Artificial Intelligence Notes

Contributor: Riya Goel

[KMV (DU)]

Computer Science Notes

3 Jan, 19

Ch-1

→ Prolog - language used in practicals.
↓
{ Programming in logic }

- science through which we make our machines intelligent so that they behave like humans.

↙ Artificial Intelligence.

{ basic idea is initial thinking process }

(1) Knowledge representation - ways to provide initial knowledge to machine.

(2) natural language processing - The language used by 2 diff. parties (client or server) must be compatible.

(3) logic building - We have certain rules to build logic. Without knowledge, a system can't be intelligent. eg - distributive law, transitive law.

(4) Adoptability to current environment - The machine adopts to changing situations, using sensors (eg - camera of robots), activators

→ Artificial - something which we create by ourselves.

CLASSTIME | Page No.
Date / /

- Scheme on which AI is based-

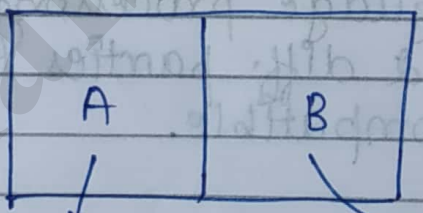
Thinking humanly.	Acting humanly
Thinking (machine)	Acting (machine)

NOTE: First test in AI was initiated by Alan Turing

↓

Turing Test (no visual contact b/w A & B)

(test encorporating intelligence)



{ w/o any visual contact, if A is able to distinguish whether the system/machine is human or machine, then AI test fails.

Interrogater → machine/system.

If A can't distinguish whether B is human or machine, then AI test is successful

↓

Basic test to test AI.

A → interrogates B.

Total turing Test

↓

When we have a visual representation (eg - cameras)

{Jade} - used for agent programming using Java.

CLASSTIME Page No.
Date / /

among A & B, then it becomes a total turing test.

* Capthas → used to check intelligence.
  ↓
  {soft bots} → internally robots but structure is like humans.

used to differentiate how much good is our intelligence.

→ Capchas are provided in combination of nos, letters, words for good websites.
  ↓ produces combination of random states.

(machine).

○ Agents - work on AI. Sense & acts on a given environment

→ The computer must possess the following capabilities to behave intelligently :-

amp.
(1) Natural language processing.

(2) Knowledge representation.

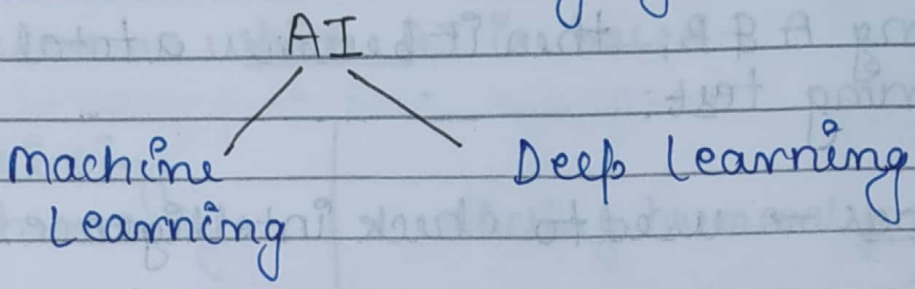
(3) Automated reasoning (feeding logic)

(4) Machine Learning (remembering the logic). once built.

Agent → program that behaves by sensing a particular condition.

CLASSTIME | Page No.
Date / /

(broad category).



→ To know behaviour of an agent, we need to know behaviour of a rational agent

ideal agent.
(produces ideal behaviour)

CLASSTIME | Page No.
Date / /

Ch-2
Intelligent Agents

~~Entity that through its~~ Sensors & activators perceives & acts on things).

- Percept - basically a sensing process. i.e sensing the environment & building up a perception and an action is built.

When we gain multiple percepts, it becomes percept sequence.

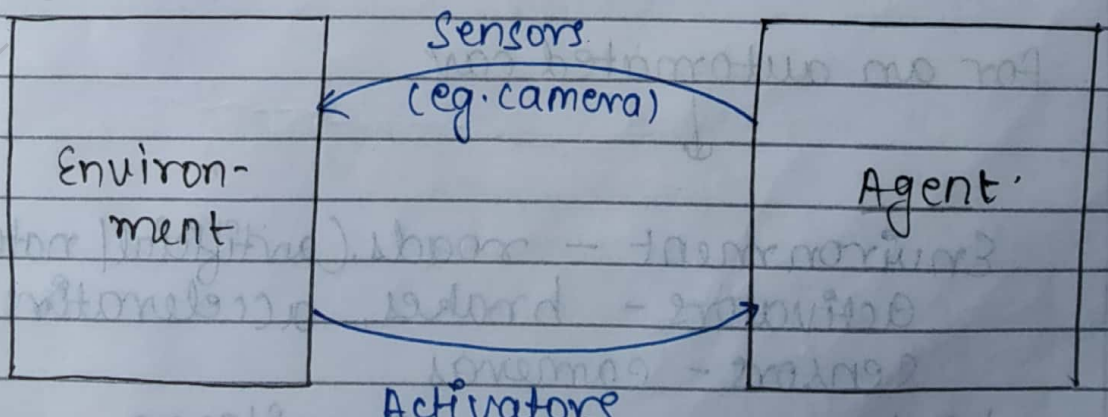
history of percepts gained throughout life.

- Function - partial implementation.
- Program - complete implementation.

Combining multiple percept → agent function

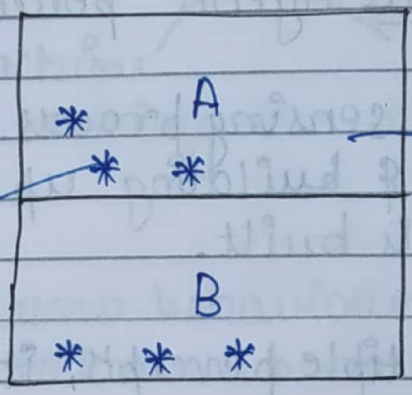
and when we combine multiple agent functions, we get agent program.

Agent sense environment with sensors then acts with activators.



eg:- (Book)

{Agent as vaccum cleaner}



dust particles.

We have 2 environments (A and B) with dust particles.

If A → dirt ⟹ Clean.
percept for vaccum cleaner. action

A → clean ⟹ move Down
B → dirt ⟹ clean.

Each agent here has a specific task to perform i.e. we didn't make a generic agent.

→ When we talk of a percept and a percept sequence, then we get a specific agent.

eg:- For an automated car

↓

Environment - roads (artificial/ natural)
Activators - brakes, accelerators.
Sensors - cameras
performance measure - mileage.

CLASSTIME | Page No.:
Date / /

NOTE:- Agent depends on environment.

PASE (PEAS)

agent's performance ach- sensors environment
vators

defines chrs. of agent.

4 Jan, 19

Software → Prolog.

?- ▯ → blink (System is ready to be questioned)

Prolog- deals with sentences and converts into high level language.
- It is a high level language
- Prolog defines relationship b/w objects.

eg- relation b/w father and son.

- finding relationship and objects in a sentence-

Basic sentence →

X is a father of Y.
Ram — relationship b/w X & Y. — Shyam.

- Define relationship name starting with small letter

father (X , Y) .
- father: clause
- X: name of father
- Y: name of father
- . : dot is end of particular statement

{X and Y are objects}

⟹ father (Ram, Shyam) .

clause

CLASSTIME | Page No.
Date / /

→ Prolog works on :-

(1) Facts → universal truths {no need for predefined objects to check its validity}.
(2) Rules ↓
{who is grandchildren of Ram}
↳ relationships already defined is in terms of parent only.

(3) questions - can be in terms of facts & rules.

$\left\{ \begin{matrix} X \to Y \\ Y \to Z \end{matrix} \right\}$ ↳ we have clubbed some 2 or more sentences, then finding out other rules, using if and else condition to our facts.

$X \xrightarrow{G \cdot P} Z$

?- □ ↓ blink.

① First step → add knowledge

File ↓
consult. ↓ provides path of file.

File ↓
New
↳ file.pl
↳ Oval icon.

'or'

Write in notepad and save with extension .pl.

CLASSTIME | Page No.
Date / /

Notepad ↴
file.pl →

father (ram, shyam). // clause.

This is predefined knowledge that system will use.

② System (questions are always asked from command prompt)

?- ↴
father (ram, shyam) // returns True.
True

?- father (ram, m)
false.

⟹ Knowledge Base is predefined information fed in system.

2 languages for AI

lisp

Prolog
(programming in logic)

Notepad
file.pl ↴

- father (ram, shyam).
- father (sohan, mohan).
- father (par, shyam).

Question - (1) whether Ram is father of Shyam or not

Simply define relationships with entities.

(2) Who is child of Ram.

NOTE:- Use capital letters for defining variables in Prolog and small letter for objects.

- father (X, Shyam)
  - X → capital letter
  - read as who is father of shyam.

Output → X = Ram // can display Ram or bar but first display relationship.

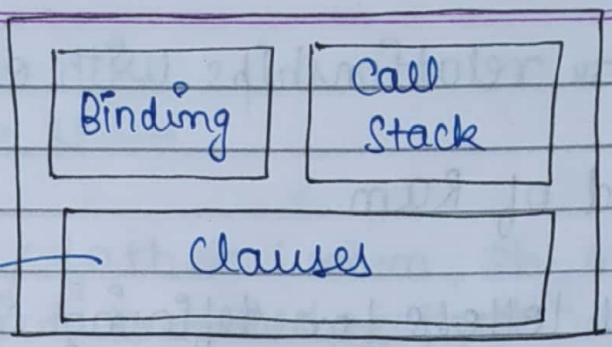
NOTE:- To display all values of X, use ';'

after execution, X = ram.
press (;) ↓
X = bar

- gtrace → Graphical Trace (for debugging)

gtrace, father (X, Shyam) ↓
- X → variable
- Shyam → object
- goes to clause father (ram, shyam)

- trace → command prompt debugging.



father (ram, shyam). (Δ) → terminate

* Notepad :-

```
father (ram, shyam).
father (pqr, shyam).
```

```
parent (ram, shyam).      // Ram is parent of shyam.
parent (shyam, sita).
```

grandparent (X, Z) :- parent (X, Y), parent (Y, Z).

If generating a new fact using two predefined facts then that new fact is not fact it is known as predicate

predicate in terms of Rule.

defined as rule, can't be defined in facts.

First program → define family relationships and define rule.

grandparent (X, Z) :- parents (X, Y), parent (Y, Z).

{sequence matters}

NOTE:- We can define 1 or 2 objects entities directly.

for multiple objects, we use predicate.

