# BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT

AVALAHALLI, DODDABALLAPUR ROAD,
YELAHANKA, BANGALORE

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## COURSE MATERIAL

# AUTOMATA THEORY AND COMPUTABILITY
# (17CS54 / 18CS54 )


PREPARED BY:
DR. HEMAMALINI B H
ASSOCIATE PROFESSOR

# INDEX

| Sl. No. | Date | Subject | Page No. |
|---|---|---|---|
| | | | |



Platinum

## INTRODUCTION TO FINITE AUTOMATA

FA are useful model for many imp kinds of hardware & software. Some of them are

1) s/w for designing & checking the behavior of digital circuits.
2) The "lexical analyzer" of a typical compiler, i.e. the compiler component that breaks the input text into logical units, such as identifiers, keywords and punctuation.
3) s/ws for scanning large bodies of text, such as collections of web pages, to find occurences of words, phrases or other patterns.
4) s/ws for verifying systems of all types that have a finite no of distinct states, like communication protocols or protocols for secure exchange of information.

### The Central concepts of Automata Theory

#### Alphabets:

An alphabet is a finite, non-empty set of symbols. It is conventionally denoted by the symbol $\Sigma$. Common alphabets are

1) $\Sigma = \{0, 1\}$, the binary alphabet.
2) $\Sigma = \{a, b, \ldots, z\}$, the set of all lower-case letters
3) The set of all ASCII characters, or the set of all printable ASCII characters.

#### Strings:

A string (or word) is a finite sequence of symbols chosen from some alphabet.
For eg 011, 111 are strings from binary alphabet $\Sigma = \{0, 1\}$.

The empty string is a string with zero occurrences of symbols. It is denoted by $\epsilon$, which may be chosen from any alphabet.

Length of a string is the number of symbols in that string. The standard notation for length of string $w$ is $|w|$.
Ex eg $|011| = 3$, $|\epsilon| = 0$

Power of an alphabet

If $\Sigma$ is any alphabet, we can express the set of all strings of a certain length from that alphabet by using an exponential notation.
$\Sigma^k$ is defined to be the set of strings of length $k$, each of whose symbols is in $\Sigma$.
Note: $\Sigma^0 = \epsilon$.
Ie $\epsilon$ is the only string of length 0.
If $\Sigma = \{a, b, c\}$
$\Sigma^1 = \{a, b, c\}$
$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
$\Sigma^3 = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc\}$ etc.
Note the confusion between $\Sigma$ and $\Sigma^1$.
$\Sigma$ is an alphabet with members a, b and c as symbols
$\Sigma^1$ is a set of strings with members as the strings a, b and c of length 1.

The set of all strings over an alphabet $\Sigma$ is denoted $\Sigma^*$. For instance,
$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ ie
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$





The set of non-empty strings from alphabet $\Sigma$ is denoted $\Sigma^+$, hence
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
- $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$

Concatenation of strings

Let $x$ and $y$ be strings. Then $xy$ denotes the concatenation of $x$ and $y$. i.e the string formed by making a copy of $x$ and following it by a copy of $y$.

If $x$ is the string composed of $i$ symbols, $x = a_1 a_2 \ldots a_i$ and $y$ is the string composed of $j$ symbols $y = b_1 b_2 \ldots b_j$, then $xy$ is the string of length $i+j$.

$xy = a_1 a_2 \ldots a_i b_1 b_2 \ldots b_j$.

Eg Let $x = 110$ $y = 001$
Then $xy = 110001$, $yx = 001110$.
For any string $w$, the eqns
$\epsilon w = w\epsilon = w$ hold.
ie $\epsilon$ is the identity for concatenation.

Languages

A set of strings all of which are chosen from some $\Sigma^*$, where $\Sigma$ is a particular alphabet, is called a language.
If $\Sigma$ is an alphabet and $L \subseteq \Sigma^*$ then $L$ is a language over $\Sigma$.
⊙ A language over $\Sigma$ need not include strings with all the symbols of $\Sigma$.

Platinum
Date: / /200

## Grammar:

A grammar G is a quadruple or 4-tuple
$G = (V, T, P, S)$ where
V → set of variables or non-terminals
T → set of terminals
P → set of productions
S → start symbol.

Each prod$^n$ is of the form $\alpha \to \beta$ where
$\alpha$ → non empty string of terminals and/or non-terminals
$\beta$ is string of terminals and/or non-terminals including the null string.
i.e. $\alpha$ is a string in $(V \cup T)^+$
$\beta$ is a string in $(V \cup T)^*$.
The gr. is also called phase structure grammar.

## Problems:

1) Let $\Sigma = \{a, b\}$. Obtain a grammar G generating set of all palindromes over $\Sigma$.

Soln
$V = \{S\}$
$T = \{a, b\}$
$P = \{ S \to a \mid b \mid \epsilon$
$S \to aSa \mid bSb$
$\}$
S is the start symbol.

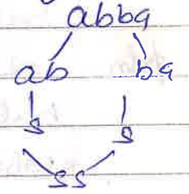
2) Obtain a grammar to generate a language of all non-palindromes over $\{a, b\}$

$V = \{S, A, B\}$
$T = \{a, b\}$
$P = \{ S \to aSa \mid bSb \mid A$
$A \to aBb \mid bBa$
$B \to aB \mid bB \mid \epsilon$
$\}$
S is the start symbol.

$P = \{ S \to aSb \mid bSa \mid abS \mid baS \mid \epsilon \}$

(OR)

3) Obtain a grammar to generate the language
$L = \{ w \mid n_a(w) = n_b(w) \}$




Soln
$V = \{S\}$
$T = \{a, b\}$
$P = \{ S \to aSb \mid bSa \mid SS \mid \epsilon \}$
S is the start symbol.

4) Obtain a gr. to generate the language
$L = \{ 0^n 1^{n+1} \mid n \geq 0 \}$

Soln
$V = \{S, A\}$
$T = \{0, 1\}$
$P = \{ S \to A1$
$A \to 0A1 \mid \epsilon \}$
S is the start symbol.

$0^n 1^n \Rightarrow 0A1 \mid \epsilon$.
$\Rightarrow 0A1 \mid \epsilon$
$\Rightarrow 00A11$

(OR)

$P = \{ S \to 0S1 \mid 1 \}$

5) What is the language generated by the gr.
$S \to 0A \mid \epsilon$
$A \to 1S$
$S \Rightarrow \epsilon$ (Null string)
$S \Rightarrow 0A$
$\Rightarrow 01S \Rightarrow 010A \Rightarrow 0101S \Rightarrow 01010A$
$\Rightarrow 0101010S \Rightarrow 01010101$
∴ The lang. generated by the gr. is
$L = \{ w \mid w \in (01)^* \}$ or
$L = \{ (01)^n \mid n \geq 0 \}$

Note: $S \to 01S \mid \epsilon$ and $S \to S01 \mid \epsilon$ } These gr to produce the same language.

Platinum
Date: / /200

6) Obtain a CFG to generate a string of balanced parantheses.

Soln The grammar G to generate a string of balanced parantheses is given by G=(V,T,P,S) where

V= {S}

T= {(, ), [, ], {, }}

P = { S → (S) | [S] | {S} | SS
S → ∈ }

S is the start symbol

7) Obtain a gr. to generate the language $L=\{ww^R \mid w \in \{a,b\}^*\}$ where $w^R$ is reverse of w.

