20/1/16
wednesday

## Theory of computation - TOC :

Deals with whether a particular problem is solvable or not and if solvable calculate its time complexity and space complexity.

**Classification :**

1) Computability theory : deals with whether a problem is solvable or not.
2) Complexity theory : deals with whether a problem is solvable or not, if solvable then calculate its time complexity and space complexity.

Three mathematical models introduced for how to process data are:

1) Finite automata : - No memory
   - knows current state only.
   - It doesn't knows previous state.
2) Push down automata : - some amount of memory.
3) Turing machine : - more amount of memory.
   - commonly used in digital computers.

21/1/2016
Thursday

# Module : 1

↳ **Set :** well defined collection of various objects.

↳ **Elements :** objects of the set.

↳ Representation of sets using capital letters.

eg: $A = \{a, e, i, o, u\}$

↳ **Classification**

1) Finite set - finite no. of elements

2) Infinite set - infinite no. of elements

↳ **Subset :**

A set $A \subseteq B$ means all elements of A are present in B.

↳ **Empty set :** no elements, represented $A = \{\ \}$

↳ **Power set :** set of all subsets of a particular set including null set.

eg. $A = \{1, 2, 3\}$

$$P(A) = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

↳ **Intersection :** ~~combination of~~ collection of common elements from both the sets.

↳ **Union :** combination of all elements of both sets.

↳ **Difference :** elements in A are not present in set B.

↳ eg: $A = \{1, 2, 3\}$ $B = \{1, 2\}$

Union → $A \cup B = \{1, 2, 3\}$

Intersection → $A \cap B = \{1, 2\}$

Difference → $A - B = \{3\}$

↳ **Disjoint set :** set which are having no elements in common.

↳ Union, intersection and difference are the 3 operations performed on sets.

↳ **Commutative law:**

$A \cup B = B \cup A$

$A \cap B = B \cap A$

↳ **Associative law :**

$A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$

↳ **Distributive law :**

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

↳ **Idempotent law:**

$A \cup A = A$

$A \cap A = A.$

↳ **Absorption law:**

$A \cup (A \cap B) = A.$

$A \cap (A \cup B) = A.$

↳ **Demorgan's law:**

$(A \cup B)^c = A^c \cap B^c$

$(A \cap B)^c = A^c \cup B^c$

↳ **Other laws:**

$(A')' = A.$

$A \cap A' = \phi$

$A \cup A' = U$

$A \cup \phi = A$

$A \cap \phi = \phi$

$A \cup U = U$

$A \cap U = A.$

↳ **Function:** A fn from set A into set B is a relation from A to B such that each element of A is related to exactly one element of B.

↳ **Domain:** Let f be a fn from P to Q, then set P is called domain of the fn.

↳ **Codomain:** Let f be a fn from P to Q, then set Q is called codomain of the fn.

22/1/16 Friday

↳ **Image:** The element in B is said to be an image of element in A if a relation exist in b/w them.

↳ **Range:** Elements in B that are related to elements in A

↳ **Pre-image:** The element in A ie related to the element in B is said to be the preimage of B in A.

↳ **Operat$^n$'s on fns:**

$f + g(x) = f(x) + g(x)$

$f * g(x) = f(x) * g(x)$

$f(x) = 3x+1, \quad g(x) = x^2.$

$f + g(x) = f(x) + g(x)$

$= 3x+1 + x^2$

$= \underline{x^2 + 3x + 1}$

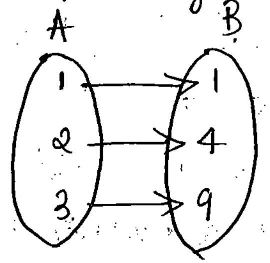
$f * g(x) = (3x+1)\, x^2$

$= \underline{3x^3 + x^2}$

↳ <u>One - One fn : (injective relation)</u>

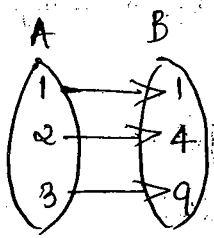
If $f : A \to B$, then every element in A should have a unique image in B.

eg:

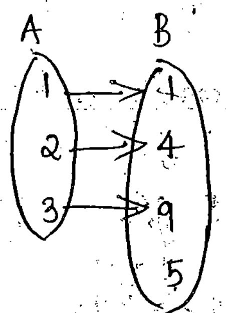


↳ <u>On- to fn : (surjective fn)</u>

Every element in B should have a preimage in A.



onto fn



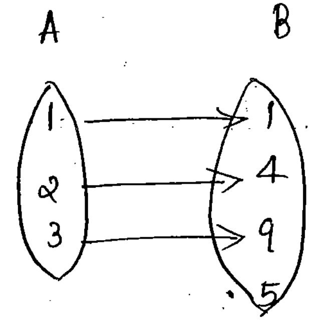
not-onto fn.