- command ↓

```
grandparent(sita, ram).   // false.
grandparent(ram, sita).   // True.

father(ram)
   ↓
Error          // 1 (no. of argument)
               // 2 (need 2 arguments)
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

7 Jan, 19

- Rational Agent - Ideal behaviour shown by an agent.

If max. outcome of an environment is known by an agent, it is rational agent.

The agent which implements things in terms of designs, etc. correctly with reference to performance measure.

to be defined by the user.

- To implement performance measure :-

① Environment - The location in real world for which an agent is defined.
W/o environment, agent can't act.
First thing of agent in a new environment is to perceive things through its sensors.

percept sequence ↓
mapping of process to corresponding action

→ Types of Environment :-

- Categories / properties of task environment.
(Accessible V/S Inaccessible)

as for each environment, we have dedicated agent.

(a) Fully Observable V/S Partially Observable.

If an agent's sensor gives it access to the complete state of the environment

at each pt. in time, then this is known as fully observable environment, whereas in partially observable environment, an agent might not be able to keep the track of all the changes because of noisy or inaccurate sensors. A vaccum agent is an example of fully observable agent & automated taxi driving is an example of partially observable environment.

Vaccum cleaner has only 2 stages - room & dirt, then it continously captures changes.

When a car infront of us, suddenly applies brakes we might / might not be able to capture these changes. {movement can't be captured by sensors}.

(b) Deterministic V/S Stochastic :-

(determined beforehand on basis of previous knowledge)

→ If next state of the environment is completely determined by the current state and action executed by the agent, then this comes under "deterministic task environment" otherwise it will be "stochastic".

{Next state = Current state + work by agent.}

eg :- automated taxi driving is a

Stochastic process

eg:- vaccum cleaner & crossword are deterministic process.

(c) Episodic V/S Sequential
↓ (at random). ↳ (flow maintained)
diff. episodes are based on diff. themes.

→ In episodic environment the agents experience is divided into atomic episodes & each episode consists of the agents perception and the corresponding single action, whereas in the sequential task environment, it doesn't depend on a single performed action.

(d) Static V/S Dynamic

→ If the environment can change while an agent is deliberately doing something, it comes under the dynamic environment otherwise the environment is static

Sudden action done by agent causes a change in environment
∴ environment becomes dynamic.

Dynamic { eg - Taxi driving as performed action of 1 car changes environment for other cars.
Also, diff. dir^n of roads.

eg - Crossword is a static process.
(environment not changing).

(e) Discrete V/S continous -

→ In a discrete environment, the percepts and the actions are not moving acc. to the time whereas in continous environment, both things are moving parallely.

perceive → perform → perceive → perform ---

continous (action acc. to time).

only talking of time frame, no use of single or multiple tasks.

each moment of time, there must be immediate action & percept.

eg:- applying brakes if driver in front of us applies brakes (in terms of reflex actions).

eg:- google assistant.

(f) Single V/S multiagent

Single agent in one environment
eg - vaccum cleaner

multiple agents in one environment
eg - taxi driving (automated).

CLASSTIME | Page No.
Date / /

- PEAS (discription about a task environment)

P - Performance measures.
E - Environment
A - Activators
S - Sensors

Table of PEAS (Read book)

(corresponds to actions)

PEAS for a Taxi driver → (name of task)

- Agent Type - Taxi driver
- performance measures - (1) Safety
  (2) Fast
  (3) Legal {corresponds to licenses and other protocols}
  (4) Comfortable Trip.
  (5) Max. mileage.
  (6) reach to the destination.
- Environment - (1) roads
  (2) Traffic
  (3) Pedestrians.
- Activators - (1) Steering
  (2) brakes
  (3) accelerator
  (4) horns
  (5) lights (front & back)
- Sensors - (1) Camera
  (2) Speedometers

(3) GPS
(4) odometers
(5) engine sensors.

- PEAS for Vaccum Cleaner

Agent Type	Performance	Environment	Activators	Sensors
vaccum cleaner	• Suction power • Electricity consumption	• room • dirt	• clean dirt	• Cameras

- Structure of Agent →

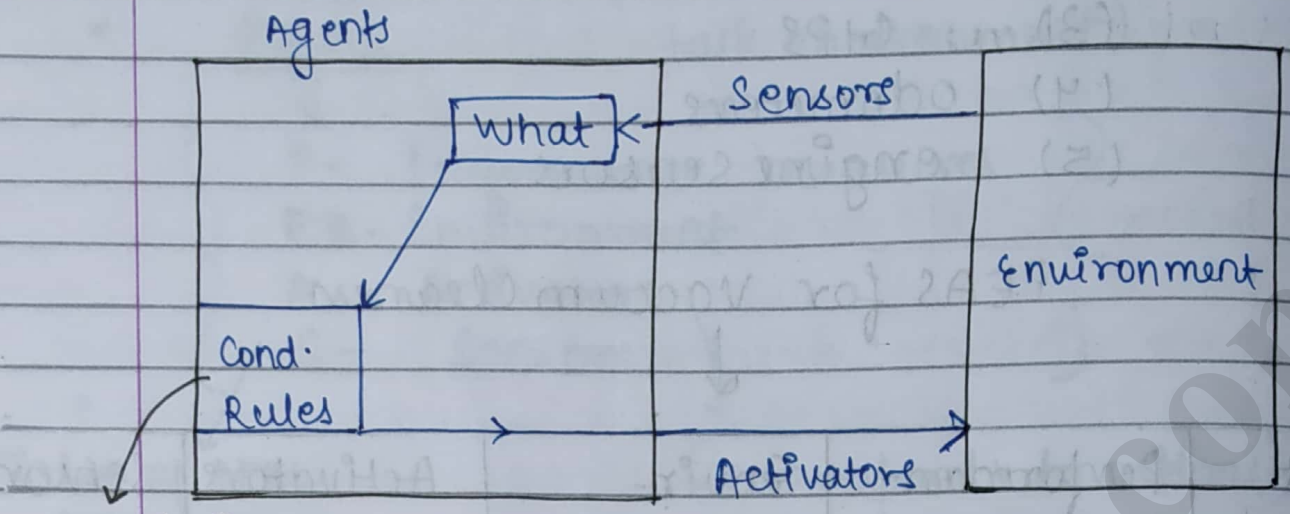
Agent = hardware + software (programs acc. to the environment).
(depending on nature of agent & environment)

* Models to develop structure of an Agent :- (based on degree of perceived intelligence and capability).

(1) Simple Reflex model -

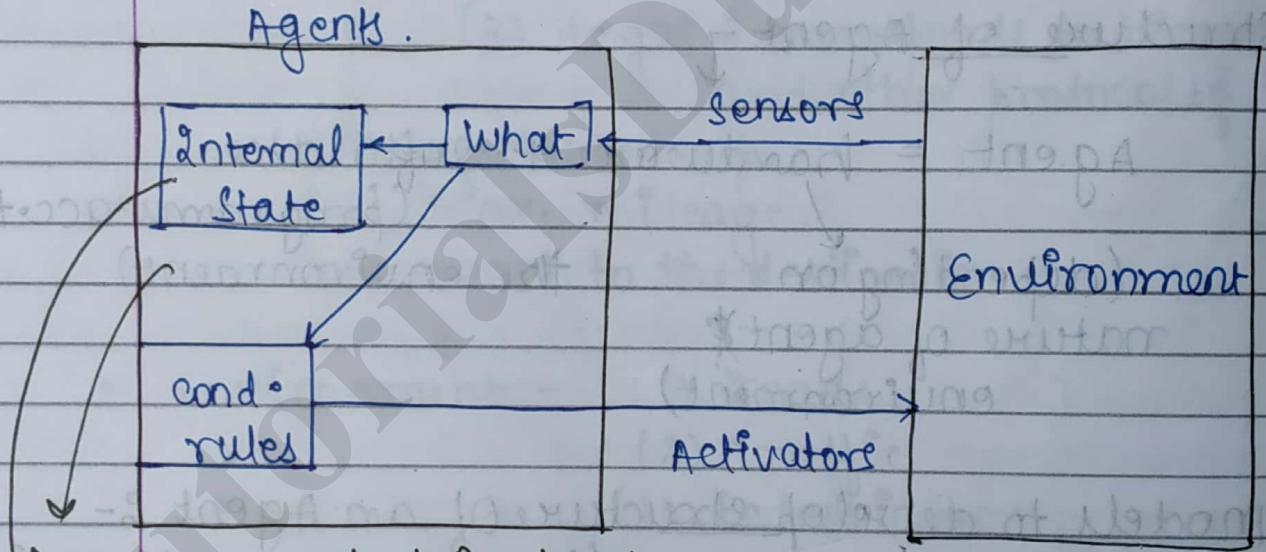
Block Diagram →

eg :- vaccum cleaner.
If A → dirt = Clean.



On analysing what we sense, an if else condition is mapped which generates an action

(2) model Based reflex agents



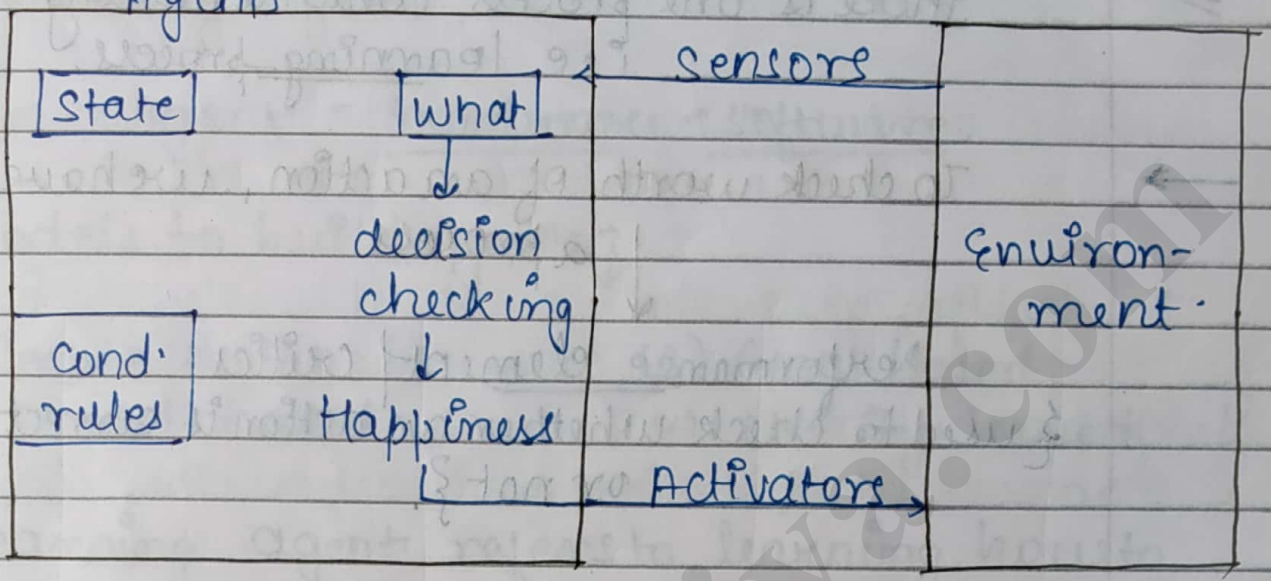
As soon as what is performed, it maps to both options and acts acc to if and else.

Info. is already stored and based on that action is performed

{ action depends on both if else and internal State }.

An agent's utility func. is essentially an internalization of performance measure. If internal utility func. & external PM are in agreement, then an agent that chooses actions to max. its utility will be rational acc. to external PM.

(3) Utility based-Agents



If an agent is for cleaning, we see how much effective it is in cleaning dust & whether agent positively grows on environment or negatively grows. and thus happiness can be determined.

{ First a decision is taken & then we satisfy happiness & we keep on going through this cycle until happiness increases }.

imp.

(4) Goal Based models (base of programming)
↳ target to be archieved.

used in practical as goal is determined.

In all the 3 models, we haven't checked about the goal achieved but in goal based models, we check whether the agent moves positively to contributing the environment.

CLASSTIME | Page No.
Date / /

NOTE:- In all these models, agent can't exist as there is one process which is lacking i.e learning process.

→ To check worth of an action, we have
↓ a process
Performance element / critics
{ used to check whether an action is correct or not }.

10 Jan, 2019

CLASSTIME | Page No.
Date / /

imp.

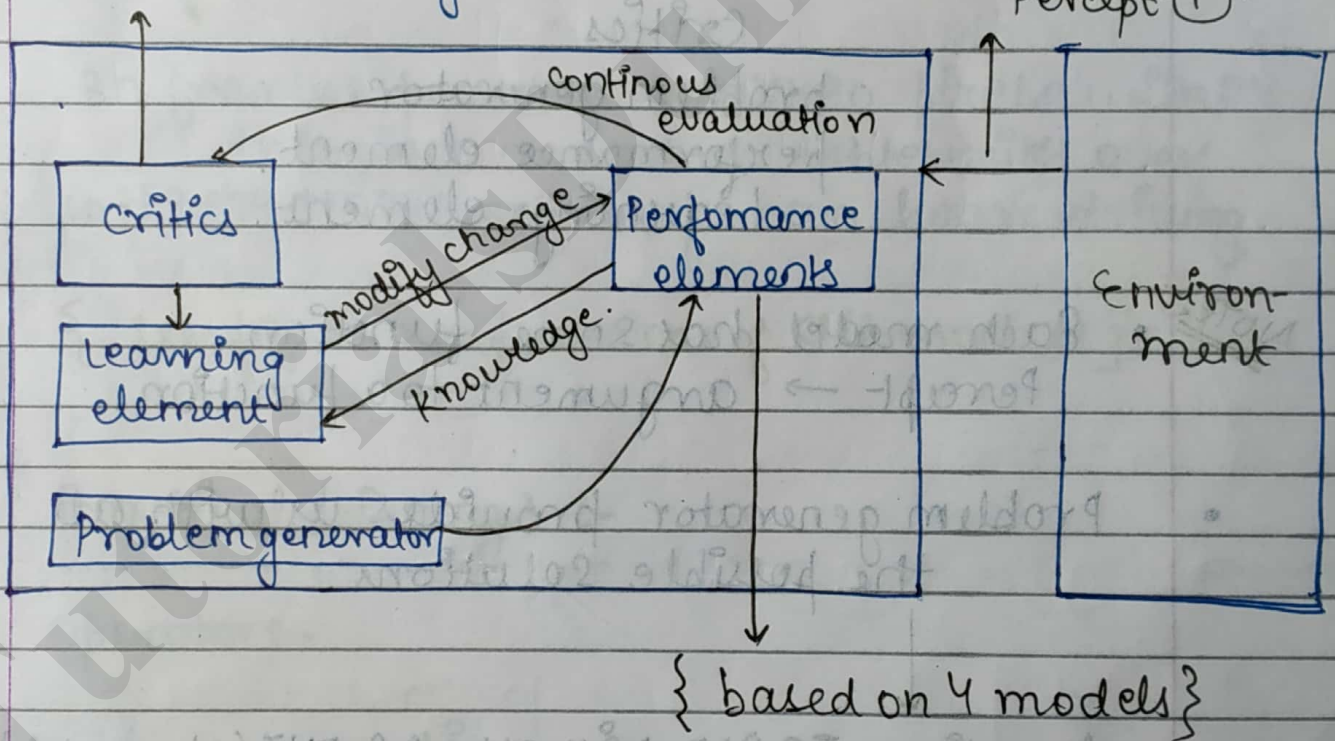
Learning Agents

Agent = hardware + software.

models to build Agents ↓

(1) Simple Reflex
(2) Model Based
(3) Goal based
(4) Utility based.

→ Learning agents refers to learning how to carry out tasks for reducing time to carry out those tasks.



→ First of all, environment generates percept and then it goes to performance element where its action is decided by the 4 models.

→ critics continously evaluates this action with corresponding standards.

→ If action is not accurate, critics will inform learning element

which in turn triggers the performance element to change the action.

- Problem generator → possible ways to incorporate same things

↓

{performance element can consult this and find out another way}

→ 4 main components ↓

Critics
problem generator
performance element.
Learning element.

NOTE:
- Each model has some function
  Percept → argument to a function.
- Problem generator provides us with all the possible solutions.

↓

function TABLE-DRIVEN-AGENT(percept) returns an action.

Static: percepts or sequence, initially empty table, a table of actions, indexed by percepts sequence initially fully specified.

append percept to end of percept action ←

LOOKUP (percepts, table)
return action.

Problem → If data is increasing, percept is changing continously with time.

let P be possible percepts, and let T be the lifetime of agent { no. of percepts }

$\sum_{t=1}^{T} |P|^t$ → lookup Table

no. of entries in lookup table.

From book ↓

——— 30 frames persec., 640X480 pixels with 24
——— , this gives a lookup table with over $10^{250,000,000,000}$ entries for an hours driving.

{ see fig 2.9 = functions of all models }

- Practical (Prolog) :-

→ Operators -

- %. - commenting
- min / max - inbuilt words for binary comparison or more.
- compare - comparison returns 3 type of values.

inferior | equal | superior.

CLASSTIME | Page No.
Date / /

- =:= , =\= → equality, inequality.
- is → variable assignment
  → strings only. {for computation purpose}.
- = → numerical values and as well as strings. {declaration}
  ↓
  assigning body in head
- =, =\= → string equality

* Functions of List →

- number
- last
- append
- length
- reverse
- min/max
- sort
- sublist

Operators
↓
mod (for remainder division)

module (modulo of 3/2 is 1)

→ a :- b, c. → end of statement.

head | body

program consists of procedure definitions..

multiple clauses are seperated by commas.

Evaluation is from LHS to RHS i.e first b is evaluated and then

CLASSTIME | Page No.
Date / /