Soln The gr. to find the reverse is given by G=(V,T,P,S) where

V= { S}

T= { a, b}

P= { S→aSa | bSb | ∈ }

S is the start symbol

Note: For the language $wcw^R$

P = { S → aSa | bSb | c

8) Obtain the gr. to generate the language $L=\{0^n 1^m 2^n \mid m \geq 1 \text{ and } n \geq 0\}$

Soln If n=0, there will be m no of 0's & 1's

A → 01 | 0A1

Otherwise when n>0,

S → A | S2

Thus the gr. can be written as

G=(V,T,P,S) where

V= {S, A}

T= {0,1,2}

P = { S → A | S2
A → 01 | 0A1 }

S is the start symbol.

9) Obtain a gr. to generate the lang $L=\{0^n 1^{2n} \mid n \geq 0\}$

Soln
V= {S}
T= {0,1}
P = { S → 0S11 | ∈ }
S is the start symbol.

| ∈ |
|---|
| 011 |
| 001111 |
| 000111111 |

10) Obtain a gr. to generate the language $L=\{a^{n+2} b^m \mid n \geq 0 \text{ and } m > n\}$

Soln: The strings can be represented by the language

L = { aabb*, aaabbb*, aaaabbbb*, - - - }

ie min no of b's is 1 less than no of a's.

In each string,

- No of b's can be one less than no of a's.
- No of b's may be equal to total no of a's.
- No of b's may be greater than total no of a's.

V= {S, A, B}

T= {a, b}

P= { S → aAB
A → aAb | ab
B → bB | ∈ }

S is the start symbol.

Derivation

S ⇒ aAB ⇒ aaAbB ⇒ aaabbB ⇒ aaabbb

S ⇒ aAB ⇒ aabB ⇒ aabb

S ⇒ aAB ⇒ aaAbB ⇒ aaabbbB ⇒ aaabbbbB ⇒ aaabbbbb

Platinum
Date: / /200

11) Obtain a gr. to generate the language
$L = \{a^n b^m \mid n \geq 0, m > n\}$

Soln The set of strings that can be generated by this language
$L = \{bb^*, abbb^*, aabbbb^*, ----\}$
To generate equal no of a's & b's
$A \rightarrow aAb \mid ab \mid \epsilon$
To generate one or more b's
$B \rightarrow bB \mid b$
Always a's precede b's. So
$S \rightarrow AB$
So, the gr. to generate the given lang. is
$V = \{S, A, B\}$
$T = \{a, b\}$
$P = \{ S \rightarrow AB$
$A \rightarrow aAb \mid ab \mid \epsilon$
$B \rightarrow bB \mid b$
$\}$
S is the start symbol.

12) Obtain a gr. to generate the language
$L = \{a^n b^{n-3} \mid n \geq 3\}$
Soln $L = \{aaa, aaaab, aaaaabb, ----\}$
The gr. $G = (V, T, P, S)$ is
$V = \{S, A\}$
$T = \{a, b\}$
$P = \{ S \rightarrow aaaA$
$A \rightarrow aAb \mid ab \mid \epsilon$
$\}$
S is the start symbol.

13) Obtain the gr. for the language
$L = L_1 L_2$ where
$L_1 = \{a^n b^m \mid n \geq 0, m > n\}$
$L_2 = \{0^n 1^{2n} \mid n \geq 0\}$
The gr for $L_1$ is
$V = \{S_1, A, B\}$
$T = \{a, b\}$
$P = \{ S_1 \rightarrow AB$
$A \rightarrow aAb \mid ab \mid \epsilon$
$B \rightarrow bB \mid b$
$\}$
$S_1$ is the start symbol.
The gr for $L_2$ is given by
$V = \{S_2\}$
$T = \{0, 1\}$
$P = \{ S_2 \rightarrow 0S_2 11 \mid \epsilon \}$
$S_2$ is the start symbol.
The resulting gr $L = L_1 L_2$ can be obtained by concatenating $S_1$ with $S_2$. i.e
$S \rightarrow S_1 S_2$
where S is the start symbol of resultant gr.
The final gr. which accepts $L = L_1 L_2$ is
$V = \{S, S_1, S_2, A, B\}$
$T = \{0, 1, a, b\}$
$P = \{$
$S \rightarrow S_1 S_2$
$S_1 \rightarrow AB$
$S_2 \rightarrow 0S_2 11 \mid \epsilon$
$A \rightarrow aAb \mid ab \mid \epsilon$
$B \rightarrow bB \mid b$
$\}$
S is the start symbol.

Platinum
Date: / /200

14) Obtain a gr. to generate the language
$L = L_1 \cup L_2$ where
$L_1 = \{a^n b^m \mid n \geq 0, m > n\}$
$L_2 = \{0^n 1^{2n} \mid n \geq 0\}$

Soln The start symbol is either $S_1$ or $S_2$.
$S \to S_1 \mid S_2$
The final gr is
$V = \{S, S_1, S_2, A, B\}$
$T = \{a, b, 0, 1\}$
$P = \{ S \to S_1 \mid S_2$
$S_1 \to AB$
$S_2 \to 0S_2 11 \mid \epsilon$
$A \to aAb \mid ab \mid \epsilon$
$B \to bB \mid b$
$\}$
S is the start symbol

15) Obtain a gr. to generate the language
$L = \{w : |w| \bmod 3 = 0\}$ on $\Sigma = \{a\}$

Soln The lang. accepted by the gr. can be written as
$L = \{\epsilon, aaa, aaaaaa, aaaaaaaaa, \ldots\}$
∴ The gr is G is
$V = \{S\}$
$T = \{a\}$
$P = \{ S \to aaaS \mid \epsilon \}$
S is the start symbol.

16) Obtain a gr. to generate the language
$L = \{w : |w| \bmod 3 = 0\}$ on $\Sigma = \{a, b\}$

Soln The gr is $G = (V, T, P, S)$ where
$V = \{S, A\}$
$T = \{a, b\}$

Platinum
Date: / /200

$P = \{ S \to AAAS \mid \epsilon$
$A \to a \mid b$
$\}$
S is the start symbol.