↳ <u>Bijective fn:</u>

If $f : A \to B$ is both onto and one-one, then it is called bijective fn.

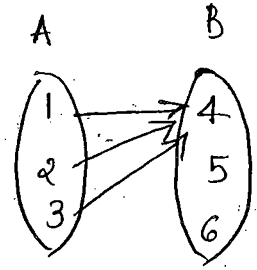
↳ <u>Into fn:</u>

Atleast one element in B should not have a preimage in A.



↳ <u>Many one fn :</u>

Many elements in A is related to single element in B.



↳ <u>Equal fn:</u>

If $f : x \to y$ and $g : x \to y$ are two fns then they are said to be equal, if:

$$f(a) = g(a)$$

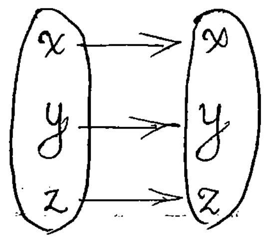
eg. $x = \{1, 2, 3\}$ $y = \{a, b, c\}$

$f = \{(1,a)\ (2,b)\ (3,c)\}$

$g = \{(1,a)\ (2,b)\ (3,c)\}$

## Identity fn :

If $f: A \to A$ be a fn then every elements of A is related to itself.

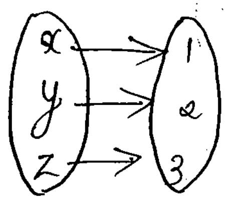



## Investable fn :

$f: A \to B$.

$f = \{(x,1)\ (y,2)\ (z,3)\}$

f is one-one and on-to.




$f^{-1} = \{(1,x)\ (2,y)\ (3,z)\}$

## Composition of fns :

$f: A \to B$

$g: B \to C$.

$fg: A \to C$.

$fg(x) = f(g(x))$

eg : $f : x^2$

$g : (x+1)$

$fg(x) = f(g(x))$

$= f(x+1)$

$= (x+1)^2$

Relat^n on a fn A&B is a subset of A×B.

27/1/16 wednesday

## Fundamental prooving techniques :

1) Mathematical induction
2) Pigeon-hole principle.
3) Diagonalisation principle.

## 1) Mathematical induction :

Prove the given statement is true for all natural no's. 2 steps :

1) Base case – $n=0$ or 1, statement is true.
2) Inductive case. – assume statement is true for n, then we prove statement is true for n+1. (induct^n hypothesis).

Q. $1+2+3+\dots\dots+n = \frac{n(n+1)}{2}$.

ans: Base case:

$n=0$,

$LHS = P(0) = 0$

$RHS = \frac{0 \times 1}{2} = 0$. $LHS = RHS$. ∴ statement is true for $n=0$.

Induction hypothesis:

Assume $n=k$, is true for the statement.

$LHS = P(k) = 1+2+3+\dots\dots+k = \frac{k(k+1)}{2}$.

Inductive step:

~~Assume~~ we have to prove this statement is true for $n=k+1$.

$LHS = P(k+1) = 1+2+3+\dots+k+k+1$.

$= \frac{k(k+1)}{2} + (k+1)$

$= \frac{k(k+1)+2(k+1)}{2}$

$= \frac{(k+1)(k+2)}{2}$ ~~= RHS~~

$RHS = \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$

∴ $LHS = RHS$

Q. For all $n \geq 1$, P.T $1^2+2^2+3^2+\dots\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$

ans: Base case:

$n=0$,

$LHS = P(0) = 0$

$RHS = \frac{0(0+1)(2\times0+1)}{6} = 0$

$n=1$,

$LHS = P(1) = 1^2 = 1$

$RHS = \frac{1(1+1)(2+1)}{6} = \frac{1\times2\times3}{6} = 1$

$LHS = RHS$. ∴ this statement is true for $n=1$.