NOTE:- A procedure is a resource of evaluating something.

- a :- b, c
  or
  a ← b ^ c
  } Same

{a is true if b is true and c is true}.

This is a program clause with head 'a' and body b, c.

→ ?- a, d, e // initial query
? signifies a query.

- multiple answers → directed backtracking.

?- happy(chris), likes(chris, bob).

→ Sum of 2 nos.

?- xyz(2,3).
X
} This won't work we need to write reading and writing statements seperately.

?- xyz. → enter ano.

xyz :- read(X), read(y), write(X+Y).

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

?- xyz (2, 3, R) → some variable result

xyz (X, Y, Z) :- Z is X+Y.

Searching Techniques

14 Jan, 19

Knowledge representation

↓

logic / Knowledge to be fed in computer (agent).

{ Book → P. W Peterson (Ch-4, 5, 6, 7) }

Ch-4
Knowledge representation

(1) Proposition Logic (PL)
(2) First order predicate logic (FOPL)

(meaningful + atomic) ← Proposition → contains atomic meaning (atomic sentences that can't be broken).

eg - It is raining // atomic sentence can't be further broken.

→ To represent atomic sentences, we have notation in AI

KR (Knowledge representation)

↓

Symbols P, X, Y, S

X = It is raining — used to denote atomic sentences.

Symbol assignment may vary.

atomic sentences which are independent of any action can be

denoted by single capital letter.

To show dependent sentences
↓
use functions
R, r, +
R(x)

eg:- it is raining in New York

can't be represented using only X

addition of an entity.

X → raining
N → New York.

{dependency/ relationship changes acc. to diff. func's used}

N(X) X(N)

2 possibilities.

A → B
B → C
A → C

} propositions

New proposition.

NOTE:- PL can't define functions and quantifiers we have only atomic sentences, but 2 or more atomic sentences can be combined to form a compound sentence.

CLASSTIME | Page No.
Date / /

- To join 2 atomic sentences
  - connectives
    - $\wedge$ → And
    - $\vee$ → OR
    - $\neg$ → Not (Unary connector)
    - $\rightarrow$ → Implication.
    - $\leftrightarrow$ → if and only if.

- While writing any sentence, we must keep in mind
  - Syntax (sentence in a sequence)
  - Semantics (meaningful sentences)

eg- It is raining ⇒ Valid sentence.

→ as we are able to get full info. and it is meaningful.
But the T/F value isn't determined & hence is not taken in consideration.
(dealing only with validity & not the truth value).

A	B	A∧B	A∨B	A→B	A↔B
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

  ↓ used in logical reference Rules.

NOTE:- 'All' can't be denoted by PL. It is a part of FOPL.

CLASSTIME | Page No.
Date / /

Q Given a representation ↓

$((P \wedge \sim Q) \rightarrow R) \vee Q$

check its validity.

P	Q	~Q	R	P∧~Q	P∧~Q→R
T	T	F	T	F	T
T	T	F	F	F	T
T	F	T	T	T	T
T	F	T	F	T	(F) → one value F ∴ not valid
F	T	F	T	F	T
F	T	F	F	F	T
F	F	T	T	F	T
F	F	T	F	F	T

- Properties of Statement -

(1) Satisfiable - A statement is satisfied if there is some interpretation for which it is true

final colum of Truth Table has atleast one T.

(2) Contradiction - A sentence is contradictory if there is no interpretation for which it is true.
(opp. of satisfiable).

(3) Valid - A sentence is valid iff. it is true for every interpretation (it is also known as tautology).

CLASSTIME | Page No.
Date / /

(4) Equivalence - 2 sentences are equivalent if they have the same truth value under every interpretation.

(5) Logical consequence - A sentence is a logical consequence of another if it is satisfied by all interpretations which satisfies the first one.

$(P \wedge Q)$

P is logical consequence of P and Q.
i.e if $P \wedge Q$ is valid, P is valid
if $P \wedge Q$ is contradictory, P is contradictory.

* Inference Rules (Equivalence rules)

(1) $P \vee P = P$, $P \wedge P = P$ } Idempotent Law.

(2) $(P \vee Q) \vee R = P \vee (Q \vee R)$, $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$ } Associativity.

(3) $P \vee Q = Q \vee P$
$P \wedge Q = Q \wedge P$
$P \leftrightarrow Q = Q \leftrightarrow P$ } Commutativity.

(4) $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$
$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$ } Distributive law

(5) $\neg (P \vee Q) = \neg P \wedge \neg Q$
$\neg (P \wedge Q) = \neg P \vee \neg Q$ } De morgan's Law

imp. (6) $P \rightarrow Q = \sim P \vee Q$ } conditional.

imp. (7) $P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$ } Biconditional.

Proof-

P	Q	$P \rightarrow Q$	$Q \rightarrow P$	$P \rightarrow Q \wedge Q \rightarrow P$	$P \leftrightarrow Q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

equal

* Inference Rules of PL-

(1) Modus ponens.

$$\frac{P, \quad P \rightarrow Q}{Q}$$

(2) Chain Rule $P \rightarrow Q$ and $Q \rightarrow R$
$P \rightarrow R$

(3) Simplification From $P \wedge Q$, infer P

(4) Conjuction From P and Q, infer $P \wedge Q$

(5) Transposition From $P \rightarrow Q$, infer $\sim Q \rightarrow \sim P$.

→ Monkey-Banana Problem
famous AI problem

CLASSTIME | Page No.
Date / /

→ First order predicate Logic (FOPL)

building new things with help of facts.

(1) PL is not sufficient for connecting multiples sentences

(2) PL doesn't represent quantity. (For all, some)

drawbacks of PL overcomed by FOPL

FOPL offers keywords of quantity

↓

(All, Some)

{ All, Not All } → Quantifiers ← { For some, For Not Some }

(To represent quantity)

∀ (For all) | ∃ (Existencial quantifier)

* Syntax of FOPL - (1) Connectors (same in PL)

∧

∨

¬

→

↔

(2) Quantifiers

Universal quantifier

Existential quantifier (For some)

(3) Constants → use of small letters / letters + numbers

eg- 101, John, a123

(4) variables → denoted with help of words and capital letters are used

(5) Functions - Symbols that denotes the relationship defined on a Domain 'D'.

eg- All employees of company are AI programmers.

(6) Predicates - Predicate symbols are used to denote relations or functional mapping from the element of a domain D to the values True / False. Capital letters or capital words are used to represent predicates.

→ Constants, variables and functions are referred as "terms" and predicates are referred as "atomic formulas".

CLASSTIME | Page No.
Date / /

eg:- (1) All employees earning $1400 or more per year pay taxes.

Entities → employees, taxes } All employees → Taxes

Quantifier → All

raw form.

(2) Some employees are sick today. → Well formed formula (WFF).

$\exists x (E(x)) \rightarrow S(x)$

(3) NO employee earns more than the President.

$\forall x ((E(x) \& GE(i(x), 1400)) \rightarrow T(x))$

x is an employee (all employees)

both cond. must be satisfied.

i(x) → income

1400 → value to be compared.

i(x), 1400 → represents income of employee greater than 1400.

(1) $\forall xy ((E(x) \& P(y)) \rightarrow GE(i(x), i(y))$

can't use same variable for employee & president.

all employees

president

i(y) → salary.

GE (i(x), i(y))

We need to define a function where income of employee < income of president

28 Jan, 19.

First order predicate logic (FOPL)

↓

Predicate

↓

Relationships.

complex sentences representation is possible using connectors.

(Wff)
Well formed formula

properly connecting sentences using connectors

Semantics - (meaning)

↳ Using truth Tables, if T comes we say that Wff is defined semantically.

→ Given Wff ↓

$\forall x \; P(x) \rightarrow Q(x)$.

Domain is needed to define quantity.

W/o domain, interpretation is not possible as quantity is not known.

- Formal System - A formal system is a set of axioms S (proven facts / atomic sentences) and set of inference rules L which can be represented as ↓

$\langle S, L \rangle$ // also known as Knowledge Base

This representation is also known as knowledge base for a particular system. (files made in practical)

- Soundness - The inference procedures L is sound if and only if any statements (S) can be derived from the formal system as a logical consequence. (result)
- Completeness - The inference produre L is complete if and only if any sentence's' can be logically implied using the procedures defined in a formal system.

* Properties of FOPL :-

equivalence rules.
$$P \rightarrow Q = \sim P \vee Q$$
$$P \leftrightarrow Q = (\sim P \vee Q) \wedge (\sim Q \vee P)$$

(all other rules are same as that of PL)

Imp. Conversion to the clausal form. (representation of symbols only, no inference rules, keywords are included)

Step: Clause

A clause is defined as the disjunction of the no. of literals (predicates).

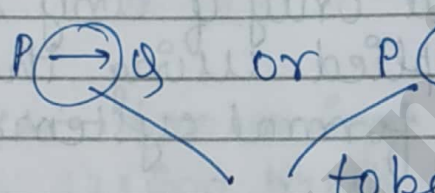
Ground Clause - A ground clause is one that

contains no variables in the given expression.

Horn clause - It is defined as a clause that contains almost one positive literal.

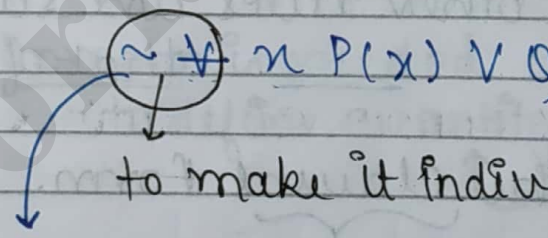
- Steps to convert to clausal form

Step-1 - Eliminate all implications and equivalence symbols.



To remove them, we can use their implied form.

Step-2 - Move negation symbols into individual atoms.



Step-3 - Rename variables if necessary so that all the remaining quantifiers have diff. variable signs.

CLASSTIME | Page No.
Date / /

$\forall x$ } creates ambiguity