17) Obtain a gr. to generate the language
$L = \{w : |w| \bmod 3 > 0\}$ on $\Sigma = \{a\}$

Soln The gr is
$V = \{S, A, B\}$
$T = \{a\}$
$P = \{ S \to aA$
$A \to aB \mid \epsilon$ (Accepts w of $|w| = 1, 4, 7, \ldots$)
$B \to aS \mid \epsilon$ (Accepts w of $|w| = 2, 5, 8, \ldots$)
$\}$
S is the start symbol.
$L = \{a, aa, aaaa, aaaaa, \ldots\}$

Soln
$S \Rightarrow aA \Rightarrow aaB \Rightarrow aaaS \Rightarrow aaaaA \Rightarrow aaaa$
$S \Rightarrow aA \Rightarrow aaB \Rightarrow aa$
$S \Rightarrow aA \Rightarrow a$
$S \Rightarrow aA \Rightarrow aaB \Rightarrow aaaS \Rightarrow aaaaA \Rightarrow aaaaaB \Rightarrow aaaaa$

18) Obtain a gr to generate the language
$L = \{w : |w| \bmod 3 \neq |w| \bmod 2\}$ on $\Sigma = \{a\}$

Soln The above lang can be written as
$L = \{aa, aaaa, aaaa, aaaaaaaa, \ldots\}$
or
$L = \{w : w \in a^*$ and $|w| = 2i$ or $3i$ or $4i$ or $5i$ for $i = 1, 4, 7, 10, 13, \ldots\}$

| $\|w\| \bmod 3$ | $\|w\| \bmod 2$ |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 4 | 5 |
| 5 | 7 |
| 7 | 9 |
| 8 | 11 |
| 10 | 13 |
| 11 | 15 |
| 13 | 17 |
| 14 | |
| 16 | |

Platinum
Date: / /200

$V = \{S, A, B, C, D\}$
$T = \{a\}$
$P = \{$ $S \to aaA$
$A \to aB/\epsilon$ (Accepts w of $|w| = 2, 8, \ldots$)
$B \to ac/\epsilon$ ( —"— = 3, 9, ...)
$C \to aD/\epsilon$ ( —"— = 4, 10, ...)
$D \to as/\epsilon$ ( —"— = 5, 11, ...)
$\}$

S is the start symbol.

Platinum
Date: / /200

20) Obtain a gr. to generate the set of all strings with exactly one a when $\Sigma = \{a, b\}$

Soln $V = \{S, A\}$
$T = \{a, b\}$
$P = \{ S \to aA \mid bS$
$A \to bA \mid \epsilon \}$
S is the start symbol.

a
ab, ba
aabbbb
bbba
babbb

$S \Rightarrow aA$
$\Rightarrow abA$
$\Rightarrow abbA$
$\Rightarrow abb$

$S \Rightarrow bS$
$\Rightarrow baA$
$\Rightarrow babA$
$\Rightarrow bab$

21) Obtain a gr. to generate the set of all strings with atleast one a when $\Sigma = \{a, b\}$

Soln $V = \{S, A\}$
$T = \{a, b\}$
$P = \{ S \to bS/aA$
$A \to aA/bA/\epsilon$
$\}$

S is the start symbol.

22) Obtain a gr. to generate the set of all strings with no more than three a's when $\Sigma = \{a, b\}$

Soln: $V = \{S, A, B, C\}$
$T = \{a, b\}$
$P = \{ S \to bS/aA/\epsilon$
$A \to bA/aB/\epsilon$
$B \to bB/ac/\epsilon$
$C \to bc/\epsilon$
$\}$
S is the start symbol.

Platinum
Date: / /200

23) Obtain the gr. to generate the language
$L = \{w \mid n_a(w) = n_b(w)+1\}$

Sol^n One extra 'a' either in the beginning or at end can be got by
$S \to AaA.$

'A' generates equal no of a's & b's using the prods,
$A \to aAb$
$A \to bAa$
$A \to AA$
$A \to \epsilon$

The final gr. to generate the given language is $G=(V,T,P,S)$ where
$V = \{S, A\}$
$T = \{a, b\}$
$P = \{ S \to AaA$
$A \to aAb \mid bAa \mid AA \mid \epsilon$
$\}$
S is the start symbol.

24) Obtain the gr. to generate the language,
$L = \{w \mid n_a(w) > n_b(w)\}$

Sol^n More no of a's can appear either in the beginning or at the middle or at the end. So,
$S \to AB \mid BA \mid ABA.$
$B \to aB \mid a$
Final gr is
$V = \{S, A, B\}$
$T = \{a, b\}$
$P = \{ S \to AB \mid BA \mid ABA$
$A \to aAb \mid bAa \mid AA \mid \epsilon$
$B \to aB \mid a \}$
S is the start symbol.

Platinum
Date: / /200

# DETERMINISTIC FINITE AUTOMATION (DFA)

## Definition

The term "deterministic" refers to the fact that on each input there is one and only one state to which the automation can transition from its current state.

A deterministic finite automation consists of:

1) A finite set of states, often denoted Q.
2) A finite set of input symbols, often denoted $\Sigma$.
3) A transition function that takes as arguments a state and an input symbol and returns a state. The transition function will commonly be denoted $\delta$.
   Eg: If q is a state, and a is an input symbol then $\delta(q,a)$ is that state p such that there is an arc labeled a from q to p.
4) A start state, one of the states in Q.
5) A set of final states or accepting states F. The set F is a subset of Q.

A DFA is a "five-tuple" or quintuple

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

where A is the name of DFA,
Q is the set of states
$\Sigma$ its input symbols
$\delta$ its transition function
$q_0$ its start state
F its set of accepting states.

Platinum
Date: / /200

## How a DFA processes Strings

The "language" of the DFA is the set of all strings that the DFA accepts.

Suppose $a_1 a_2 \cdots a_n$ is a sequence of input symbols. We start out with the DFA in its start state, $q_0$. We consult the transition function $\delta$, say $\delta(q_0, a_1) = q_1$ to find the state that the DFA A enters after processing the first input symbol $a_1$. We process the next i/p symbol, $a_2$, by evaluating $\delta(q_1, a_2)$; let us suppose this state is $q_2$. We continue in this manner finding states $q_3, q_4, \cdots, q_n$ such that $\delta(q_{i-1}, a_i) = q_i$ for each $i$. If $q_n$ is a member of F, then the input $a_1 a_2 \cdots a_n$ is accepted. and if not then it is "rejected".

## Transition Diagrams

A "-" for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:

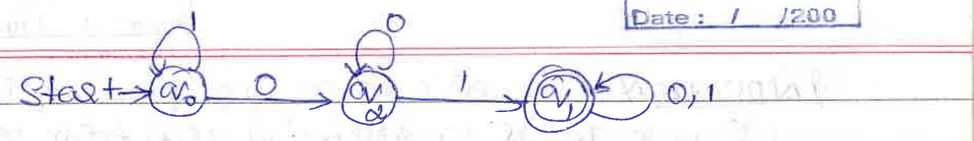
a) For each state in Q there is a node.

b) For each state q in Q and each i/p symbol a in $\Sigma$, let $\delta(q, a) = p$. Then the tr. diagram has an arc from node q to node P, labeled a. If there are several i/p symbols that cause transitions from q to p, then the TD can have one arc labeled by the list of these symbols.

c) There is an arrow into the start state $q_0$, labeled start. This arrow does not originate at any node.

d) Nodes corresponding to accepting states (that in F) are marked by a double circle. States not in F

Platinum
Date: / /200




## Transition Tables (TT):

A TT is a conventional, tabular representation of a function like $\delta$ that takes two arguments and returns a value. The rows of the table correspond to the states, and the columns correspond to the inputs. The entry for the row corresponding to state q and the column corresponding to input a is the state $\delta(q, a)$.

Eg

| | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_2$ | $q_0$ |
| $* q_1$ | $q_1$ | $q_1$ |
| $q_2$ | $q_2$ | $q_1$ |

## Extending the Transition function to Strings

If $\delta$ is the tr. fn, then the extended tr fn constructed from $\delta$ will be called $\hat{\delta}$. The ext tr fn is a fn that takes a state q and a string w and returns a state P - the state that the automaton reaches when starting in state q and processing the sequence of i/ps w. We define $\hat{\delta}$ by induction on the length of the i/p string, as follows:

BASIS: $\hat{\delta}(q, \epsilon) = q$. That is, if we are in state q and read no i/ps, then we are still in state q.