Induction hypothesis:

Assume this statement is true for the value $n=k$.

$$LHS = P(k) = 1^2 + 2^2 + 3^2 + \dots + k^2 = \frac{k(k+1)(2k+1)}{6}$$

<u>Inductive step</u>:

we have to prove this statement is true for $n = k+1$.

$$P(k+1) = 1^2 + 2^2 + 3^2 + \dots + k^2 + (k+1)^2$$

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

$$= \frac{k(k+1)(2k+1) + 6(k+1)^2}{6}$$

$$= \frac{k+1\left(k(2k+1) + 6(k+1)\right)}{6}$$

$$= \frac{k+1\left(2k^2 + k + 6k + 6\right)}{6}$$

$$= \frac{k+1\left(2k^2 + 7k + 6\right)}{6}$$

$$= \frac{k+1\left(2k^2 + 4k + 3k + 6\right)}{6}$$

$$= \frac{k+1\left(2k(k+2) + 3(k+2)\right)}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

$$RHS = \frac{(k+1)(k+1+1)(2(k+1)+1)}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

$\therefore$ LHS = RHS

Q. P.T $2^n > n$ for all +ve integer's $n$.

ans: <u>Base case</u>:

$n = 1$,

$2^1 > 1$

$2 > 1$

LHS > RHS

This statement is true for $n = 1$.

<u>Induction hypothesis</u>:

Assume statement is true for $n = k$.

$P(k) = 2^k > k$.

<u>Inductive step</u>:

we have to prove this statement is true for $n = k+1$.

$$P(k+1) = 2^{k+1} > 2k$$
$$= 2^k \cdot 2^1 > 2k$$

2) <u>Pigeon hole principle</u>:

If $n+1$ or more pigeon fly to $n$ pigeon hole then atleast one pigeon hole is occupied by atleast 2 pigeons.

<u>Examples</u>:

8 people choosen at random then 2 of them are born on the same day of the week.

3) <u>Diagonalisation principle</u>:

The compliment of the diagonal is different from each row.

↳ <u>Example</u>:

$A = \{a, b, c, d, e, f\}$

$R = \{(a,b)\ (a,d)\ (b,b), (b,c)\ (c,c), (d,b)\ (d,c), (d,e)\ (d,f)\ (e,e), (e,f), (f,a)\ (f,c)\ (f,d)\ (f,e)\}$

| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | | X | | X | | |
| b | | X | X | | | |
| c | | | X | | | |
| d | | X | X | | X | X |
| e | | | | | X | X |
| f | X | | X | X | X | |

$D = (b, c, e)$

$\overline{D} = (a, d, f)$

↳ <u>Primitive and partial recursive fn</u>:

A partial fn from $X \to Y$ is a fn from $f: X' \to Y$ where $X'$ is a subset of $X$. If

If $X' = X$ then $f$ is called total fn and is equivalent to a fn.

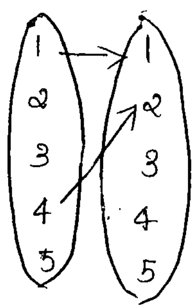
28/1/16
Thursday

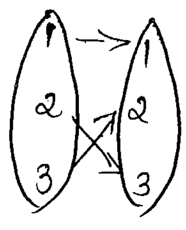
eg: Consider ☐re root fn restricted to integers.

$g : z \to z$

$g(n) = \sqrt{n}$.

$z \to z$ $\qquad$ $x \to y$





eg: total fn

eg: partial fn.

## Primitive recursive fn:

1) Initial fn
2) Composition and recursion

if it is an initial fn or if it can be obtained from initial fn by composition and recursion a finite no. of times.

1) Initial fn:

① Zero function. - returns zero value only.

$Z(x) \to 0$.

② Successor fn - returns successive value

$S(x) \to x+1$

③ Projection fn - returns the specified value.

$- \; U_2^3(4,3,2) = \underline{3}$

2) Composition and recursion fn:

If $f_1, f_2 \ldots f_k$ ~~→ k~~ are partial fns of n ~~variabl g → n~~ variables and $g$ is a partial fn of k variables, then composition of $g$ with $f_1, f_2 \ldots f_k$ is a partial fn of n variables defined by

$g(f_1, f_2 \ldots f_k)$

$g(f_1(x_1, x_2 \ldots x_n) \ldots f_2(x_1, x_2 \ldots x_n) \ldots f_k(x_1, x_2 \ldots x_n))$

eg: $f_1(x,y) = x+y$ $\qquad$ $g(x,y,z) = x+y+z$.