$\exists x$ }

∴ Rename it as
$\forall x$ and $\exists y$.

Step-4- Replace existencial quantifier / quantified variables with special functions and eliminate the corresponding quantifiers.

↓

Skolermization - process to remove existencial quantifiers using functions.

Step-5- Drop all universal quantifiers and put the remaining expressions into CNF form.

↓

Confugation Normal form.

conjunction

Step-6- Drop all ~~confugation~~ symbols. writing each clause previously connected by a connective on a seperate line.

- To eliminate existencial quantifiers :- (Step 4).

- Few variables / special functions are used. This process is known as "skolermization".

1st

(1) If the leftmost quantifier in an expression is an existencial quantifier, then replace all the occurences of the variable it quantifies with a constant not appearing anywhere in the expression and delete the quantifier then.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

② For each existential quantifier that is preceeded by one or more universal quantifier, then replace all occurences of the existentially quantified variable by a func. symbol not appearing elsewhere in the expression. The arguments assigned to that function should match to all the variables appearing in each universal quantifier which exists before the existential quantifier. Then delete the corresponding existential quantifier.

$\forall(x)\ \exists(y)$

Here only 1 existential quantifier is present, thus the func. symbol which will be made will have 1 parameter, as only 1 existential quantifier is present.

5-8 marks.

Q Given expression :-

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

① Remove $\rightarrow$ or $\Leftrightarrow$ symbol.

$$P \rightarrow Q = \sim P \vee Q$$

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

negation is only inside brackets as ( ) denotes dependent values and thus

$\exists x \forall y$ is independent of whole expression.

② Remove negation and make them into individual atoms.

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

$\sim \forall z$ becomes $\exists z$ and negative sign shifts to P.

③ Checking for ambiguity & renaming variables if necessary.

If it had been $\exists x \forall y (\exists x \sim P(f(x), y, z)$

- x won't be replaced here as it is not dependent on any value inside ( ).
- changing to f(s)

Same as of ②

④ Implementing Skolermization-

LHS has $\exists x$ but also $\exists z$ is preceeded by $\forall y$

- $\exists x$ → Step ①.
- Both steps will be carried.
- $\exists z$ → Step ②

$\underset{①}{\exists x} \; \underset{②}{\forall y (\exists z)}$ → func. parameter & 1 value

CLASSTIME | Page No.
Date / /

→ Removal of ∃x

$\forall y\ \exists z \sim P(f(a), y, z) \vee (\exists u\, Q(a,u) \wedge \exists v\, R(y,v))$

replacing x by constant a which must not be available throughout exp.

→ Now on leftmost we have ∀y

↓

Now we remove $\exists z$

↓

no. of arguments = 1 as ∀ precedes it.

$\forall y\ (\sim P(f(a), y, g(y)) \vee (Q(a, h(y)) \wedge R(y, l(y))$

name of func. can be made using small symbols but shouldn't occur in exp.

parameter y as universal has 'y'

$\forall x \forall y\ \exists z$

↓

$g(y, x)$ → move from RHS to LHS.

Combination of 3 steps.
(i) remove $\exists z$
(ii) remove $\exists u$
(iii) remove $\exists v$

CLASSTIME | Page No.
Date / /

(5) Remove ∀ quantifier

$(\sim P(f(a)), y, g(y)) \vee (Q(a, h(y)) \& R(y, l(y))$.

If by default, no quantifier is given, always consider ∀ and not ∃X.

$(P) \vee (Q \wedge R)$

We can apply distributive law.

$(P \vee Q) \wedge (P \vee R)$ ⟶ CNF form

↳ remove ∧

$P \vee Q$ and $Q \vee R$ ⟶ $P \vee Q$, $Q \vee R$

Final form.

4 Feb, 19

CNF, DNF and PNF

- $C_1 \vee C_2 \vee C_3 \vee C_4 \longrightarrow$ DNF (Disjunction)
- $C_1 \wedge C_2 \wedge C_3 \wedge C_4 \longrightarrow$ CNF (Conjunction)

C_1, C_2, C_3, C_4 clauses.

(can be converted using laws or inference rules only).

$\longrightarrow$ PNF
(prenex normal form)

– It is the form in which all quantifies are present at the L.H.S of the clause.

* Steps to convert to PNF :-

(1) Eliminate all occurences of implication or double implication from given formula.

(2) Move all negations to inward such that in the end negations will only appear as part of literals.

(3) Rename the variables if necessary.

(4) The PNF can now be formed or obtained by moving all quantifiers to front of the formula.

eg:- Convert into PNF :-

$\forall x (\exists y\ R(x,y) \wedge \forall y \sim S(x,y)) \longrightarrow \sim(\exists y\ R(x,y))$ (P)

CLASSTIME | Page No:
Date / /

① Remove implications using $P \rightarrow Q \equiv \sim P \vee Q$

$\forall x (\sim \exists y\ R(x,y) \wedge \forall y \sim S(x,y)) \vee \sim (\exists y\ R(x,y) \wedge P)$

② Move negations to literals.

$\forall x (\forall y \sim R(x,y) \vee \exists y\ S(x,y)) \vee \forall y \sim R(x,y) \vee \sim P$

③ Rename the variables.
to remove ambiguity.

$\{ \sim \exists y = \forall y \}$
$\{ \sim \forall y = \exists y \}$

$\forall x \forall y_1 \sim R(x, y_1) \vee \exists y_2\ S(x_1, y_2) \vee \forall y_3 \sim R(x_1, y_3) \vee \sim P$

④ moving all quantifiers to front of the formula.

$\forall x \forall y \exists y_2 \forall y_3 (\sim R(x,y_1) \vee S(x,y_2) \vee \sim R(x,y_3) \vee \sim P)$

NOTE:- Representing expression using quantifiers, variables.

{ constants → a, b, c
Variables → x, y, z }

↓

formula

- Unification :- (one / single)

mother (sita, xyz)
mother (def, xyz) } Clauses

Def (Any substitute that makes 2 or more expressions equal is called unifier for expression)

converting 2 clauses to a single form using either substitution or replacement is unification.

↳ if replacing a variable replace it everywhere.

For the above example, if we substitute mother(def | sita, xyz).

Unify ↳ term → (any predicate)
variable → (x, y, z)

NOTE:- 2 given clauses need not to be unified everytime.

may or may not be.

- Unification - process through which we are unifying 2 clauses through substitution or replacement.
- Resolution :- (process)

↓

used to build check validity of a clause obtained by some other clauses.

↓

Prove by contradiction

add negation of some clause and try to prove that

if proved → contradiction (hence not valid).

In exam (Given) { 5 General Statements ↓ convert into WFF ↓ then clausal form ↓ then Resolution. }

→ Resolution principle is a syntactic inference procedure in which a set of clauses or axioms are added to the information which we need to proove. (negation of info.)

OR

→ After applying the resolution, if we are getting a unsatisfiable set or empty set, then this process is known as "Refutation".

* There are 4 types of Resolution :-

eg:- { ~PVQ , ~QVR } 2 given clauses.

new clause obtained is resolvent

If a literal present in normal & one in negation, then only resolvent can be formed.

resolvent → ~PVR ~PVQVR

remove this literal to form resolvent ~PV~QVR

(literals can be as well as less)

P, Q, R ?

CLASSTIME | Page No.
Date / /

(i) Type 1

Binary Resolution

dealing with 2 clauses.

$\sim P(x, a) \vee Q(x)$
$\sim Q(b) \vee R(x)$

Check for literal present in both form

↓

Yes, Q
but Q(x) and Q(b)

use unification, → term / variable.

$\sim P(x, a) \vee R(x)$

∴ b/x not x/b
X

substitute Q(b/x)

(ii) Type 2

Unit Resulting Resolution
{UR}

- more than 2 clauses. {multiple clauses}

$\sim$ married(x, y) $\vee \sim$ mother(x, z) $\vee$ father(y, z)

married(sue, joe)

$\sim$ father(joe, bill)

↓

3 clauses

Here, { married & father } is present in both form.

~ married (x sue/x, Joe/y)
↓ skip this step {~P VP = 1}.
married (Sue, Joe)

→ gets cancelled.

Replace all x by sue/x and y by y/Joe, z by Bill/z.

Final Clause → ~ mother (sue/x, Bill/z).

(3) Linear Resolution :- (Type-3)
When each resolved clause C_i is a parent to the clause C_{i+1}, this process is known as linear resolution.

$C_i \rightarrow C_1 \rightarrow$ substitute D_0 in C_1 to obtain C_2
$C_{i+1} \rightarrow C_2$

∴ New clause is obtained by substituting in previous clause.
(getting clauses in sequence)
↓
i.e $C_1 \rightarrow C_2 \rightarrow C_3$ .....

(4) Linear Input Resolution (Type-4)

If one of the parents in linear resolution is always from the orginal set of clauses, then we have implemented linear input resolution.

{Peterson - 4, 3, 6, 7}

7 Feb, 19

CLASSTIME | Page No.
Date / /

connection b/w entities

Ch-7

Semantic Network & Frames.

no standard relationship.

↓

enhancement of knowledge representation

providing info. to agent through which it draws out new inferences & acts on environment.

PL → Semantic N/w.

FOPL.

defining relationship with help of objects.

We need to define as many times as there are no. of same obj. in relationship.

father (ram, xyz);
father (ram, def);
father (ram, pqr).

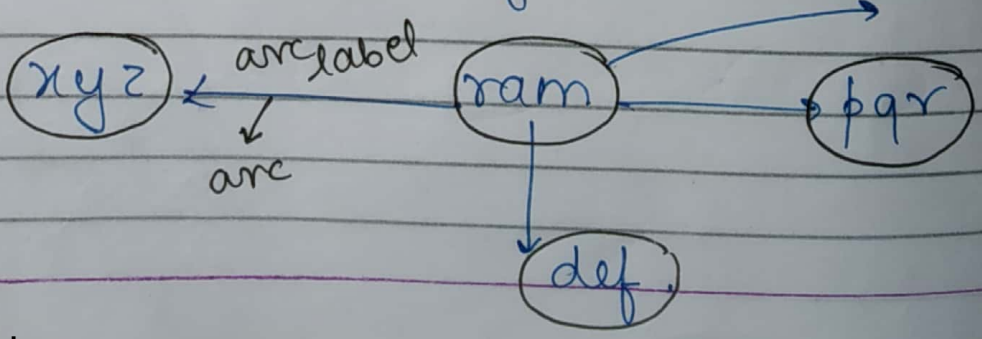
common object.

↓

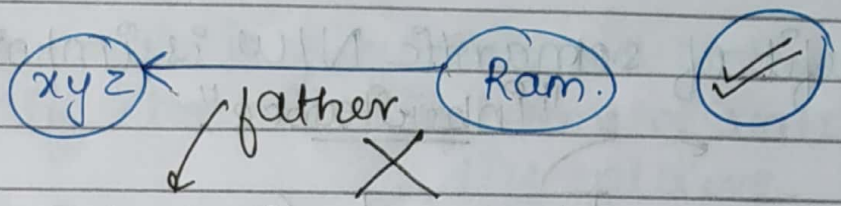
To overcome this problem, we have Semantic N/w

combining multiple entities into a single meaningful unit.

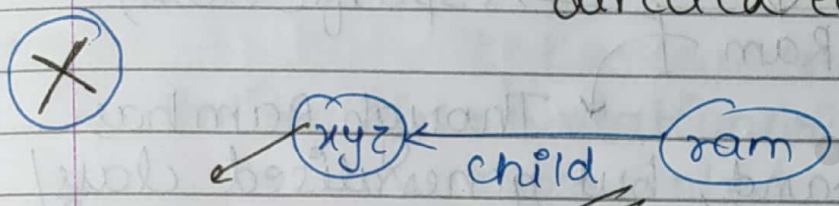