P.T.O

INDUCTION:

Suppose w is a string of the form xa; i.e, a is the last symbol of w, and x is the string consisting of all but the last symbol. For ex:, w = 1101 is broken into x = 110 & a = 1. Then

$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$ — (1)

To compute $\hat{\delta}(q, w)$ first compute $\hat{\delta}(q, x)$, the state that the automation is in after processing all but the last symbol of w.

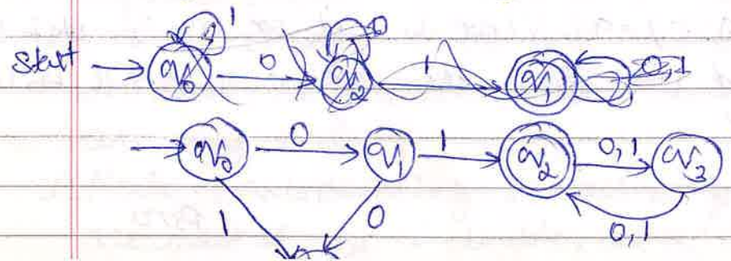
Suppose this state is P, i.e $\hat{\delta}(q, x) = P$. Then $\hat{\delta}(q, w)$ is what we get by making a transition from state P on input a, the last symbol of w. i.e $\delta(q, w) = \delta(P, a)$.

Properties of Transition functions

$\delta(q, \epsilon) = \delta^*(q, \epsilon) = q$
$\delta(q, wa) = \delta(\delta^*(q, w), a)$
$\delta(q, aw) = \delta^*(\delta(q, a), w)$

Ex $w = 01101$ (TT - prev page) - (Ullman)

$\delta(q_0, \epsilon) = q_0$
$\delta(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = \delta(q_0, 0) = q_1$
$\delta(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_1, 1) = q_2$
$\delta(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1) = \delta(q_2, 1) = q_3$
$\delta(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1) = \delta(q_3, 1) = q_2$
$\delta(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0) = \delta(q_2, 0) = q_3$
$\delta(q_0, 01101) = \delta(\hat{\delta}(q_0, 0110), 1) = \delta(q_3, 1) = q_2$








The language of a DFA

The lang. of a DFA $A = (Q, \Sigma, \delta, q_0, F)$ denoted by $L(A)$ is defined by

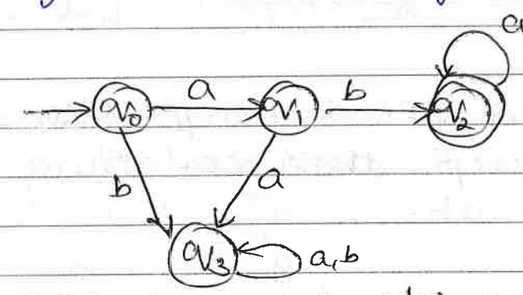
$L(A) = \{w \mid \hat{\delta}(q_0, w) \text{ is in } F\}$

i.e

The lang. of A is the set of strings w that take the start state $q_0$ to one of the accepting states. If L is L(A) for some DFA A, then we say L is a regular language.

Problems

1) Obtain a DFA to accept strings of a's and b's starting with the string ab.

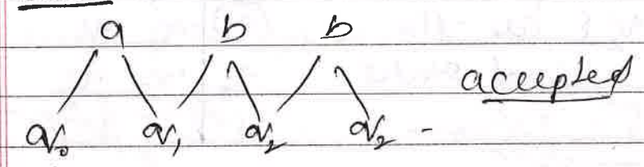



The DFA to accept strings starting with ab is

$A = (Q, \Sigma, \delta, q_0, F)$ where
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$q_0$ is the start state
$q_2$ is the final state

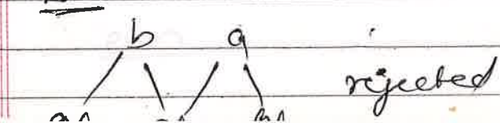
| $\delta$ | a | b |
|---|---|---|
| → $q_0$ | $q_1$ | $q_3$ |
| $q_1$ | $q_3$ | $q_2$ |
| * $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_3$ | $q_3$ |

abb




accepted

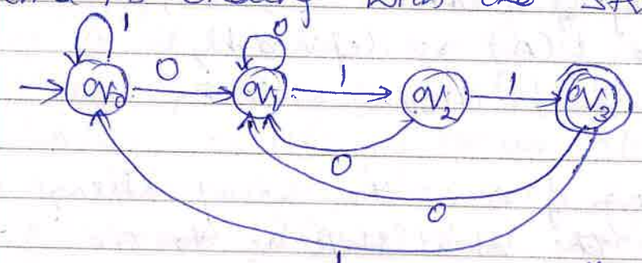
ba




rejected

Platinum
Date: / /200

2) Design a DFA to accept string of 0's and 1's ending with the string 011.




011 ✓
10011 ✓
0110 ✗
011011 ✓
01101 ✗
111011

$L = \{(0+1)^*011\}$

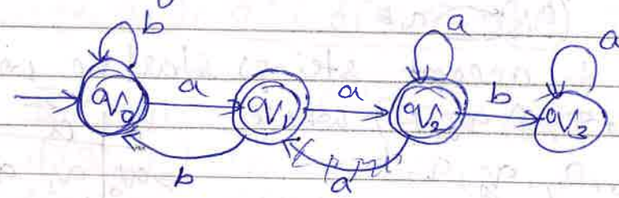
The DFA $A = (Q, \Sigma, \delta, q_0, F)$ is

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$q_0$ is the start state
F is the final state

| | 0 | 1 |
|---|---|---|
| →$q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_1$ | $q_3$ |
| ($q_3$) | $q_1$ | $q_0$ |

3) Obtain a DFA to accept strings of a's and b's except those containing the substring aab.

Soln



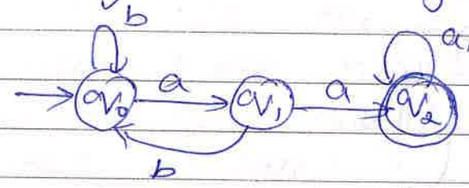

aab ✗
a ✓
b ✓
aba ✓
aaab

The DFA $A = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$q_0$ is the start state
$\{q_0, q_1, q_2\}$ are the final states.

| | a | b |
|---|---|---|
| →$q_0$ | $q_1$ | $q_0$ |
| ($q_1$) | $q_2$ | $q_0$ |
| ($q_2$) | $q_2$ | $q_3$ |
| $q_3$ | $q_3$ | $q_3$ |

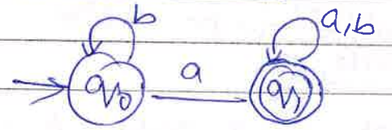
4) Obtain a DFA to accept strings of a's & b's having a substring aa.