$f_2(x,y) = 2x$

$f_3(x,y) = xy$

$g(f_1(x,y), f_2(x,y), f_3(x,y)) \leftarrow g(f(x))$

$= g(x+y, 2x, xy)$.

$= x+y+2x+xy$

$= \underline{3x+y+xy} \to 2$ variables.

Recursion

$f(x) \to N$

$k, h(x,y)$

1/2/16
Monday

A fn $f(x)$ over $N$ is defined by recursion if there exist a const $k$ and a fn $h(x,y)$ such that $f(0) = k$ and $f(n+1) = h(n), f(n)$.

eg: $n!$

$f(0) = 1 \qquad (0! = 1)$

$$f(n+1) = h(n, f(n))$$
$$= h(S(n), f(n))$$
$$= h(n+1, f(n))$$
$$= h(0, f(0))$$
$$= h(S(0), f(0))$$
$$= h(1,1)$$
$$= \underline{1}$$

$n!$

$f(0) = k$

$f(n+1) = h(x,y)$

$2!$

$f(0) = 1$

$$f(\overset{n+1}{2}) = h(\overset{n}{S(1)}, \overset{f(n)}{f(1)})$$
$$= h(2, f(1))$$
$$= h(2, h(S(0), f(0)))$$
$$= h(2, h(1,1))$$
$$= h(2, 1\times 1)$$
$$= h(2,1)$$
$$= 2\times 1$$
$$= \underline{\underline{2}}.$$

2/2/16
Tuesday

A fn $f$ of $n+1$ variables is defined by recursion if there exist a fn $g$ of $n$ variables and a fn $h$ of $n+2$ variables and $f$ is defined as:

imp

$$\boxed{\begin{aligned} &f(\overbrace{x_1, x_2 \ldots x_n}^{n}, \overset{1}{0}) = g(x_1, x_2 \ldots, x_n) \quad \text{— ①} \\ &f(x_1, x_2 \ldots x_n, y_n+1) = h[\underbrace{x_1\, x_2 \ldots x_n}_{n}, \overset{1}{y}, \underbrace{f(x_1, x_2 \ldots x_n, \overset{1}{y})}_{2}] \end{aligned}}$$

Q. PT $f_1(x,y) = x+y$ is primitive recursive.

$f_1(x,y) = x + y$ - 2 variable fn.

$g \to 1$ variable, $h \to 3$ variable.

$f(x,0) = x+0$

$= x = U_1^1(x)$

$g(x) = x = U_1^1(x)$.

$f(x,0) = g(x) = U_1^1(x)$

$f_1(x, y+1) = (x+y)+1$

$= f_1(x,y) + 1$

$= S(f_1(x,y))$

$= S\left[U_3^3(x, y, f_1(x,y))\right]$

$\to h(x,y,z) = S\left[U_3^3(x,y, f_1(x,y))\right]$

$= S\left[U_3^3(x,y,z)\right]$

Conclusion:

Since. $f_1$ can be obtained ~~by~~ from ~~app~~ initial fn by applying recursion & composit^n finite no. of times.

4/2/16
thursday

Q S.T. the fn. $f_2(x,y) = x * y$ is primitive recursive.

ans: $f_2(x,y)$ is a fn of ~~single~~ 2 variable and to apply recursion we need a fn $g$ of 1 variable and $h$ of 3 variables.

$f_2(x,0) = x*0 = Z(x)$.

$g(x) = x = Z(x)$

$f_2(x,0) = g(x) = Z(x)$.

$f_2(x, y+1) = (x * (y)+1)$

$= \cancel{f_2(x,y)}\ (x*y) + x$.

$= f_2(x,y) + x$.

$= f_1(f_2(x,y), x)$

$h(x, y, f_2(x,y)) = f_1(f_2(x,y), x)$

$= f_1\left(U_3^3(x,y, f_2(x,y)), U_1^3(x,y, f_2(x,y))\right)$

$h(x,y,z) = f_1\left(U_3^3(x,y,z), U_1^3(x,y,z)\right)$

Conclusion:

Since $f_2$ can be obtained from initial fn by applying recurs^n & composit^n finite no. of times.