or



We are able to combine common entity b/w multiple objects.



We need to define arc label as the last object at end of arrow as arcs are directed edges.



nodes are objects/ entities

Relationship is arc label.

We can define diff. relationships using a common object

mapping b/w 2 objects.

- Semantic N/w is also known as "Associative N/w".

association b/w 2 entities.

eg- (Book) Tweety is a bird of yellow color
Design it in terms of semantic N/w.

(1) Identify Entities.
(2) Identify arc labels.

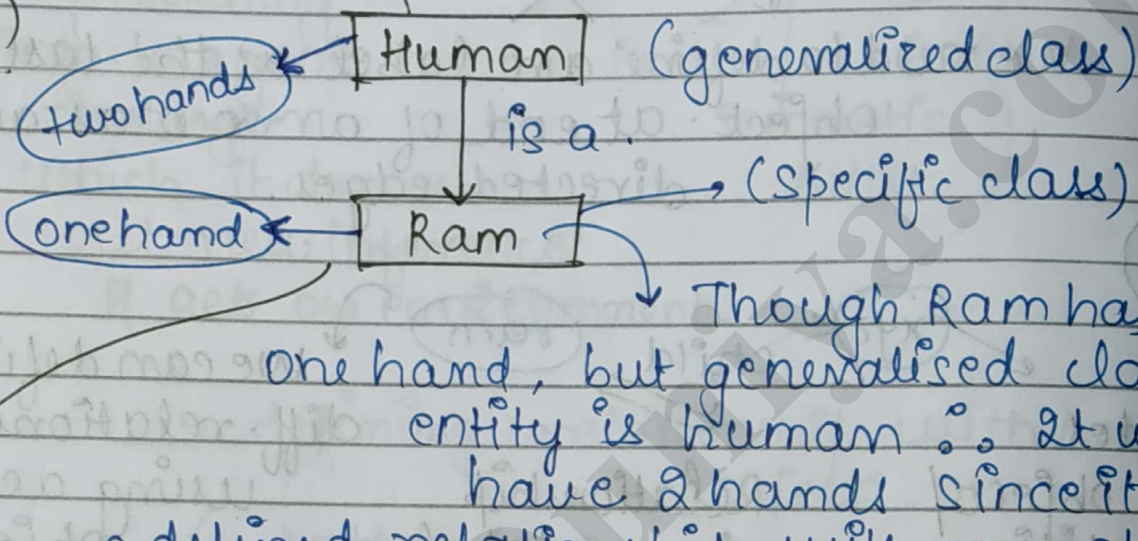
has some particular chrs./ behaviour.

Entities -

→ Always take a constant / real word attr. as entity.

→ Benefit of semantic N/w is importance of "Inheritance".

(central entity)

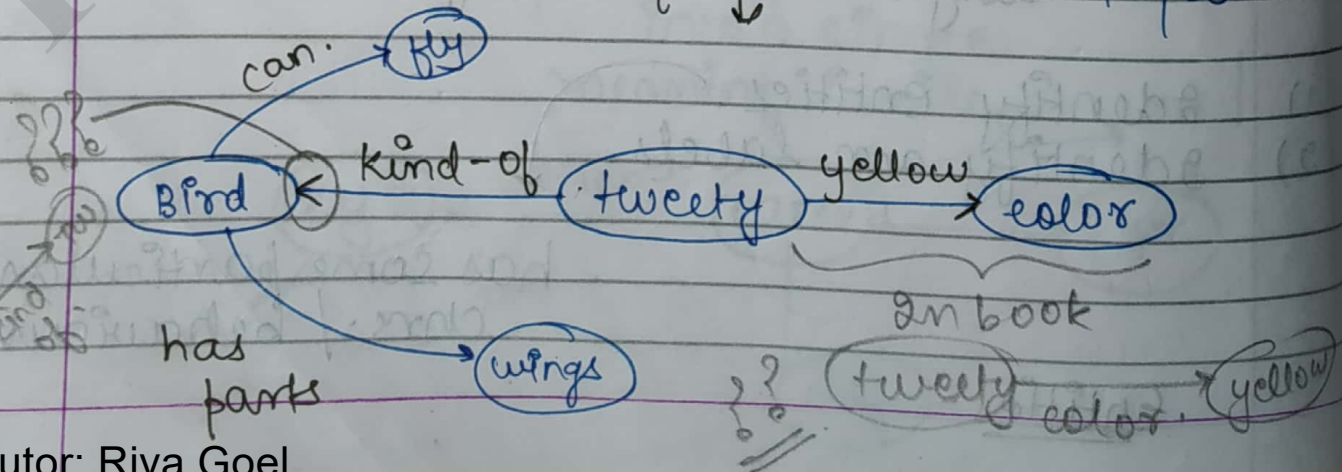


Though Ram has one hand, but generalised class/ entity is human ∴ it will have 2 hands since it has a defined relationship with some other entity.

can't restrict an entity to a particular characteristic if it is linked to some other entity.

→ Representation is not always a down arrow (↓). Semantic N/w is just mapping. (↑) is also possible in semantic N/w.

For the above example,

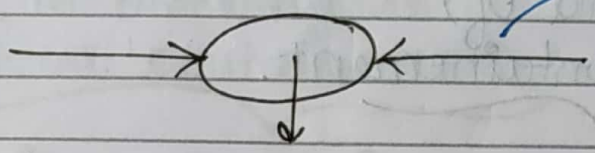


Ch-7 & 9 → Ex. 2mp.

CLASSTIME | Page No.
Date / /

→ Here, we can also represent some hidden info. like can fly, has wings.

Color → It is an entity as it has some diff chrs. i.e in terms of values i.e diff colours.



(central entity derives things)

→ If arrows converge to common entity, then relationship will never be represented.
From central entity, we always draw out the arrows.

eg (in book) Ex 7.1 {Read}.

→ Conceptual Graphs.
- type of representation scheme.

• Types of Relationships available.

(1) Generic-generic (5 parts)
→ Superset & Subset.
→ Subset & Superset

eg - fighting-ships - battle ships.
Subset . superset.

This can also be a superclass / superdomain of some other entity.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

Agni — fighting ship — battle ship.

fighting ship has other entities.

→ AKO (A Kind Of)
→ Conceptual containement.

One automatically included in other.
eg - figure circle inheritance.

→ Role Value Restrictions.
eg - an elephant trunk is a cylinder 1-3m in length.

defining a role of entity with its restriction

role of trunk is carrying weight but restriction is on length of 1-3m.
defines max. limit.

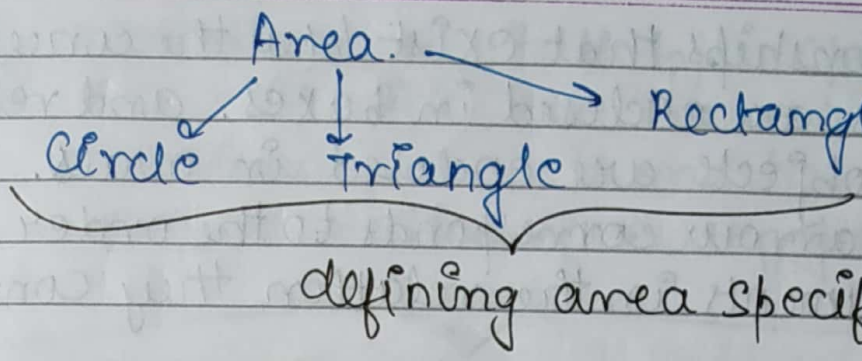
Cylinder is a characteristic and not a restriction.

(2) Generic - Individual relationships

→ Abstraction.
- Things declared as abstract are compulsory to provide implementation.
- In next class, we define the specific implementation of that generic method.

Specific terms.

(in programming terms)



defining area specifically.

→ Set membership.

→ Conceptual containement.

entity has its own values and also it will inherit all characteristics of its superclass.

Human
↓
Ram

generic to specific.

Two hands
↓
One hand

generic to generic.

→ Conceptual Graphs -

Semantic N/w is a directed one. Thus it can't traverse back from leaf node.

This disadvantage is overcome by conceptual graphs.

- A conceptual graph is a graphical representation of mental perception which consists of basic or primitive concepts & the

relationships that exist b/w the concepts.

- Concepts are enclosed in boxes. and relations b/w the objects are enclosed in ovals. The direction of the arrow corresponds to the order of the arguments in the relation they connect.

- Concept symbols referred to entities, actions, properties or events in real world.

- A concept may be individual or generic.

eg:- Joe is eating soup with spoon.

NOTE:- Individual concepts have a type field followed by a referent field. (Generic version)

- First concept — Joe. (individual concept).

Generic version [Referent version of Joe = Person.

- 2nd — soup (individual)
- 3rd concept - spoon.

generic - food. → specific field (generic - utensil).

→ Relations ↓

includes some fixed versions. (eating & using)

If for relations, we talk b/w specific-specific, we have a fixed relationship b/w them, thus knowledge representation of every person will be same.

This is not the case in semantic N/w.

NOTE:- Relations may not be same, differs acc. to person.

Use/using :-

But if relation gets fixed, same info. is derived by 2 diff. agents
∴ reversing order of reading in-order remains same.

Person : Joe

Generic to specific.

Food : Soup

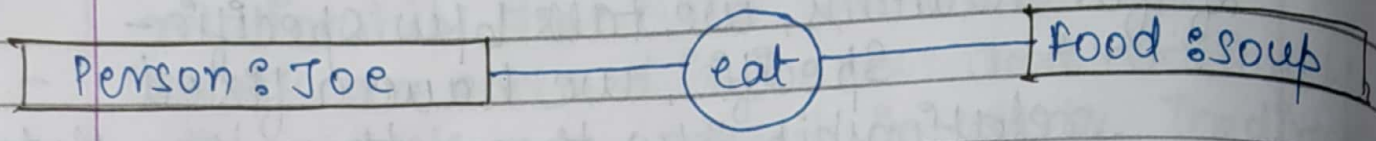
Utensil : Spoon

to define referrents on the LHS.
from where it is referred
(Generic term)

can be considered as a generic term as spoon has diff. types (same case with soup)

→ But Joe is specific, & thus person (=referrent) can't be omitted.

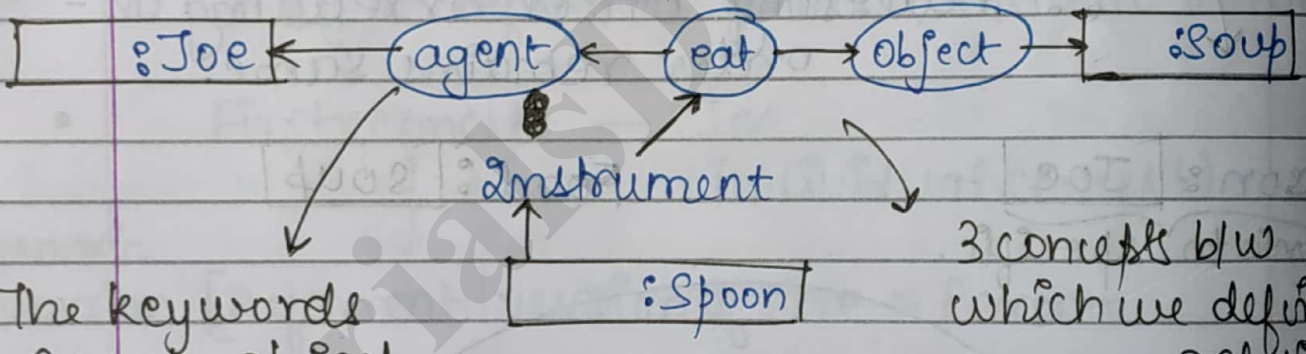
types of entities is possible.



no complete info. only with this relation "eat".

Utensil : Spoon

In conceptual graphs, instead of arc labels we have fixed relation / term depicted by ( ).



The keywords agent, object, instrument are placed depending upon entity.

3 concepts b/w which we define relations.

Agent → used with generic & relation.

NOTE:- Here, arc label will be same for all agents as relation name are fixed.

CLASSTIME | Page No.
Date / /

boxes → concepts.
circles → relationships.
ovals → are labels / mappings.

- frames – same info. in terms of boxes.

1 frame
car (superframe).
engine
brakes
clutch
characteristics.

2nd frame → engine characteristics.

- Scripts :- {dialouges in terms of concepts and relations}.

{Ch 7 + Ch 4 (Back Ex.)} → 20 marks

CLASSTIME | Page No.
Date / /

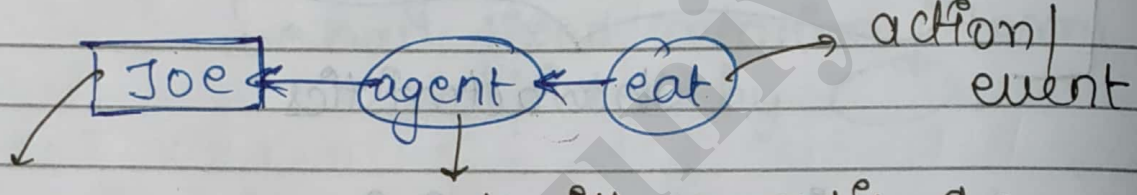
4 Feb, 19

Conceptual Graphs

consists of concept and relation nodes.

Concept nodes represent entities, attributes, states, events

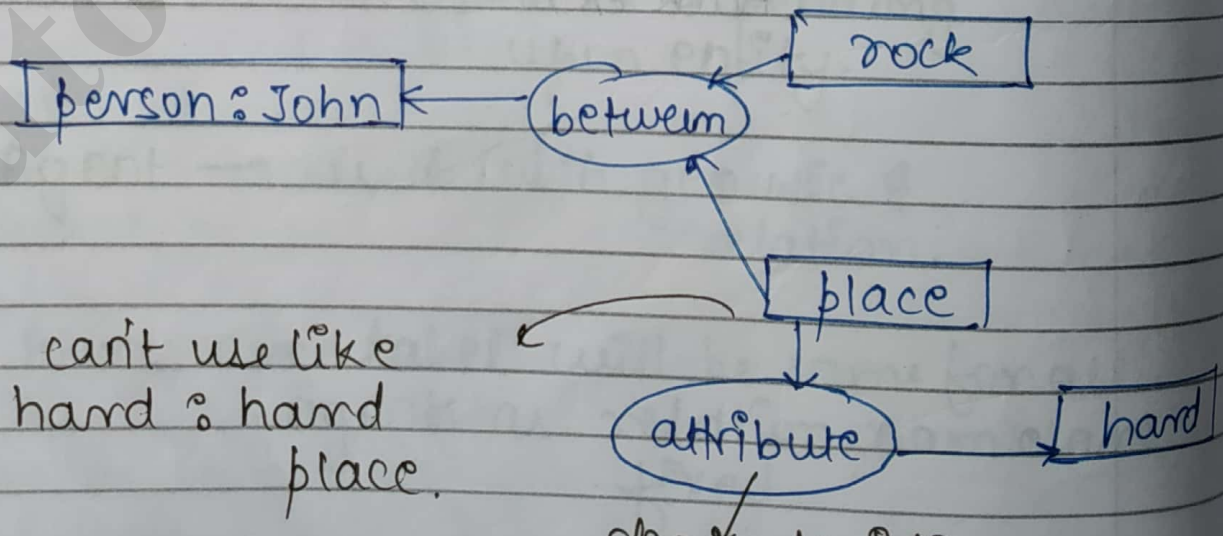
Relation nodes show how concepts are interconnected.



Q Draw conceptual graph for :-

(1) John is b/w a rock & a hard place
(2) John is going to Boston by bus.

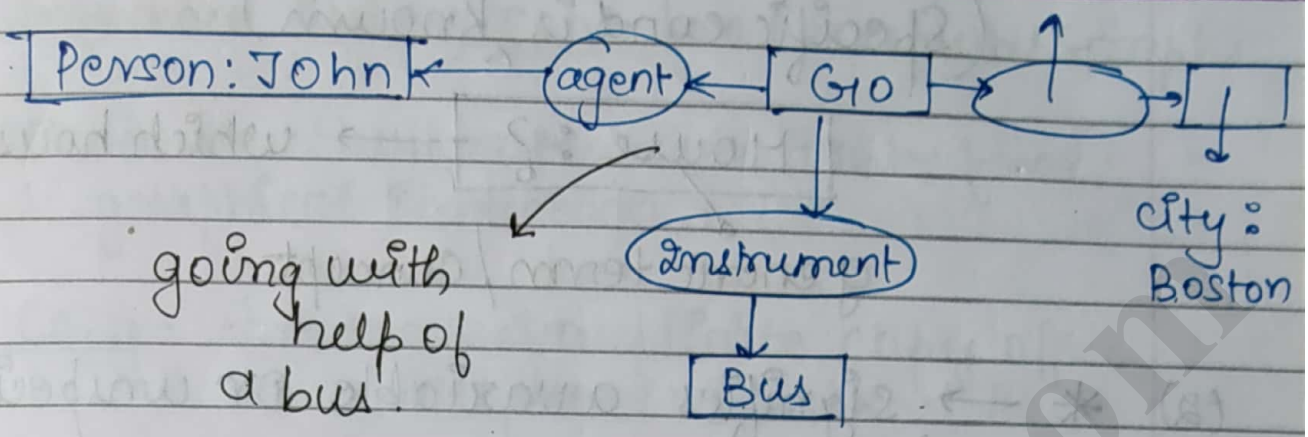
(1)



We can directly use [Hard] but it is wrong interpretation in graph.

CLASSTIME | Page No.
Date

(2)



→ The operator is testing earth resistance using a megger

Person: Operator → Agent → Test → Object → Parameter: Earth Resistance

Instrument

Concepts

person Test megger Earth Resistance.

megger

imp: ~~(Table of Agent, Object)~~

- Other symbols used in conceptual graphs. (on RHS of colon)

(1) ? → which

Ram is going to a house.

own house

someone's house.

(Specific cond is known)

House : ? → which house

generic term / concept.

(2) * → signifies a variable or unspecified house.

eg:- some house

House : *

eg:- on house → specific concept / referent field is known
∴ no unknown quantity, no need to use special symbols

House : on

(3) ∀ (universal quantifier) – used for "forall"

(4) @ – it is used for quantity

(5) # – used to refer a specific value.

eg- house no. 226.

House : #226 ✓

House : 226 ✗

- Inference Rules / cond. for Conceptual Graphs

→ There are 4 operators available for graphical Inference :-

(1) Copy - produces a duplicate copy of conceptual Graph.

(2) Restrict - It modifies a graph by replacing a type label of a concept with a subtype or a specialisation from generic to individual by inserting a referent of the same concept type. / content type.

eg :- Figure
↓
Rectangle. → drawing a new inference (Rectangle is type of figure)

(3) Join - It combines 2 identical graphs i.e CG_1 and CG_2 by attaching all relation arcs from CG_2 to CG_1 and then erasing CG_2.

(4) Simplify - It eliminates one of 2 identical relations in a conceptual Graph when all connecting arcs are of the same name / connecting arcs are also the same.

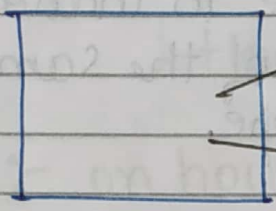
↓
works on relations, whereas in join we merge whole of the graph.

Knowledge Representation

* Frame Structures -

- **Frames** - - DS that captures the implicit connections of info. from explicitly organised DS.

- used to represent info. in slots & values.
- Giving a name to a particular frame.



Slots (chrs. of frame)
-name, rno, address, sem...

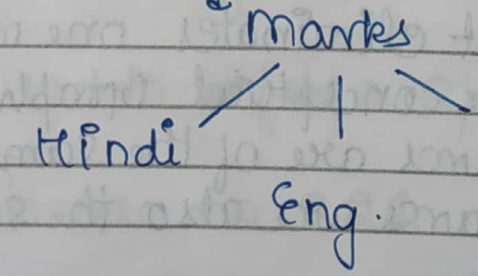
Values
- Riya, 914, Sri Nagar, 6thsem

Student
(Frame)

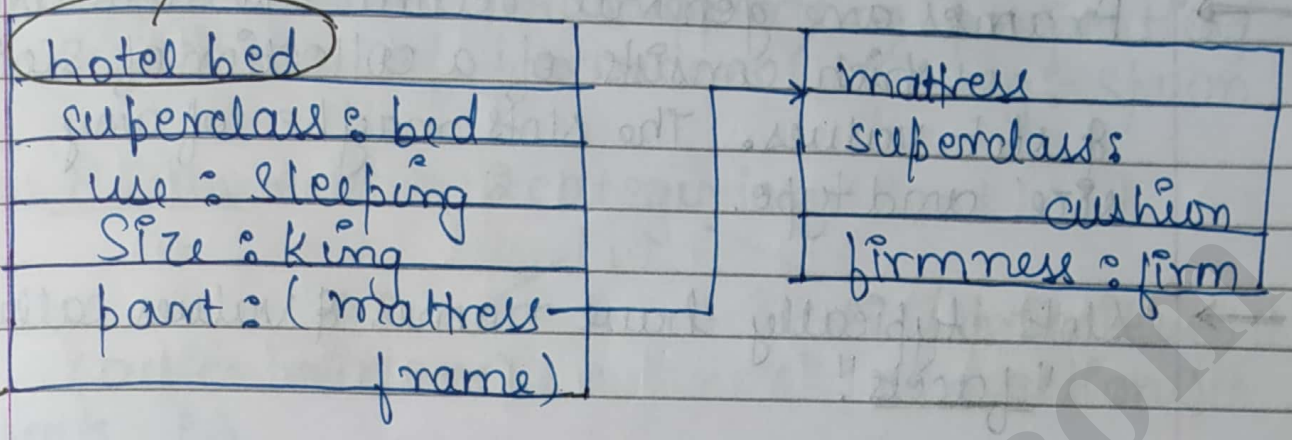
Default values can also be given to attributes.

values corresponding to each attribute

NOTE:- We can define a subframe of a frame.



giving name to a frame



pair of slot & values.

superclass : bed

slot value.

Q Create a frame named student. (see from book)

First give a name to student.

Student
id = 1
name : Riya
specialization of : person
address : RaniBagh
grade : A
contains : (marks, course)

Through frames, inheritance is possible

Capital letter (no use of _)

used to define subframes

Course
name : CS
degree : graduation
sem : 6
year : 2019

CLASSTIME | Page No.
Date / /

→ Frames are general record like structures which consists of a collection of slots & slot values. The slots maybe of any size and type.

→ Slots typically have names & values called "facets".

* Scripts & Conceptual Dependancy :-

