0

All different (T, W, O, F, U, R)

Sol :- (Y_3) (Y_2) (Y_1)
T W O
\+ T W O
F O U R

T W O F U R Y_1 Y_2 Y_3

[illegible]

$O + O = R + 10 Y_1$

$W + W + Y_1 = U + 10 Y_2$

$T + T + Y_2 = O + 10 Y_3$

$T = 7$

$7 + 7 + Y_2 = O + 10$

$4 + Y_2 = O$

$Y_2 = 0, O = 4$

$Y_2 = 1, O = 5$

$8 = R + 10 Y_1$

$Y_1 = 0, R = 8$

$Y_1 = 1, R = 8$

$W + W = U$

$W = 3$

$U = 6$

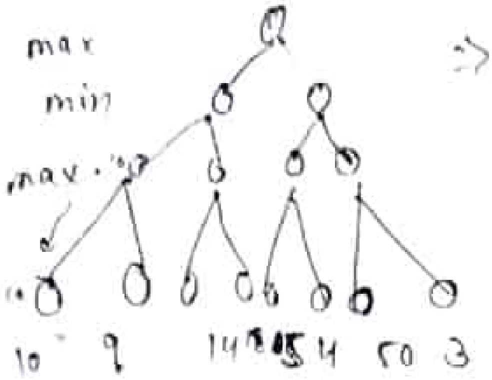
R = 8

734
\+ 734
1468

Pruning = cutoff

Q. SEND
MORE
+
MONEY

Minmax Search



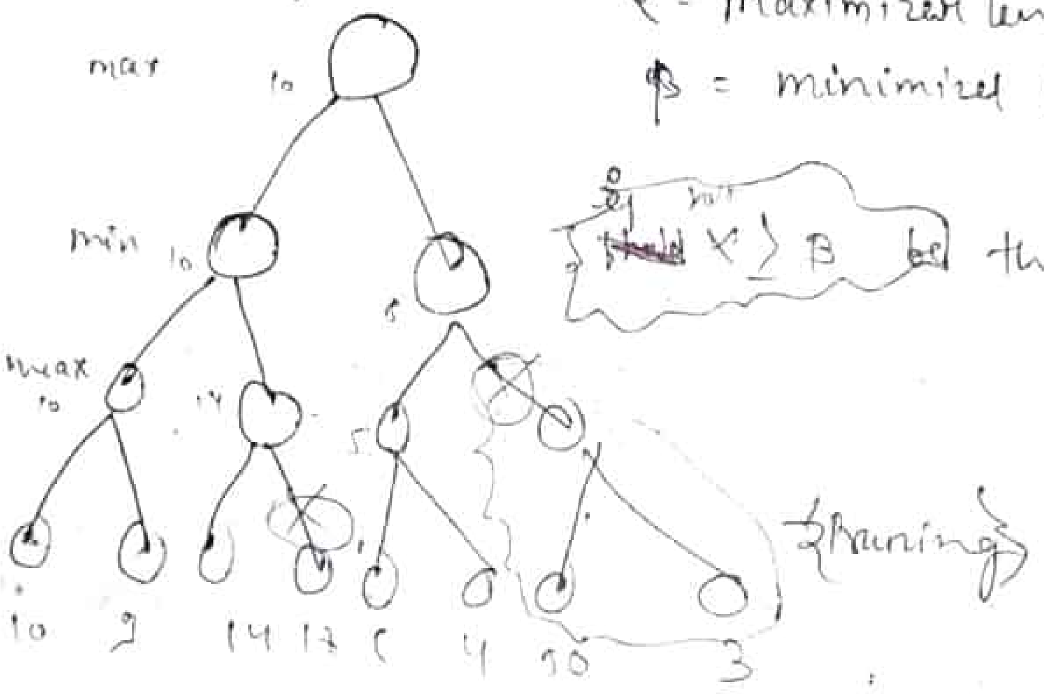
=> it is optimal but ~~tool~~ time consuming

α - β pruning:- {cutting leaves}

α - maximizer level

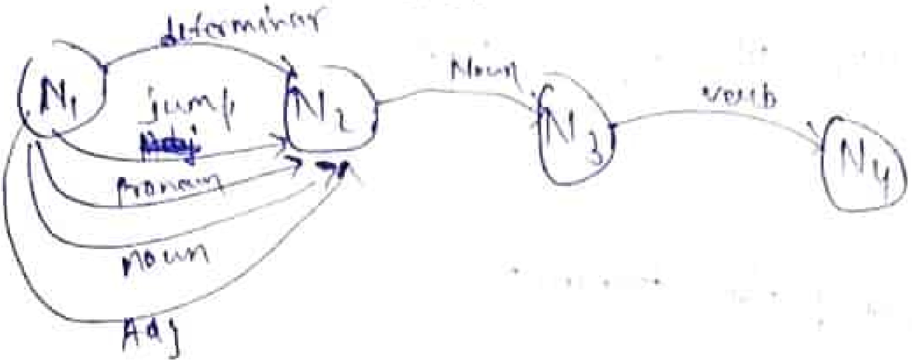
β = minimizer level

if ~~should~~ α ≥ β be then get

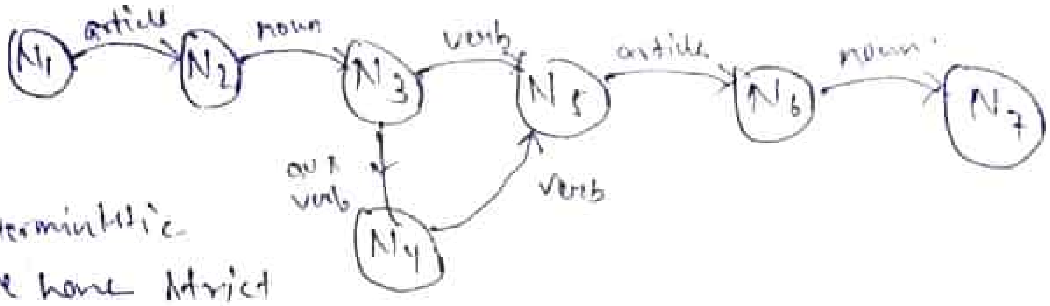


eg languages generated by these grammars are term
type - 0,1,2,3 , type-1 =>

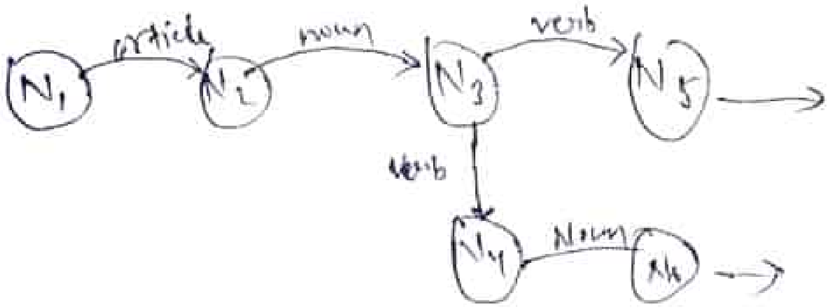
* Transition Network :-



Deterministic & Non-deterministic Parser :-



Deterministic
we have strict
rule to go the network



in Non-de
we have
possibility
means we
have differen
if one fail.

Recursive Transition N/w !-

train
top level RTn S :