Q. S.T $f(x,y) = x^y$ is a primitive recursive fn.

ans: $f(x,y)$ is a fn of 2 variables and to apply recursion we need a fn $g$ of 1 variable and $h$ of 3 variables.

$$f(x,y) = x^y$$

$$f(x,0) = x^0 = 1 = S(0)$$

$$g(x) = 1 = S(0).$$

$$f(x,y+1) = x^{y+1}$$
$$= x^y \cdot x$$
$$= f(x,y) \cdot x.$$
$$= \cancel{f_2\ x,y}\ f_2(f(x,y), x)$$

$$h(x,y,f(x,y)) = f_2(f(x,y), x)$$
$$= f_2\left(U_3^3(x,y,f(x,y)), U_1^3(x,y,f(x,y))\right)$$

$$h(x,y,z) = f_2\left(U_3^3(x,y,z), U_1^3(x,y,z)\right)$$

Conclusion:
Since $f$ can be obtained from initial fn by applying recursⁿ & composⁿ finite no. of times.

↳ Countable set:

• A set is said to be countable, if it's elements can be numbered using natural no$_s$. ie there exist a one to one mapping from this to the set of natural no's.

• A countable set is either finite or countably infinite. (denumerable).

- Set of prime no → countable.
- Set of all integers → countable & countably infinite.

Q. P.T the set of real no's b/w 0 and 1 is uncountable.

ans:

| | | | | |
|---|---|---|---|---|
| $f_0$: | .9 | 4 | 2 | 4 |
| $f_1$: | .6 | 8 | 6 | 2 |
| $f_2$: | .2 | 8 | 6 | 4 |
| $f_3$: | .6 | 5 | 3 | 2 |
| $f(n)$: | .4 | 1 | 5 | 7. |

Diagonal 9362.

taking 9's compliment of 9362.

$$
\begin{array}{r}
.9999 - \\
.9362 \\
\hline
.0637
\end{array}
$$

There exist other no's which are not included in the matrix. ∴ the set is uncountable.

## Partial recursive fn:

1) Initial fn.
2) Composit$^n$, recursion, minimizat$^n$.

## Minimizat$^n$:

Let a fn $g(x,y)$ is defined over 2 variables $x$ and $y$. then the minimizat$^n$ operator '$\mu$' is defined over $g(x,y)$ as follows:

$\mu_y(g(x,y))$ = minimum value of $y$ such that $g(x,y)=0$

eg: $g(0,1) = 1$
$g(0,2) = 2$ minimum value of
$g(0,3) = 0$ $\underline{f(0)=3}$
$g(0,4) = 0$

Q. S.T $f(x) = x/2$ is partial recursive fn.

ans: $f(x) = x/2$

$y = x/2$

$2y = x$

$2y - x = 0$

values of $x$ should be even, then only we can perform minimizat$^n$.

**Alphabets**: finite non empty ~~of~~ set of symbols.
- set of symbols
- denoted by $\Sigma$. eg: $\Sigma = \{a,b\}$

**Strings** - by joining alphabets string can be formed and is denoted as $\omega$.
- $|\omega|$ - length of string.
- eg: $|\omega| = abba = 4$.

**Empty string**: $\lambda, \epsilon$ ($\lambda$-empty string) contains no strings alphabets.

**Kleen Closure or star closure**: denoted by $\Sigma^*$
- $\Sigma^* = \{\lambda, a, b, abb, ab\}$
- set of all strings obtained by alphabets &

empty string.

↳ **Positive closure:** denoted by $\Sigma^+$.

- set of all strings excluding empty string:

$\Sigma^+ = \{ a, b, abb \dots \}$

$\Sigma^+ = \Sigma^* - \epsilon$

↳ **Concatination of string:**

$w_1 = 100 \quad w_2 = 01$

$\Sigma = \{0, 1\}$

$w_3 = 10001$

↳ **Reversing of a string:**

$w_1 = 100$

$R = 001$

↳ **Language:** subset of all strings given by alphabets.

Questⁿ → Language of all strings consisting of n zeroes followed by n ones.

- $\Sigma = \{0, 1\}$, equal no. of 0s & 1s.
- Language represented by L.
- $L = \{\lambda, 01, 0011, 000111 \dots\}$
- $L \subseteq \Sigma^*$

eg: set of all strings ~~equ~~ containing equal no. of zeroes and ones.

- $\Sigma = \{0, 1\}$
- $L = \{\lambda, 01, 10, 0011, 1100 \dots\}$

↳ **Grammer:** denoted by G.

- $G = (V, T, P, S)$ (quatripple)
- V → set of non-terminal, always denoted by capital letters
- T → set of terminals, always denoted by small letters
- P → productⁿ rule
- S → start symbol and also non-terminal represented using capital letters.