aim is to standardize things.

moving towards standarization using scripts & new names i.e talking of physical entities.

* 5 parts to represent sentences into structures

□, ○ X → X
use ⇒

→ There are 5 diff. types of components that are ontological building blocks.

(state of being), next step of AI.

(1) Entities (in standard terms)

PA (picture aiders)

PP (Picture producers)

- Picture producers (PP) are actors or physical objects. eg :- Joe.

CLASSTIME | Page No.
Date / /

- Picture aiders are supporting properties/ attributes of producers (PP). eg:- Spoon.

(2) Actions - 2 categories

AA (action aiders) - Supports PA.

PA (primitive actions).

(3) Conceptual Cases -

Types available are :-

- Objective (o)
- Directive (d)
- Instrumental (i)
- Recipient (r) (written on arrows)

gmp.

Primitive Acts of CD Theory :- (PA)

(1) ATRANS - Transfer of an abstract relation- Ship (i.e give). (location not changed)

(2) PTRANS - Transfer of physical location of an object (i.e go). (location changes)

(3) PROPEL - Application of physical force to an object (eg :- push)

(4) MOVE - movement of a body part by its owner (i.e kick)

(5) GRASP - Grasping of an object by an action (eg - throw)

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

(6) INGEST - Ingesting of an object by an animal (eg - eat)

(7) EXPEL - Expulsion of something from body of an animal (eg - cry) (though changes)

(8) MTRANS - Transfer of mental info. (eg tell)

(9) MBUILD - Building new info. out of old (eg - decide)

(10) SPEAK - Production of sounds (eg - say)

(11) ATTEND - Focussing of sense organ towards a stimulus (eg - listen)

(4) Conceptual Tenses -

↓ talking in present, past or future.

Tenses available are.

→ Conditional (c)
→ Continuing (k)
→ future (f)
→ Past (p)
→ Present (nil) → no letter for present available
→ Transition (t)
→ Interrogative (?) → can't use 'i' as 'i' is used in conceptual case (Instrumental)

→ Negative (\)
→ Timeless (delta)
→ Start Transition (ts)
→ Finish Transition (tf).

imp.
(5) Conceptual Dependency – whole str. is based on this.

- has fixed rules.
- how is one concept dependent on another.

Rule1 :- PP ⟺ ACT — action & entity are directly related.

eg - John ran
under PP — action under PTRANS.

John ⟺ (P) PTRANS. — physical Transfer.

denotes past tense.

John ⟺ PTRANS

use of double arrow.

nothing mentioned :. tense is present.

Rule 2 :- ACT ← PP

eg - John pushed the bike.

action of push.

Conceptual objective

John ⟺ PROPEL ← bike.

John pushed. (under Rule 1)

thing to be pushed.

Rule 2 says a PP is related / contributes an action

ACT ← PP

Rule 3 :- PP ↔ PP (both are entities)

generic-specific / specific-generic.

eg- John is doctor.

CD Rep :- John ⟷ doctor.

blank i.e present tense.

no use of ⟺

diff. from ⟷

Rule 4 :- PP ← PP

not generic-specific / specific generic

both entities are diff. & not dependent

eg - John's dog.

(entities will transfer & not ←)

John ← dog ✗ for dog's John

pass-by

dog ← John

Rule 5 :- PP ⟺ PA

eg - John is fat

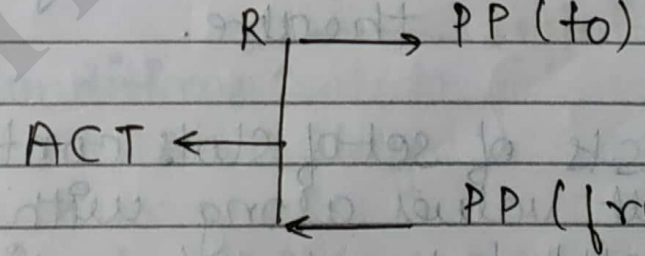
CD Rep - John ⟺ weight (>80).

NOTE :- For words like height, tall thin, for these values we also consider a reference value.

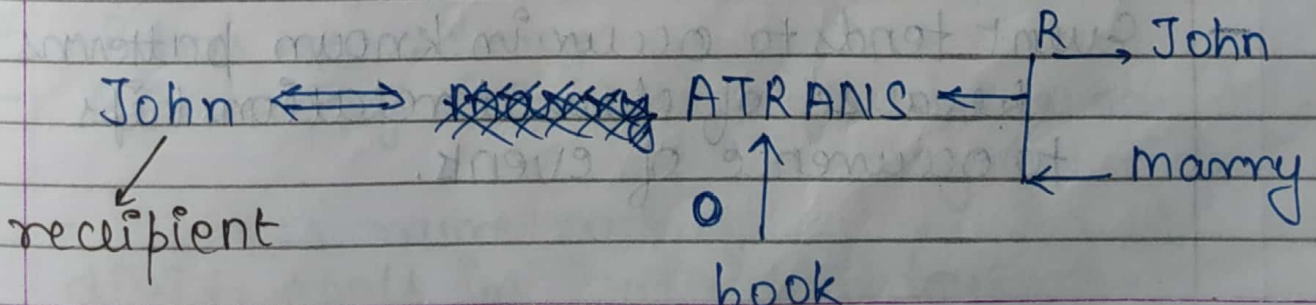
i.e >80 fat person
<40 thin person.
>5' tall person.

V.V. gmp.

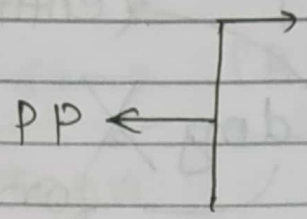
Rule 7 :-



eg - John took the book from marry.



Rule 8 :-



- Script Structures :-

- Sequence of dialouges. represented by ED
- useful in describing certain stereotyped situations such as going to theatre.
- It consists of set of slots containing default values along with some info. about type of values similar to frames.
- Event tends to occur in known patterns because of causal relationships to occurence of events.

CLASSTIME | Page No.
Date / /

* Script Components -

(1) Entry conditions. - Initial context / goal.

(2) Results

(3) Props. - properties.

(4) Track - script name.

eg- Script : Play in Theatre.

- Track : Play in Theatre.
- Props : Tickets, seats, play.
- Roles :
  - person (who wants to see a play) - P
  - Ticket Distributor - TD.
  - Ticket checker - TC

Definition of goal.

- Entry conditions :
  - P wants to see a play
  - P has money.

(Here we have scenes : scene1, - - - -)

- Results :
  - P saw a play
  - P has less money.
  - P is happy (Optional if he liked the play).
  - TD has more money.

divide goals in Individual tasks.

Scene 1: Going to theatre.
P PTRANS P into theatre.

Scene 2: Buying tickets.

21Feb19

Scripts

conceptual dependancy are needed to define a script.

- Script components
(eg. from book)

fixed format

Script Name: Food market } same
Track: Super market }
Role: main entities.
- shopper
- attendant
- checkout clerk (scans items & make bills)
- other shoppers

Entry conditions: shopper needs groceries } 2 cond.
food market open.

Props: (properties/attributes upon which entities (work))
- Shopping cart
- display
- cashier
- items
- money → Put in box

{Track is specific version of script Name}

CLASSTIME | Page No:
Date / /

NOTE:- Roles has those entities who produce a dialouge in script.

NOTE:- entry cond. ⟶ starting base cond.

eg- for a restraurant
↳ entry cond = restraurant open person is hungry.

* Next step is defining dialouges {independent tasks make 1 scene}.

Scene 1 : Enter market.
Shopper PTRANS Shopper into market
Shopper PTRANS shopping cart to shopper.

grnp. Scene has to defined in terms of conceptual dependencies to standardize our problem.

Scene 2 : Shop for items
Shopper MOVE Shopper through lanes.
Shopper ATTEND eyes to display items → (used for move)
shopper PTRANS items to shopping cart.

Scene 3 : check out
Shopper MOVE Shopper to checkout
Shopper WAIT Shopper Turn
Shopper ATTEND eyes to prices/changes.
Shopper ATRANS money to cashier
(abstract Transfer)
attendant ATRANS bags to shopper.

Shopper gives money to cashier.

(Here we have PP only)

NOTE:- Shopper PTRANS Shopper.

Here, we used shopper 2 times to show conceptual Dependency

PP ⟺ ACT.

Here we needed to define an action for producer, thus we used shopper.

Shopper ATTEND eyes.

Both PP & PA are present :. Shopper twice won't be used.

Scene 4: Exit market.

Shopper PTRANS Shopper to exit

Results: Shopper has less money.
cashier/ market has more money
market has less items.

NOTE:- Shopper PTRANS Items to Shopping cart

actual inference is customer needs that item.

SAM → Script Analyzer mechanism.

NOTE:- main aim of KR is inferencing new info.

Ch-4

4.8 Non Deductive Inferences

not deducible

→ **Abductive inference :-**

P
P → Q } Q is true

Deductive inference (modes ponens).

no complete knowledge upon which we can conclude things.

Diff. b/w modesponens & abductive inference

→ We adda symbol 'c' where c represents a possible clausal relationship.
(assumption)

P
P(c) → Q

→ **Inductive inference :-**

$P(a_1), a_2 \ldots$ satisfy conditions

Induced info.

$$\frac{P(a_1)\ P(a_2) \cdots\cdots\cdots P(a_k)}{\forall x\ P(x)}$$

Forall cases P(x) is True

Conclusion

NOTE:- Inductive inferences can be false for a particular case.

eg:- Pigeon is a bird → Then we can assume that all birds can fly.
But if we see that ostrich is a bird but it can't fly.

→ Analogical inference :-

Given 2 cond., if structure matches in terms of cond., then we say that we can assume stmts. 1 & 2 are True/related.

4.9 Representation Using Rules

If, then, else

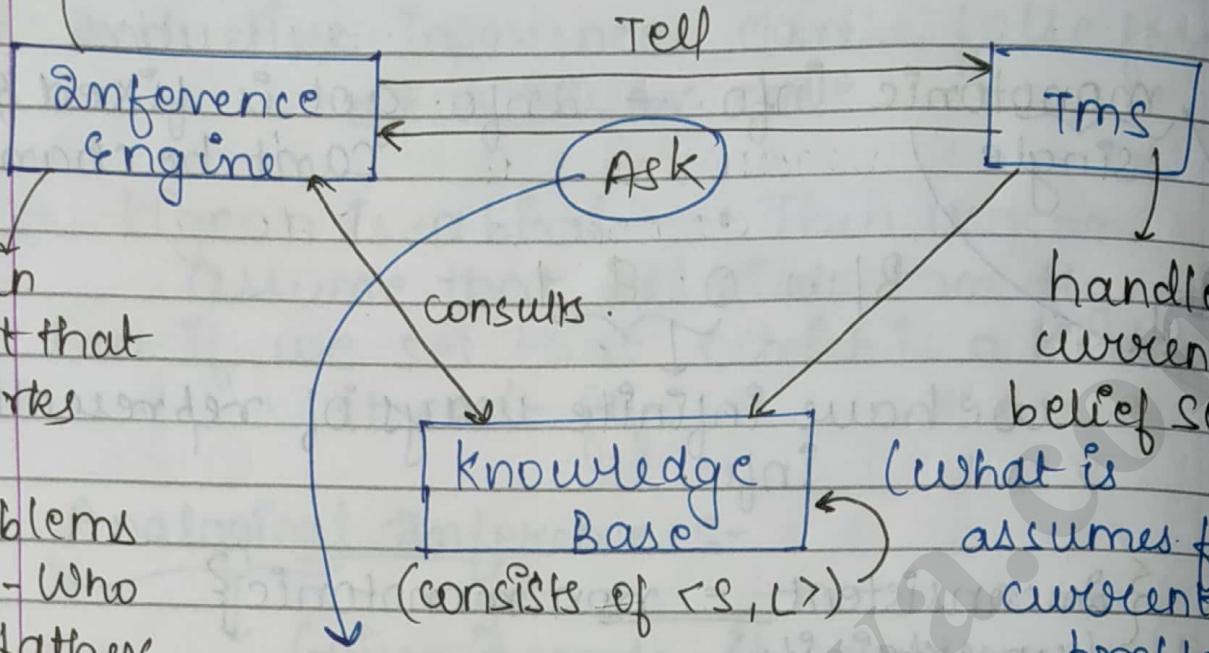
Ch-5

- monotonic info. — info. got is fixed & can't be changed.
  - single
  - or certainity
  - B/w 0 & 1
    - we have infinite ways of representing info.
  - System will always ↑se its Knowledge Base. (can't decrease).
    - {inconsistent = non monotonic} (uncertainity).
      - system can shrink / glow.
    - To handle all uncertainities we have a system
      - TMS (Truth maintain system)
        - for a new conclusion, it doesn't remove previous conclusions but instead saves a copy of that conclusion for further use.

Block Diagram {imp.}

{network consists of trees / graphs as a cycle is maintained}

problem solver



main part that works on problems (eg - Who is father of John) based on KB & replies back

handles current belief set. (what is assumes for its current problem)

on what basis inference engine has deduced things.
Thus, TMS updates its belief set & Knowledge Base changes.

eg :-

Initially $P \rightarrow Q$

P

$\therefore Q$ True

Now $\sim P$

Inference engine deduces Q is not possible & tells TMS about this.
TMS asks inference engine & if it is satisfied, it changes its current belief set & Knowledge Base gets updated.

Copy of conclusion isn't erased. (stays in memory)

interms whether system remains consistent

used to make system consistent

CLASSTIME | Page No.
Date / /

→ DDB (dependency directed Back Tracking)
(retracking)

conclusion Q is dependent on P & P → Q

NOTE:-

<S, L>
S → set of axioms
L → inference rules.

→ TMS is used to maintain the consistency of the knowledge being used by the problem solver & not to perform any inference functions.