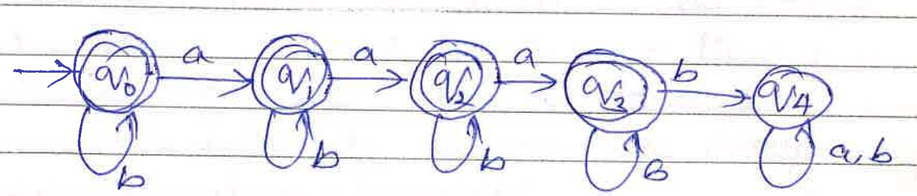

5) Obtain DFA's to accept strings of a's & b's having i) exactly one a
ii) atleast one a
iii) not more than three a's

i) exactly one a

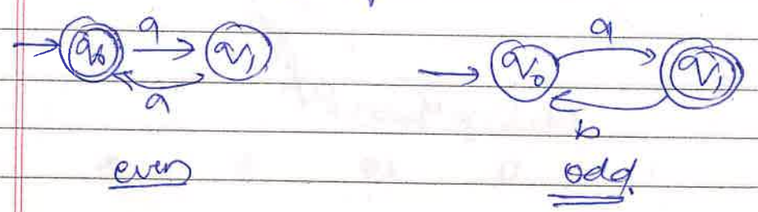

ii) atleast one a



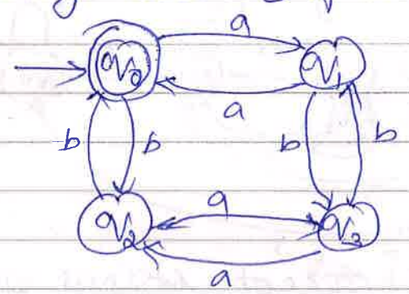

iii) not more than 3 a's




6) Obtain a DFA to accept even no of a's and odd no of a's.

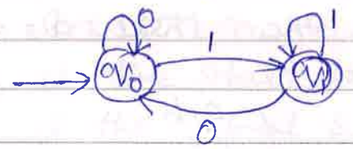


7) Obtain a DFA to accept strings of a's & b's having even no of a's and b's.

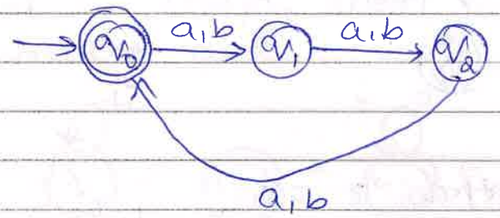



8) Obtain a DFA to accept binary odd nos

001
011
101




9) Obtain a DFA to accept the language $L = \{w : |w| \bmod 3 = 0\}$ on $q, \Sigma = \{a, b\}$

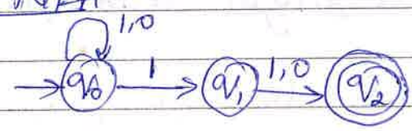



# NONDETERMINISTIC FINITE AUTOMATA (NFA)

An NFA has the power to be in several states at once. This ability is often expressed as an ability to "guess" something about its input.

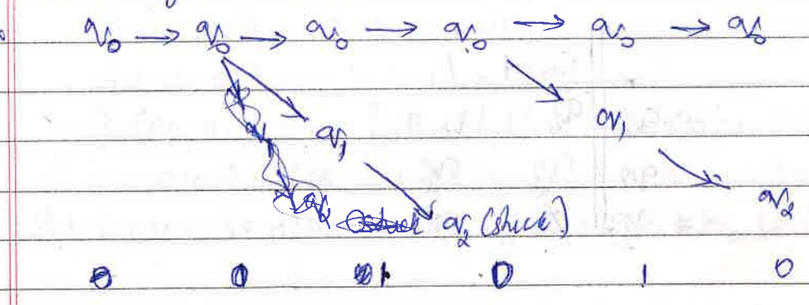
For instance, when the automaton is used to search for certain sequences of characters (eg: keywords) in a long text string, it is helpful to "guess" that we are at the beginning of one of those strings and use a sequence of states to do nothing but check that the string appears, character by character.

An NFA




Note that now there are 2 possible transitions labeled 1 out of $q_0$ & hence NFA has 2 options - it can move to either $q_0$ or $q_1$. In fact at each such point, it starts a new thread corresponding to each of the new transitions. Also note that there are no transitions out of $q_2$. Whenever this happens, the corresponding thread just ceases to exist.

Ex



Platinum
Date: / /200

## Defination of NFA

An NFA is represented like a DFA:

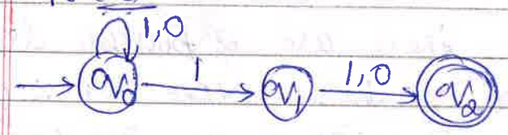
$A = (Q, \Sigma, \delta, q_0, F)$

Where

1) $Q$ is a finite set of states
2) $\Sigma$ is a finite set of input symbols
3) $q_0$, a member of $Q$, is the start state
4) $F$, a subset of $Q$, is the set of final (or accepting) states.
5) $\delta$, the transition function is a function that takes a state in $Q$ and an input symbol in $\Sigma$ as arguments and returns a subset of $Q$.

Note: $\delta$ returns a set of states in the case of an NFA and a single state in the case of a DFA.

For ex




$A = (Q, \Sigma, \delta, q_0, F)$ where
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$q_0$ is the start state
$q_2$ is the final state

| | 0 | 1 |
|---|---|---|
| $\to q_0$ | $q_0$ | $\{q_0, q_1\}$ |
| $q_1$ | $q_2$ | $q_2$ |
| $* q_2$ | $\emptyset$ | $\emptyset$ |

Platinum
Date: / /200

## The Extended Transition function

We extend the tr. fn $\delta$ of an NFA to a function $\hat{\delta}$ that takes a state $q$ and a string of input symbols $w$, and returns the set of states that the NFA is in if it starts in state $q$ and processes the string $w$.

Formally, we define $\hat{\delta}$ for an NFA's tr. fn $\delta$ by:

BASIS:

$\hat{\delta}(q, \epsilon) = \{q\}$. That is, without reading any input symbols, we are only in the state we began in.

INDUCTION:

Suppose $w$ is of the form $w = xa$, where $a$ is the final symbol of $w$ and $x$ is the rest of $w$. Also suppose that $\hat{\delta}(q, x) = \{p_1, p_2 \cdots p_k\}$.

Let $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2 \cdots r_m\}$

Then $\hat{\delta}(q, w) = \{r_1, r_2, \cdots r_m\}$

In formally, we compute $\hat{\delta}(q, w)$ by first computing $\hat{\delta}(q, x)$, and by then following any transition from any of these states that is labeled $a$.

Ex Process the string 01010 using $\hat{\delta}$ by NFA.

$\hat{\delta}(q_0, \epsilon) = \{q_0\}$
$\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0\}$
$\hat{\delta}(q_0, 01) = \delta(q_0, 1) = \{q_0, q_1\}$
$\hat{\delta}(q_0, 010) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
$\hat{\delta}(q_0, 0101) = \delta(q_0, 1) \cup \delta(q_2, 1) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
$\hat{\delta}(q_0, 01010) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

Platinum
Date : / /200

## The Language of an NFA

If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA, then,
$L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \phi\}$
That is $L(A)$ is the set of strings $w$ in $\Sigma^*$ such that $\hat{\delta}(q_0, w)$ contains atleast one accepting state.