Q. Consider a grammer $G = [\{S\}, \{a, b\}, P, S]$

$S \to \lambda, \ S \to aSb$

ans:

$S$

$\Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aabb$

5/2/16 Friday

$L = \{ a^n b^n : n \geq 0 \}$

Q. Find the grammer that generates $L = \{a^n b^{n+1}, n \geq 0\}$.

ans. $S \to aSb$

$S \to b$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aaaSbbb$

$\Rightarrow aaabbbb$

9/2/16 Tuesday

imp → Noam Chomsky classified grammer in to 4 based on product$^n$ rule.

1) Type 0:

- Having less restrictions or unrestricted.
- Called unrestricted grammer or phase structure grammer
- Every product$^n$ will be of the form $\alpha \to \beta$, $\alpha \neq \lambda$, $\alpha$ & $\beta \to (V \cup T)^*$
- $\Sigma^* \to$ 0 or more elements.
- $\Sigma^+ \to$ 1 or more elements.
- Large no. of strings can be generated.
- Language generated by type 0 grammer is called recursive enumerable language.

eg: $aBc \to a.bc.c$

2) Type 1:

- Also called context sensitive grammer.
- Every product$^n$ rule is of the form $\alpha \to \beta$.
- $\alpha, \beta \to (V \cup T)^*$, $|\beta| \geq |\alpha|$, $\alpha \neq \lambda$.

ie no. of elements of $\beta \geq$ no. of elements in $\alpha$.

- No. of strings generated is less than that of strings generated in type 0.
- ~~P~~ A product$^n$ rule of the form, $\phi A \psi \to \phi \alpha \psi$ is called type 1 product$^n$ where A is a variable, $\phi$ is called left context, $\psi$ is called right context and $\phi \alpha \psi$ is called the replacement string.

eg: $aAbcD \to abcDbcD$ [A replaced by bcD]

3) Type 2:

- Also called context free grammer.

**Note:**

$Abcd \rightarrow bcd$

- not type 1 grammer.
- bcoz no. of elements of RHS is not greater than or equal to no. of elements of LHS.

- No left context or right context.
- Pdn rule, $A \rightarrow \alpha$
- eg: $S \rightarrow Aa$
  $S \rightarrow Bc$
- $\alpha \rightarrow (V \cup T)^*$

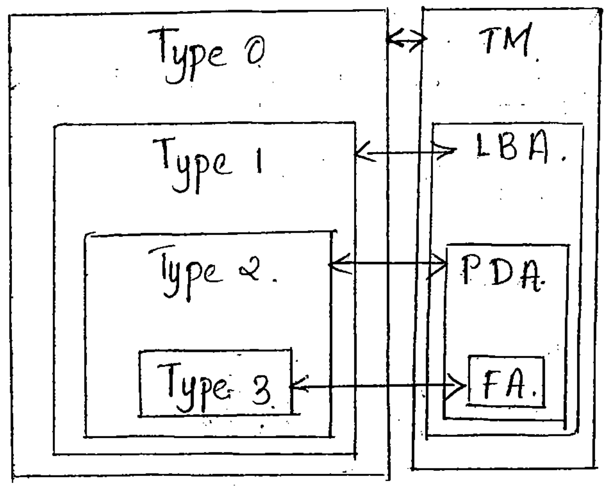
4) Type 3:

- Also called regular grammer.
- Pdn rule of the form:

$A \rightarrow aB / a$. → right linear grammer

$A \rightarrow Ba / a$ → left linear grammer.

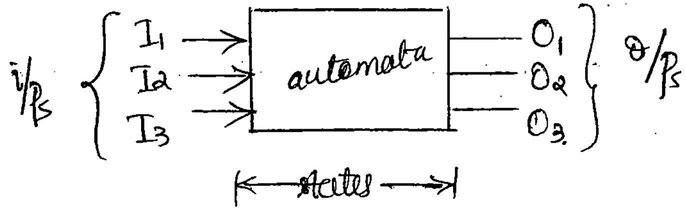
- Most restricted grammer.
- ~~Rtg~~ Left side should contain a single non terminal and the RHS ~~the~~ can be a terminal followed by a non terminal or a non terminal followed by a terminal.
- Language generated this grammer is the least no.




- TM - turing machine - can solve all types of G.
- LBA - Linear Bounded Automata. - can solve type 1, type 2 and type 3 Gs.
- PDA - Push Down Automata. - can solve type 2 and type 3 Gs.
- FA - Finite Automata - can solve type 1 G.