→ The inference Engine solves domain problems based on current belief set. whereas TMS maintains the currently active belief set.

- Record maintainence in form of DDB :- network.
  - The nodes in the network representing Knowledge Base entries such as premises, conclusions and inference rules. are attached and justified with the help of justifications.

- Premises - It is a fundamental belief which is assumed to be always True. can be directly inferenced.

(no supporting statement)

eg - Ostrich can't fly → Ostrich is a bird. belief.

TutorialsDuniya.com

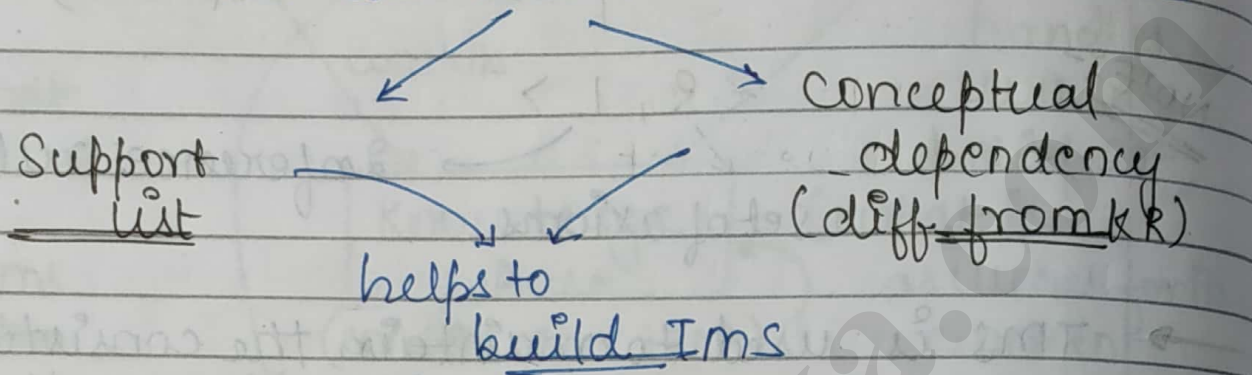
Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

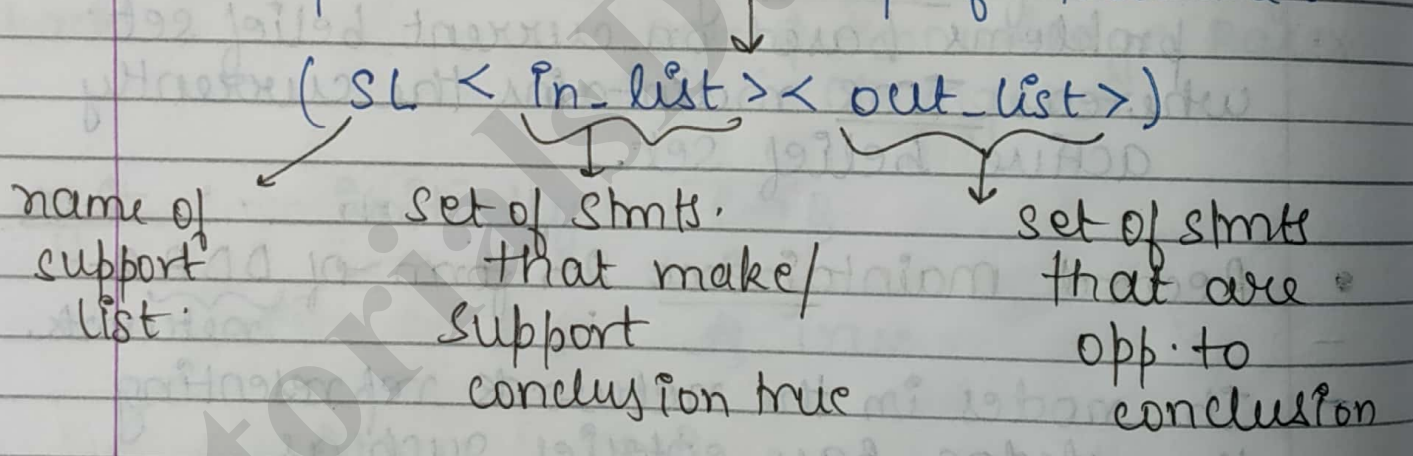
Please Share these Notes with your Friends as well

(supports conclusion)

- Justification - records which are used to prove another statement.
There are 2 types of Justification records which are used to maintain records



* Support list → It is a data structure that contains 2 lists with a name Inlist & Outlist and can be represented with help of structure :-



eg- Sybil is a non-flying bird (Ostrich)

Given statement. (KB)

We are trying to identify multiple conclusions & then will segregate into premises or Justification.

whether it contributes to system.

node	Status	meaning	Supportlist	Comment
n1	IN	Sybil is a bird	(SL () ())	premise
		info. from given stmt.	empty as it is directly infered	
n2	OUT	Sybil can fly	(SL (n1) (n3))	unjustified belief.
		direct as well as indirect info.	supports n2	contradicts n2
n3	IN	Sybil can't fly	(SL (n5) (n4))	justified belief
			opposite.	
n4	OUT	Sybil has wings	(SL () ())	retracted premise. (unjustified belief)
n5	IN	Sybil is an ostrich	(SL () ())	premise.

only 1 can contribute in system

clear info. from given sentence
We don't need to prove it using any supporting node

non-flying → no wings. → no direct inference
But Sybil has wings
↓ derived from
Sybil is a bird. (from inheritance)

This is not an induced info ∴ retracted premise.

CLASSTIME | Page No.
Date / /

→ Now if a new node comes
↓
ostrich can fly.

Work of TMS { System becomes inconsistent upon addition of a new node & thus, we need to redefine str. of support list
i.e changing inlist, outlist & status.

* To represent a network

→ There are 4 components / symbols that are used to represent belief N/w.

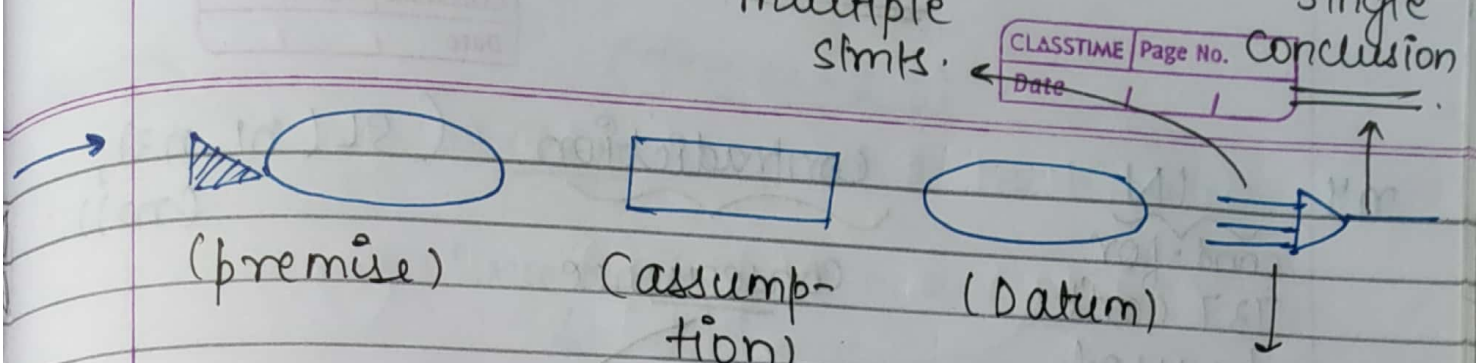
(1) Premise - true proposition requiring no justification.

(2) Assumptions - It is a current belief that could change.

(3) Datum - Datum is either currently assumed or IE derived belief.

(4) Justifications - These are the nodes that supports any datum or assumption to be believed as True or false.

Symbols used are ↓



{See from book}.

{final result will be given by this}

* conceptual dependancy
↓
represented using

(CP <consequent> <inhypo> <outhypo>).

consequence (result) — an support list, conceptual dependencies are never made.

eg- An aircraft will take off only if its crew members are complete
↓

contradicts n1 ↖

n1 IN type(aircraft) = 737 (SL()(n2))

n3 IN class(crew) = A (SL(n8, n9, ...), ()).

n2 OUT type(aircraft) = L400

2nd aircraft doesn't contribute to system

will takeoff if 737 doesn't take off.

1, 2, 4, 5, 7

CLASSTIME | Page No.
Date / /

n4 IN cond. for 737 can't be used. — contradiction. (SL(n1, n3), (n2)).

Consequence → CP(n4(n1, n3), ()).

n5 IN — no good n1 — actual implementation of DDB.

Now n1 gets status OUT & n2 gets IN instead of OUT. as n2 is possible solution for n1

This is DDB → first go to root problem & then try to find some solution for it.

assignment.

Default Reasoning & close world assumptions

common sense reasoning.

If P is available ~P is also "

⇓

Thus if we conclude Q, then ~Q can also be concluded.

25/2/19 (Till Sec 6.3)

CLASSTIME | Page No.
Date / /

Ch-6

Uncertain
↓
prob. of uncertainity.

prob. in real world is of its occurence & not its failure.

Finding probability of an event given an event has already occured.

Bayes Theorem / Cond. probability (dependency).

P(A/B)
- to be found
- given prob.

prob. of A given that B has already occured.

→ Set
State Space - Set where we have all possible values of probability.

- Bayes Theorem :- Bayes Theorem states that for 2 events H & E with a probability P(E) > 0, then conditional probability of event H can be calculated

given that E has already occured

$$P(H|E) = \frac{P(H \& E)}{P(E)} \quad \text{---(1)}.$$

$$P(E|H) = \frac{P(H \& E)}{P(H)} \quad \text{---(2)}$$

For another cond. if we want to calculate prob. of E where H is given.

* Freq Distribution]

$$rf(H \& E) = \frac{No(H \& E)}{n} \quad \text{---(3)}$$

$$rf = \frac{No(E)}{n} \quad \{\text{no. of } E\} \quad \text{---(4)}$$

rf → relative freq.

n → Total no.

$$\boxed{\frac{rf(H \& E)}{rf(E)} = \frac{No.(H \& E)}{No.(E)}} \quad \text{---(5)}$$

Rewritting (1),

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)} \quad \text{---(6)}$$

→ This eqn expresses the probability of event H is occuring when it is known that event E is occured is the same as probability of E has occured when H is given.

CLASSTIME | Page No.
Date / /

Q Consider the problem of determining the probability that a patient has a certain disease D_1 given that symptom E was observed. For randomly chosen patients, $P(D_1)$ is given as 0.05 and $P(E)$ is 0.15. Also, $P(E|D_1) = 0.95$, based on doctor's previous experience.

$$P(D_1|E) = \frac{P(E|D_1)\,P(D_1)}{P(E)}$$

$$= \frac{0.95 \times 0.05}{0.15}$$

$$= \underline{0.316}$$

→ $$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)}$$

{Probability in terms of Bayesian Theorem}

occurence of prob.

Occurence of negative prob.

↓

$$P(\neg H|E) = \frac{P(E|\neg H)\,P(\neg H)}{P(E)}$$

$$\boxed{P(H|E) \;\Big|\; P(H|E) = \frac{P(E|H)\,P(H)}{P(E|\neg H)\,P(\neg H)}}$$

* $$\boxed{O(H|E) = L(E|H) \cdot O(H)}$$

final eqn to be used in Bayes Theorem.

likelihood of event

defines odds of event

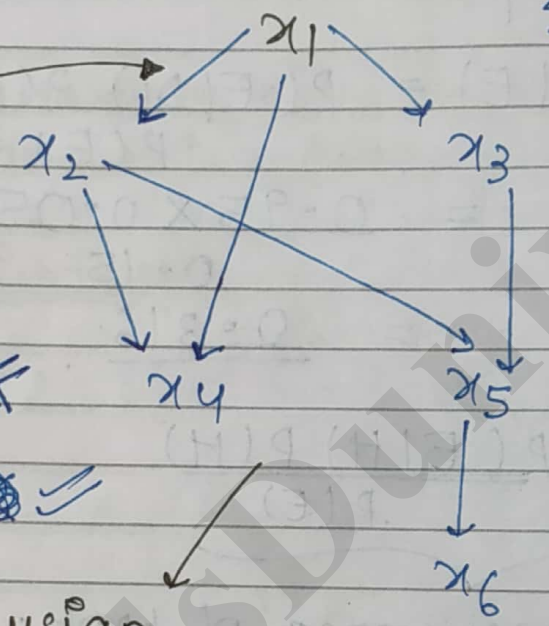
CLASSTIME | Page No.
Date / /

- How to represent real worded probabilities in Baysian N/w

notation used. { x_1 → is an event
$P(x_1)$ → prob. of event x_1 }

{x_2 is dependent on x_1}

{x_3 is dependent on x_1}



$(x_1 | x_2)$ ✗

$(x_2 | x_1)$ ✓

Event x_1 is an independent event whose prob. can be found independently.

Baysian N/w
(multiple events)

$P(x_1, x_2 ---- x_6) = P(x_6 | x_5)$

combined probability.

shows dependency of x_6 on x_5.

$P(A | B)$

When writing in RHS, take events in such an order which has highest subscript as we need to move from highest subscript to root node.

Now for x_5 ↓

$$P(x_5 \mid x_3, x_2)$$

$$\therefore P(x_1 \dots x_6) = P(x_6 \mid x_5)\, P(x_5 \mid x_3, x_2)\, P(x_4 \mid x_2, x_1)\, P(x_3 \mid x_1)\, P(x_2 \mid x_1)\, P(x_1)$$

Q (In exam) Either make Bayesian N/w or given an expression, draw Bayesian N/w.

6.3 Possible Worlds Representation

Identifying multiple meanings from a sentence.

For every sentence, 2 words exist

(event occurs) True — P

False (event doesn't occur) — 1 − P

- Consistent - prob. of an event to occur is known.
- Inconsistent - prob. of occurence of an impossible event

possible worlds representation is based on this.

CLASSTIME | Page No.
Date / /

For more than 1 sentence
↓
possible worlds
(possible values)

2^S → no. of sentences

For 3 sentences → P, Q, P → Q

∴ Possible words = $2^3 = 8$.

To find its consistency, we need to find Truth Table.

representation on basis of P & C

P	Q	P → Q	(~P V Q)
T	T		
T	F		
T	T		
T	F		
F	T		
F	F		
F	T		
F	F		

× → for 2 symbols we can't define 8 entries
(P → Q only depends on 2 stmts P & Q ∴ 8 symbols can't be used)

In real world, we can only have 4 possible cases using 2 symbols.

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

consistent

P	Q	P → Q
F	T	F
T	F	T
F	F	T

→ inconsistent/ impossible.

8 March, 19

CLASSTIME | Page No.
Date / /

Ch-6

Possible World Representation

2 soln

consistent (possible soln) | inconsistent (no possible soln)

P	Q	P → Q
T	T	T
T	F	F
T	T	T
T	F	F
F	T	T
F	F	T
F	T	T
F	F	T

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

→ If K of the truth assignments are consistent then K sets of possible worlds can be modelled with the corresponding interpretations. The probability distribution is then defined over the possible worlds where p_i is the probability that w_i is the actual world and can be represented as :-

$$\sum_{i=0}^{1} p_i = 1$$

→ We use a matrix to represent possible world with 'L' columns and 'K' rows to represent the truth values for a given sentence and it can be represented with

$$q = Vp$$

where q represents the sum of probabilities, and V & P is the product (vector) gained with the help of 'q'.

* Matrix Representation

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

P	Q	P → Q
1	1	1
1	0	0

$$\begin{Bmatrix} P_1 + P_2 + P_3 \\ P_1 + P_3 \\ P_1 + P_3 + P_4 \end{Bmatrix}$$

Q Define sentence S_1, S_2, S_3. Determine probalistic truth values of S_1, S_2, S_3. given pw_1, pw_2, pw_3, pw_4 (p_1, p_2, p_3, p_4)