## EQUIVALENCE OF DFA & NFA OR CONVERSION FROM NFA to DFA

## SUBSET CONSTRUCTION METHOD

The subset construction starts from an NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Its goal is the description of a DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(D) = L(N)$. The i/p alphabets of the two automata are the same, and the start state of D is the set containing only the start state of N. The other components of D are constructed as follows.

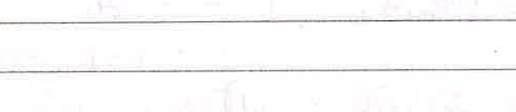
- $Q_D$ is the set of subsets of $Q_N$. ie $Q_D$ is the powerset of $Q_N$. If $Q_N$ has n states, then $Q_D$ will have $2^n$ states. Often, not all these states are accessible from the start state of $Q_D$. Inaccessible states can be "thrown away", so effectively, the no of states of D may be much smaller than $2^n$.

- $F_D$ is the set of subsets S of $Q_N$ such that $S \cap F_N \neq \phi$. That is $F_D$ is all sets of N's states that include atleast one accepting state of N.

- For each set $S \subseteq Q_N$ and for each input symbol a in $\Sigma$.

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a).$$

That is, to compute $\delta_D(S, a)$, we look at all the states p in S, see what states N goes to from p on input a, and take the union of all those states.

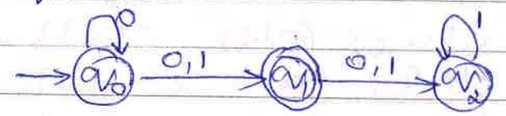



| $\delta$ | 0 | 1 |
|---|---|---|
| $\phi$ | $\phi$ | $\phi$ |
| $\rightarrow \{q_0\}$ | $\{q_0\}$ | $\{q_0, q_1\}$ |
| $\{q_1\}$ | $\{q_2\}$ | $\{q_2\}$ |
| $*\{q_2\}$ | $\phi$ | $\phi$ |
| $\{q_0, q_1\}$ | $\{q_0, q_2\}$ | $\{q_0, q_1, q_2\}$ |
| $*\{q_0, q_2\}$ | $\{q_0\}$ | $\{q_0, q_1\}$ |
| $*\{q_1, q_2\}$ | $\{q_2\}$ | $\{q_2\}$ |
| $*\{q_0, q_1, q_2\}$ | $\{q_0, q_2\}$ | $\{q_0, q_1, q_2\}$ |

Platinum
Date: / /200

**Problems:**

1) Convert the following NFA into an equivalent DFA.




Step 1: $q_0$ is the start of DFA.

& $Q_D = \{[q_0]\}$ --- (1)

Step 2 Find the new states from each state in $Q_D$ and obtain the corresponding transitions

Consider the state $[q_0]$:

When a=0,

$\delta_D([q_0], 0) = \delta_N([q_0], 0)$

$= [q_0, q_1]$ — (2)

When a=1,

$\delta_D([q_0], 1) = \delta_N([q_0], 1)$

$= [q_1]$ --- (3)

Since the states obtained in (2) & (3) are not in $Q_D$ (1), add these two states to $Q_D$ so that

$Q_D = \{[q_0], [q_0, q_1], [q_1]\}$

Consider the state $[q_0, q_1]$:

When a=0,

$\delta_D([q_0, q_1], 0) = \delta_N([q_0, q_1], 0)$

$= \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \{q_2\}$

$= [q_0, q_1, q_2]$ --- (4)

When a=1

$\delta_D([q_0, q_1], 1) = \delta_N([q_0, q_1], 1)$

$= \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

$= \{q_1\} \cup \{q_2\}$

$= [q_1, q_2]$ --- (5)

Platinum
Date: / /200

Since these 2 states are not in $Q_D$ add them to $Q_D$ so that

$Q_D = \{[q_0], [q_0, q_1], [q_1], [q_0, q_1, q_2], [q_1, q_2]\}$

The transitions are

| $\delta$ | 0 | 1 |
|---|---|---|
| $[q_0]$ | $[q_0, q_1]$ | $[q_1]$ |
| $[q_0, q_1]$ | $[q_0, q_1, q_2]$ | $[q_1, q_2]$ |
| $[q_1]$ | | |
| $[q_0, q_1, q_2]$ | | |
| $[q_1, q_2]$ | | |

Consider the state $[q_1]$:

When a=0

$\delta_D([q_1], 0) = \delta_N([q_1], 0)$

$= [q_2]$ --- (6)

When a=1

$\delta_D([q_1], 1) = \delta_N([q_1], 1)$

$= [q_2]$ --- (7)

Since $[q_2]$ is not present in $Q_D$ add it to it.

$Q_D = \{[q_0], [q_0, q_1], [q_1], [q_0, q_1, q_2], [q_1, q_2], [q_2]\}$

Consider the state $[q_0, q_1, q_2]$:

When a=0,

$\delta_D([q_0, q_1, q_2], 0) = \delta_N([q_0, q_1, q_2], 0)$

$= \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_1\} \cup \{q_2\} \cup \emptyset$

$= [q_0, q_1, q_2]$

When a=1,

$\delta_D([q_0, q_1, q_2], 1) = \delta_N([q_0, q_1, q_2], 1)$

$= \delta_N(q_0, 1) \cup \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= \{q_1\} \cup \{q_2\} \cup \{q_2\}$

$= [q_1, q_2]$

Since these two states are already in $Q_D$ do not add them to $Q_D$. But add the corresponding Tr. to tr. table.

Consider the state $[q_1, q_2]$

When $a=0$,

$\delta_D([q_1,q_2],0) = \delta_N([q_1,q_2],0)$

$= \delta_N[q_1,0), \delta_N(q_2,0)$

$= \{q_2\} \cup \phi$

$= [q_2]$

When $a=1$

$\delta_D([q_1,q_2],1) = \delta_N([q_1,q_2],1)$

$= \delta_N(q_1,1) \cup \delta_N(q_2,1)$

$= \{q_2\} \cup \{q_2\}$

$= [q_2]$

Since $[q_2]$ is already in $Q_D$, just add the tr to TT.

Consider the state $[q_2]$:

When $a=0$,

$\delta_D([q_2],0) = \delta_N([q_2],0)$

$= \phi$

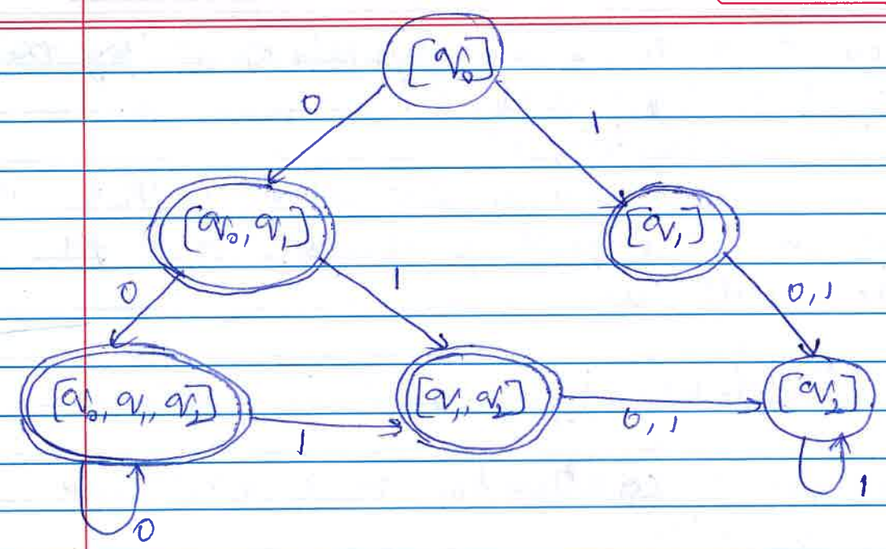
When $a=1$

$\delta_D[q_2],1) = \delta_N([q_2],1)$

$= [q_2]$

The final TT. is

| $\delta$ | 0 | 1 |
|---|---|---|
| $[q_0]$ | $[q_0,q_1]$ | $[q_1]$ |
| $[q_0,q_1]$ | $[q_0,q_1,q_2]$ | $[q_1,q_2]$ |
| $[q_1]$ | $[q_2]$ | $[q_2]$ |
| $[q_0,q_1,q_2]$ | $[q_0,q_1,q_2]$ | $[q_1,q_2]$ |
| $[q_1,q_2]$ | $[q_2]$ | $[q_2]$ |
| $[q_2]$ | $\phi$ | $[q_2]$ |




**Theorem-1**

If $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ by the subset construction then $L(D) = L(N)$.

**Proof:** What we actually prove first, by induction on $|w|$, is that

$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$

Notice that each of the $\hat{\delta}$ functions refers a set of states from $Q_N$, but $\hat{\delta}_D$ interprets this set as one of the states of $Q_D$ (which is the power set of $Q_N$), while $\hat{\delta}_N$ interprets this set as a subset of $Q_N$.

**BASIS:** Let $|w| = 0$; that is $w = \epsilon$. By the basis definition of $\hat{\delta}$ for DFA's and NFA's both $\hat{\delta}_D(\{q_0\}, \epsilon)$ and $\hat{\delta}_N(q_0, \epsilon)$ are $\{q_0\}$.

**INDUCTION:**

Let $w$ be of length $n+1$, and assume the stmt for length $n$. Break $w$ up as ~~here~~, $w = xa$,

where a is the final symbol of w. By the inductive hypothesis,

$\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$.

Let both these sets of N's states be $\{P_1, P_2, \cdots P_k\}$

The inductive part of the defn of $\hat{\delta}$ for NFA's tells us

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(P_i, a) \quad ---(1)$$

The subset construction, on the other hand, tells us that

$$\delta_D(\{P_1, P_2, \cdots P_k\}, a) = \bigcup_{i=1}^{k} \delta_N(P_i, a) \quad ---(2)$$

Now, let us use (2) and the fact that $\hat{\delta}_D(\{q_0\}, x) = \{P_1, P_2, \cdots P_k\}$ in the inductive part of the defn of $\hat{\delta}$ for DFA's:

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a)$$
$$= \delta_D(\{P_1, P_2, \cdots P_k\}, a)$$
$$= \bigcup_{i=1}^{k} \delta_N(P_i, a) \quad ---(3)$$

Thus eqns (1) & (3) demonstrate that $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$. When we observe that D and N both accept w if and only if $\hat{\delta}_D(\{q_0\}, w)$ or $\hat{\delta}_N(q_0, w)$ respectively, contain a state in $F_N$, we have a complete proof that $L(D) = L(N)$.

**Theorem-2.**

A lang. L is accepted by some DFA iff L is accepted by some NFA.

**Proof:** (If) the "if" part is the subset construction & Thm-(1).







(Only if). In this part, we have only to convert a DFA into an identical NFA.

If we have the tr. diagram for a DFA, we can also interpret it as the tr. dig of an NFA, which happens to have exactly one choice of tr in any situation.

More formally, let $D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA. Define $N = (Q, \Sigma, \delta_N, q_0, F)$ to be the equivalent NFA, where $\delta_N$ is defined by the rule:

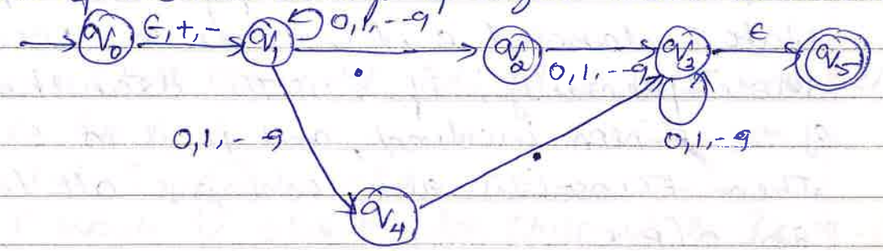
- If $\delta_D(q, a) = p$, then $\delta_N(q, a) = \{P\}$.

It is then easy to show by induction on |w|, that if $\hat{\delta}_D(q_0, w) = p$ then $\hat{\delta}_N(q_0, w) = \{P\}$.

w is accepted by D if and only if it is accepted by N i.e. $L(D) = L(N)$.

## Finite Automata with Epsilon-Transitions

Construct an ε-NFA that accepts decimal numbers consisting of
1) an optional + or − sign
2) A string of digits
3) A decimal point
4) Another string of digits. Either this string of digits or the string (2) can be empty, but atleast one of the two strings of digits must be nonempty.



Platinum
Date: / /200

## The Formal Notation for an ε-NFA:-

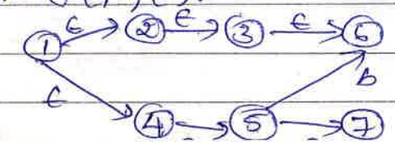
We represent an ε-NFA A by $A=(Q, \Sigma, \delta, q_0, F)$ where all components have their same interpretation as for an NFA, except that $\delta$ is now a function that takes as arguments:
1) A state in Q and
2) A member of $\Sigma \cup \{\epsilon\}$, that is either an input symbol, or the symbol ε. We require that ε, the symbol for the empty string, cannot be a member of the alphabet Σ.

## Epsilon-closures:-

We ε-close a state q by following all transitions out of q that are labeled ε. However, when we get to other states by following ε, we follow the ε-transitions out of those states & so on, eventually finding every state that can be reached from q along any path whose arcs are all labeled ε.

## Recursive Defn:-

BASIS: State q is in ECLOSE(q).
INDUCTION: If state p is in ECLOSE(q), and there is a transition from state p to state r labeled ε, then r is in ECLOSE(q). More precisely, if $\delta$ is the transition function of the ε-NFA involved, and p is in ECLOSE(q), then ECLOSE(q) also contains all the states in $\delta(P, \epsilon)$.

Ex 


ECLOSE(1) = {1, 2, 3, 4, 6}

Platinum
Date: / /200

## Eliminating ε-Transitions

Let the ε-NFA $E=(Q_E, \Sigma, \delta_E, q_0, F_E)$.
Then the equivalent DFA
$D=(Q_D, \Sigma, \delta_D, q_D, F_D)$
is defined as follows:

1) $Q_D$ is the set of subsets of $Q_E$. More precisely, we shall find that the only accessible states of D are the ε-closed subsets of $Q_E$, i.e. those sets $S \subseteq Q_E$ such that $S = \text{ECLOSE}(S)$. Put another way, the ε-closed sets of states S are those such that any ε-transitions out of one of the states in S lead to a state that is also in S. Note that ∅ is an ε-closed set.