11/2/16
Thursday

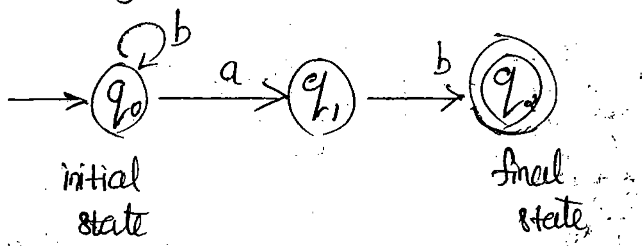
# Automata




- **State diagram**: used to represent transitions.

  $\Sigma = \{a, b\}$

  string = bbab.



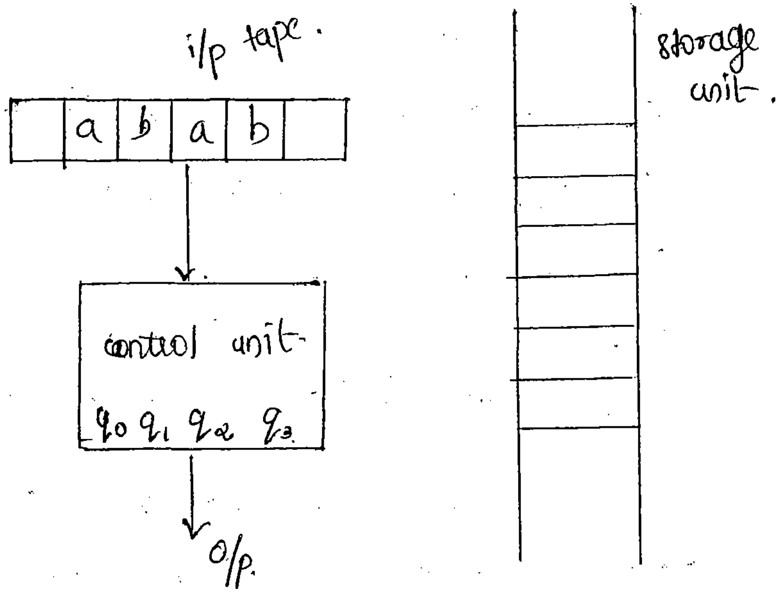

↳ **Moore machine**: system that depends on i/p and not on the states.

↳ **Melay machine**: system that depends both i/p and state.

↳ **Automata components**:

**i/p tape** → set of i/p symbols ie combinatⁿ of string

- i/p tape is divided in to different □re groups which contains alphabets.




**Main processing unit** → control unit → moves to different states.

**Storage unit**: → required data that is needed later is stored in memory storage unit.

↳ **Finite Automata (FA)**

2 types:

1) DFA
2) NFA.

1) ## Deterministic FA :

5 tuples are there to represent a FA.

$(Q, \Sigma, \delta, q_0, F)$

$Q \rightarrow$ set of states (finite non empty sets)

$\Sigma \rightarrow$ i/p alphabets (finite non empty set)

$q_0 \rightarrow$ initial state

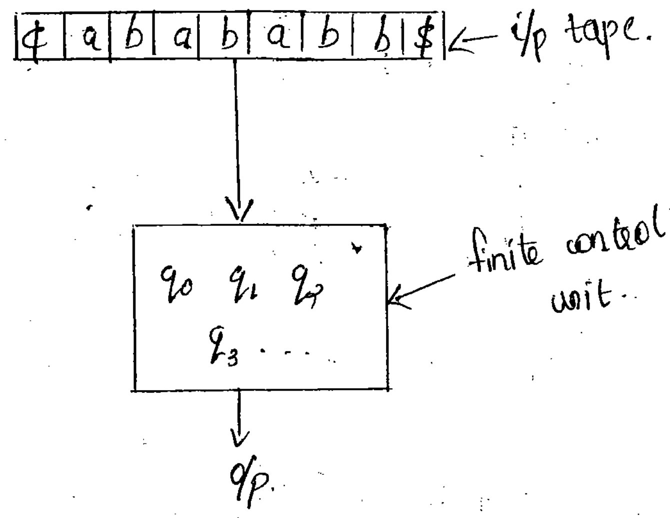
$F \rightarrow$ set of finite states

$\delta \rightarrow$ transit^n fn, represented as $\delta(q_0, a) \rightarrow q_1$

FA can be represented 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where $Q$ is the finite non empty set of states, $\Sigma$ is the finite non empty set of i/p alphabet, $q_0$ which is an element of $Q$ which is the initial state / start state, $F$ is the subset of $Q$ is the set of finite state or accepting state, $\delta$ is the transit^n fn which maps $Q \times \Sigma \rightarrow Q$. which is called direct transit^n fn. This fn describes the change of state during transition. Represented by transit^n table / state dgm.