$S_1 - p$
$S_2 - q$
$S_3 \rightarrow p \rightarrow q$

ques in ppr

Q Baysian N/w or Possible World Representation

8 March, 19

CLASSTIME | Page No.
Date / /

Ch-12

Natural Language Processing

→ chrs. of an Agent
↓
has actuators, sensors, machine learning. natural language processing.
(or implements)

→ Turing Test incorporates NLP, it builds reply & then gives / provides responses.

→ Parsing - used for mapping in NLP.

RTN ATN
2 techniques.

* Terminologies in NLP :-

→ Levels of knowledge used in Language understanding

(1) Phonological - • This knowledge relates the sound to the corresponding words we recogonize.
• Phoneme is the smallest unit of the sound.

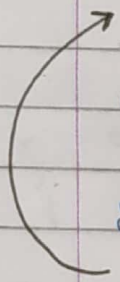
(2) Morphological - • This knowledge relates the word with their construction from the basic units called 'morphemes'. (smallest unit)

eg:- friendly where friend is basic unit (morpheme) & 'ly' is suffix.

(3) Syntactic - • It relates the knowledge to the corresponding words where they are put together to provide a structured or gramatically correct sentences.

(4) Semantic - • It is concerned with meanings of words & phrases and how they can be combined to provide a meaningful sentence.

(5) Pragmatic - • This is high level knowledge which relates to the usage of sentences in diff. context & how the context is going to affect the meaning of the corresponding sentence.



{meaning of sentence changes depending upon situation}

(6) World - World knowledge relates to the (Space set) language a user must have in order to understand and carry on a conversation.

{World is whole set space where every representation is present & agent acts acc. to it}

* General Approaches to Natural Language Understanding

There are 3 ways to represent/understand a language.

(1) with use of keywords & pattern matching.
(2) with combination of syntax or syntactic and semantics
(3) comparing & matching the inputs to real world situations.

(1) → I am _____ } identifying keywords from a given pattern
We are _____.

mapping 2 languages through pattern matching.

eg:- can (keyword) can be derived from I can.

Obselete mechanism.

System to identify scripts & frames.
→ SAM (Script Analyse mechanism).

Initial AI system → ELIZA
(identifies patterns & keywords).

(3) → cat → real world entity depicted as INGEST in scripts.

(2) ⟶ working with syntax & semantics to build meaningful sentences.
{use of parsing in this method}

* Grammar & Language :-

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

L = (set of strings)

can be a single chr or combination of chrs.

basic atomic unit - symbol.

⟶ The finite set of 'V' symbols of the language is called the alphabet or vocabulary.

⟶ A well formed sentences are constructed using a set of rules called "grammar".

⟶ Grammar can be represented with the representation of

$G = (V_n, V_t, s \text{ and } \beta)$

when V_n represents non terminal symbols.

V_t - Terminal symbol. (eg:- Ram, seeta).

eg:- noun. (can be further decomposed) Ram X non a non terminal (as can't be decomposed further).

s - starting symbol

p - It is a finite set of productions or rewrite rules.

- eg - To process xyz

we need to identify meanings of x, y, z

Rewriting in terms of Grammar G.

$xyz \longrightarrow xtz$ {y rewritten to t}.

$\{V_n \rightarrow V_t\}$ { providing y with a fixed constant value}.

→ eg:- A simple Grammar G can be constructed with vocabulary

$G_N = \{ S, NP, V, VP, N, ART \}$.

- G_N : Vocabulary consists of non terminal symbols.
- S : Start Symbol.
- NP : Noun phrase
- VP : Verb phrase
- ART : Article

Now we provide QT (terminal) values

ART → will consist of a, an, the.

ART → Ram ✗ (Ram is not an article).

Q_T = { boy, popsicle, frog, ate, kissed, flew, the, a }

amp. To build a sentence, we need to know
↓ production rules (P)
helps to build 's'.

Prod. Rules for this example
↓
P :- S → NPVP

we can't provide them terminal symbols directly.

To convert them to terminal symbols, we need prod. rules.

prod. rule of NP. NP → ART N

2 combinations (the/a)
∴ sentence can be started using The or a.

N → boy/frog.

prod. rule of VP.

VP → V NP
(V ART N) → Same as.
(NP)

N → boy, popsicle, frog.
V → ate, kissed, flew.
ART → the, a.

* Now we ~~build the~~ sentence

S → NP VP
↓
→ ART N VP
→ ART N V NP
→ ART N V ART N

We terminate when we get terminal symbols i.e they become leaf nodes.

now we can assign terminal values.

the boy ate the popsicle.
1st sentence.

We are able to get possible worlds

Now the boy kissed the frog.
true
not a representation in real world.
(not a good soln), i.e not semantically correct).

multiple combinations

the popsicle ate the boy.

CLASSTIME | Page No.
Date / /

NOTE: Building a language is easy but implementing it using semantics is difficult.

NOTE: Without a starting symbol, sentence wont be build.

Q (in exam) parsing (topdown or bottom up approach).

Building
Sentence given prod. rules.

Q (building NLw RTN & ATN).

11 march, 19 Ch-12

CLASSTIME | Page No.
Date / /

* Types of Grammar :-

• Chomsky Grammar :-

(1) Type 0 - Type 0 is known as simple Grammar & it can be obtained by making a simple restriction that 'y' can't be empty while rewriting the sentences.

$$xyz \longrightarrow xwz$$

(2) Type 1 - Type 1 is known as 'context sensitive Grammar' and it can be obtained by adding the restrictions that the length of string on the RHS must be atleast as long as the string on LHS, i.e the non terminals symbols will increase

eg :- xyz, abyz } $x \rightarrow ab$ → length increases upon rewritting.

In production of xwz from xyz, y must be a nonterminal symbol.

(3) Type 2 - Type 2 Grammar is 'context free Grammar'. It can be characterised with the help of ↓

<symbol> → <symbol1> - - - - <symboln>

where n should be greater than or equal to 1. $(n \geq 1)$
LHS must have a single non terminal symbol, so that it can be rewritten.
{no restriction on length as in case of Type 1}

(a) widely used in designing of F.A.

Type 3 - It is known as 'Regular Grammar' or 'Finite State Grammar', and it can be represented with form of

$A \rightarrow aB$
$A \rightarrow a \qquad B \rightarrow b$

where capital letters are non terminals & small letters are terminals.

- Structural Representation of A Sentence

– Representation is in form of a List.
– Same tree like structure as in previous topic of sentence building.

(S) → NP VP

root node → signifies NP has some elements

S(NP((ART the)
(N boy))

This (represents that S has some elements in it

) → termination of combinations in NP.

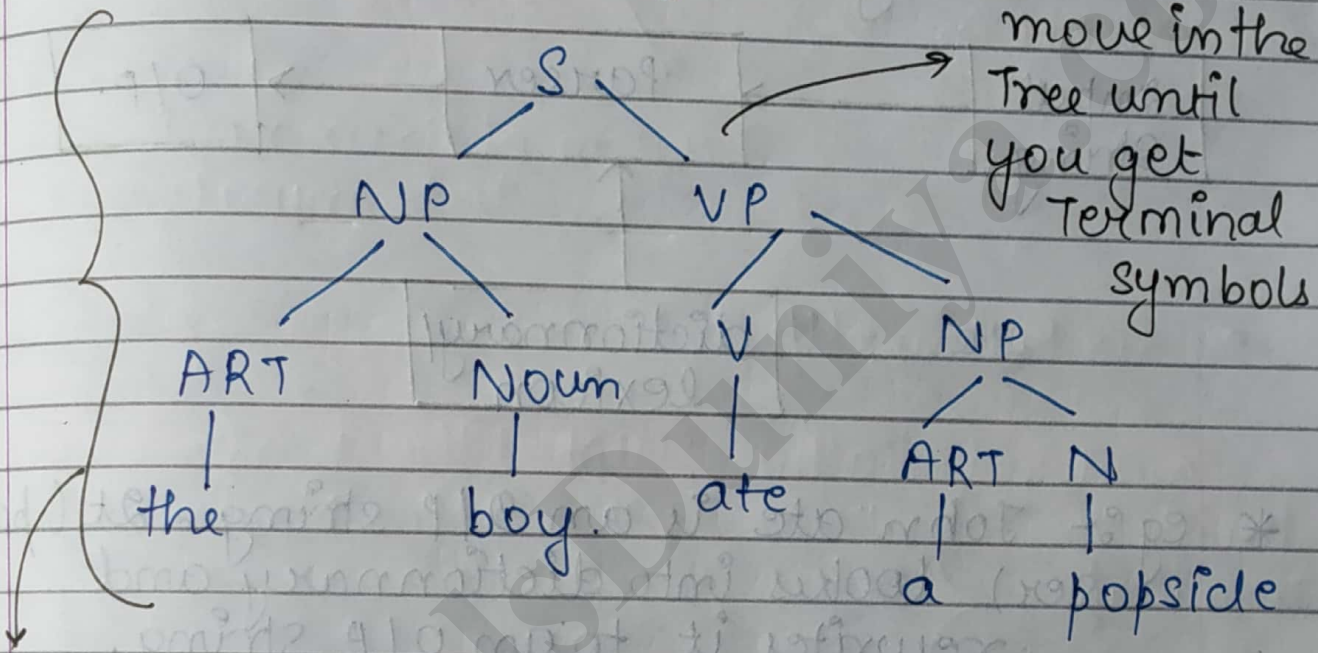
S not terminated as S contains VP also.

CLASSTIME | Page No.
Date / /

(VP (V ate)
(NP (ART a)
(N popsicle))))

for VP
for S.
for NP

Now we draw the Structure in terms of a Tree.



Structural representation of a sentence

{PP - preposition phrase}

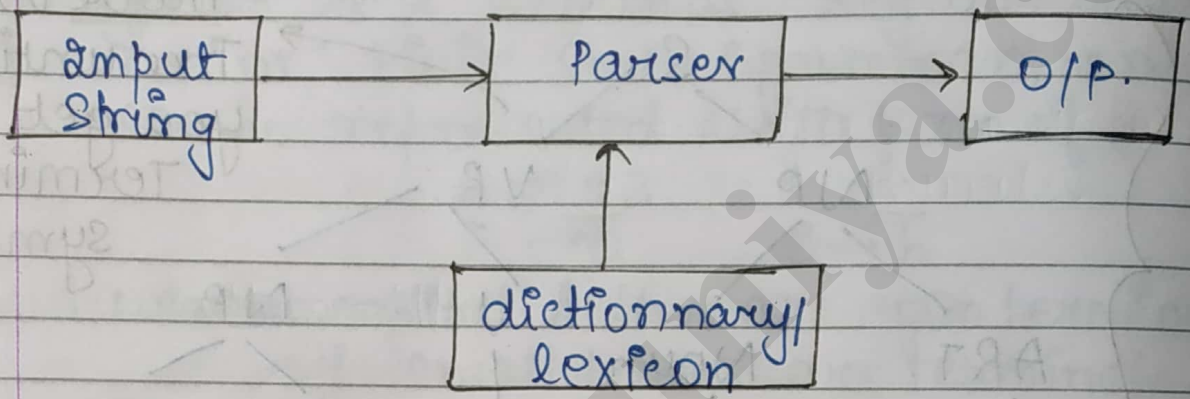
* Basic Parsing Techniques :-

{How to analyse the meaning of each symbol/ word of a sentence}

→ Parsing is the process of analysing a sentence by taking it apart word by word and determining its structure from its constituent parts & subparts.

→ To determine the meaning of a word, a parser must have access to lexical or dictionary from where the possible words can be identified (including their semantic) to apply it.

Block Diagram :-

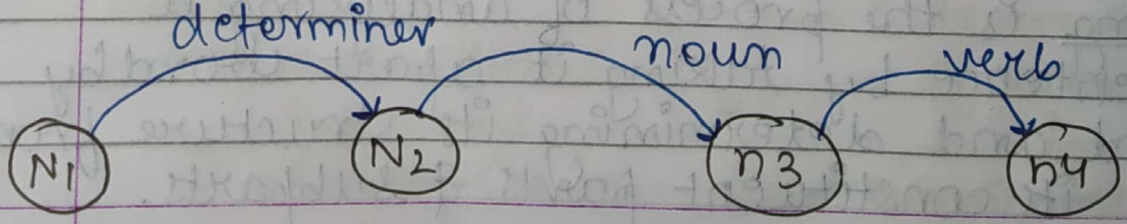


* eg :- John ate is an I/P string. It (parser) looks into dictionnary and rewrites it to an O/P string.

→ For John, there is no other word for it as it is a Terminal symbol.
For ate, it has similar words {eat, ate, eaten} and thus it starts to produce an O/P acc. to it.

{Refer Table (in Book)}.

* Transition Networks :-



how we are transitting from one form to another using grammar components / labels

Transition N/w.

condition.

If cond = True go to next node
else get up to the same position only.

→ A Transition Network consists of a no. of nodes and labelled arcs. The nodes represent diff. states in traversing a sentence & the arcs represents rules or Test conditions req. to make the transition from the current state to the next state.

→ If a Transition N/w can be successfully traversed, then only we have a recogonized permissiable sentence structure.

* Parsing Techniques

Top down. Bottom Up.

① Top down - • Parsers maybe designed to process a sentence either in Top down or Bottom up approach where top down parser begins by hypothesing a sentence with a symbol 's' and successively predicts lower level constituents until individual Terminal symbols can be identified. i.e reaching up to leaf nodes.

eg:- S → NP VP
→ Name VP
→ Kathy V NP
→ Kathy jumped NP
→ Kathy jumped ART N
→ Kathy jumped the N
→ Kathy jumped the horse.

Top down approach (don't miss steps in b/w & only replace one symbol in one step)

movement from non terminals to terminals.

- For ~~Bottom~~ Up :-

movement from Terminals to non terminals.

Kathy jumped the horse
" " the N
⋮
S → NP VP.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well