↳ ## Operat^ns of finite automata :

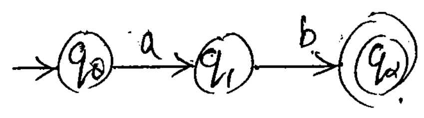
- FA has no memory.
- 2 elements in FA are i/p tapes & control units.
- ¢ and \$ is used to represent the starting and ending of a string in i/p tape.
- Reading unit reads the string from left ~~right~~ to ~~left~~ right.
- If the system is in final state, that particular ~~state~~ string will be accepted otherwise rejected.




1) ## Transition graph / transit^n diagram / state diagram :

vertices - states , pointed arrow without any i/p.

edges - i/ps , final state represented with double O/e.

eg : 

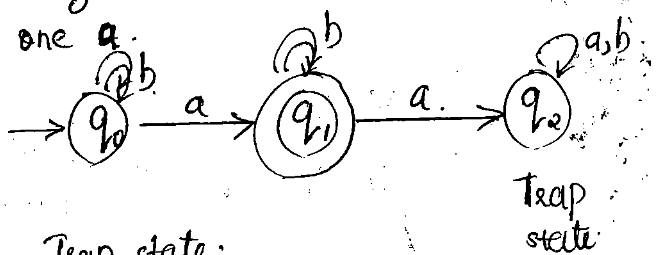

2) Transition table :

| Q/Σ | 0, | 1 |
|---|---|---|
| $q_0$ | $q_1$ | $q_0$. |
| $q_1$ | $q_2$ | $q_1$. |

3) Transitⁿ fn:

$$\delta(q_0, 0) = q_1$$

imp Q. Design a DFA that accepts all strings with exactly one a.

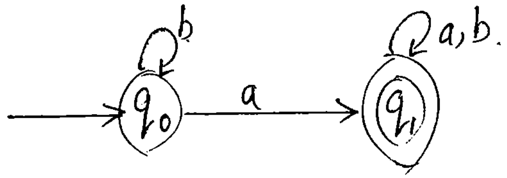
ans: 


→ bbbabbb
→ a
→ bba
→ abb

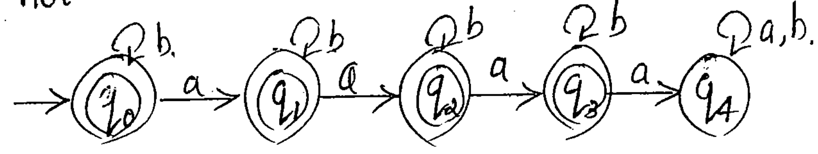
Trap state :

- It has incoming arrow only.
- It has no. outgoing arrow.

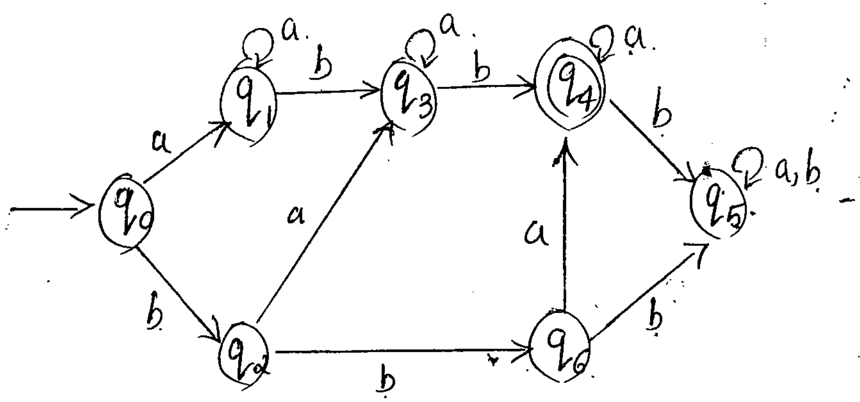
Q Construct a DFA that accepts all strings with atleast one a.

ans: 


Q Construct a DFA that accepts all strings with not more than 3a.

ans : 


Q. All strings with atleast one a and exactly two b.

ans : 


HW Q Construct a DFA that accepts all strings starting with ab.

ans :