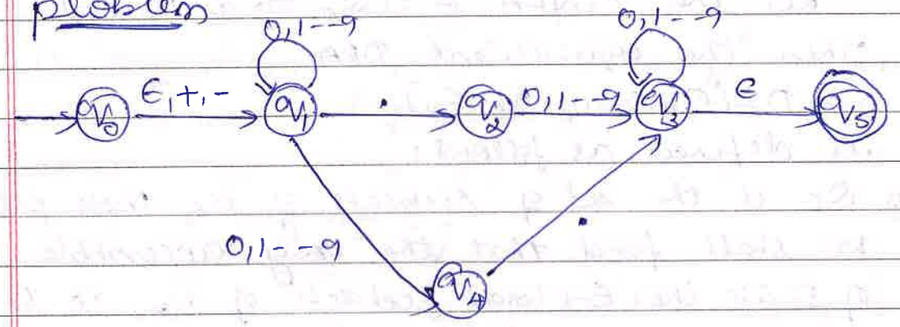
2) $q_D = \text{ECLOSE}(q_0)$; i.e. we get the start state of D by closing the set consisting of only the start state of E. Note that this rule differs from the original subset construction, where the start state of the constructed automaton was just the set containing the start state of the given NFA.

3) $F_D$ is those sets of states that contain atleast one accepting state of E. That is $F_D = \{S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \emptyset\}$.

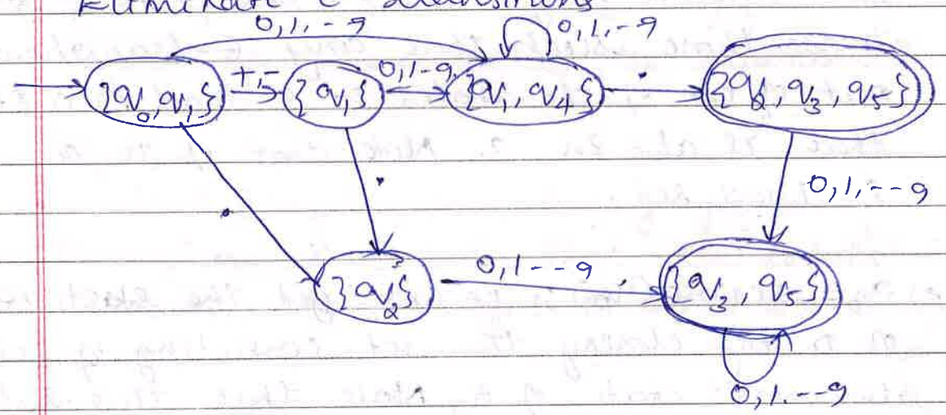
4) $\delta_D(S, a)$ is computed, for all a in Σ and sets S in $Q_D$ by:
(a) Let $S = \{P_1, P_2 \cdots P_k\}$
(b) Compute $\bigcup_{i=1}^{k} \delta(P_i, a)$; let this set be $\{r_1, r_2 \cdots r_m\}$
(c) Then $\delta_D(S, a) = \bigcup_{j=1}^{m} \text{ECLOSE}(r_j)$.

# REGULAR EXPRESSIONS

problem




Eliminate ε-transitions




$\delta_D(\{q_0,q_1\},+) = \{q_1\}$
$\delta_D(\{q_0,q_1\},-) = \{q_1\}$
$\delta_D(\{q_0,q_1\},\cdot) = \{q_2\}$
$\delta_D(\{q_0,q_1\},0) = \{q_1,q_4\}$
$\delta_D(\{q_0,q_1\},1) = \{q_1,q_4\}$
$\delta_D(q_2,0) = \{q_3,q_5\}$
$\delta_D(q_2,1) = \{q_3,q_5\}$
$\delta_D(q_3,0) = \{q_3,q_5\}$
$\delta_D(q_3,1) = \{q_3,q_5\}$
$\delta_D(q_4,\cdot) = \{q_3,q_5\}$
$\delta_D(q_1,\cdot) = \{q_2\}$
$\delta_D(q_1,0) = \{q_1\}$
$\delta_D(q_1,1) = \{q_1\}$

$\delta_D(\{q_1,q_4\},\cdot) = \delta_D(q_1,\cdot) \cup \delta_D(q_4,\cdot)$
$= \{q_2\} \cup \{q_3,q_5\}$
$= \{q_2,q_3,q_5\}$

$\delta_D(\{q_0,q_1\},0) = \delta_D(q_0,0) \cup \delta_D(q_1,0)$
$= \phi \cup \{q_1\}$
$\delta_D(\{q_1,q_4\},0) = \{q_1,q_4\}$
$\delta_D(\{q_1,q_4\},1) = \{q_1,q_4\}$
$\delta_D(\{q_3,q_5\},0) = \{q_3,q_5\}$

$\delta_D(\{q_2,q_3,q_5\},0) = \{q_3,q_5\}$

# REGULAR EXPRESSIONS

Regular exprns define the regular languages that various forms of automata describe.

R.E. offers a declarative way to express the strings we want to accept.

## The Operators of R.Es :-

1) The union of two languages L and M denoted L∪M, is the set of strings that are in either L or M or both.
For ex: if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$ then $L \cup M = \{\epsilon, 10, 001, 111\}$.

2) The concatenation of languages L and M is the set of strings that can be formed by taking any string in L and concatenating it with any string in M. denoted with a dot or with no operator.
For ex: if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$ then $L.M$ or $LM = \{001, 10, 111, 001001, 10001, 111001\}$

3) The closure (or star or kleene closure) of a lang L is denoted $L^*$ and represents the set of those strings that can be formed by taking any no of strings from L, possibly with repetitions and concatenating all of them
For ex: Let $L = \{0, 11\}$
$L^0 = \{\epsilon\}$
$L^1 = L$, represents choice of one string from L.
$L^2 = \{00, 011, 110, 1111\}$, pick two strings from L with repetitions allowed.
$L^3 = \{000, 0011, 0110, 1100, 01111, 11011, 11110, 111111\}$
To compute $L^*$, compute $L^i$ for each $i$ & take the union of all these languages. $L^i$ has $2^i$ members
L is finite here.

2) Let L be the set of all strings of 0's.
L is infinite.
$L^0 = \{\epsilon\}$ &
$L^1 = \{L.\ L^2 = L.L,\ L^3 = L...$
∴ Every string of 0's can be written as the concatenation of two strings of 0's.
∴ $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$ is L in this case.

3) $\phi^* = \{\epsilon\}$.
$\phi^0 = \{\epsilon\}$
$\phi^i$ for any $i \geq 1$ is empty.
$\phi$ is one of only two langs whose closure is finite.

Building Reg. Exprns
The algebra of R.Es uses constants & variables and operators for 3 operations viz union dot and star.
Recursive defn. of R.E's

BASIS: The basis consists of 3 parts:
1) The constants $\epsilon$ and $\phi$ are R.Es denoting the langs $\{\epsilon\}$ and $\phi$ resp^ly. that is $L(\epsilon) = \{\epsilon\}$ and $L(\phi) = \phi$.

2) If a is any symbol, then a is a R.E. This expn denotes the lang. {a}. ie $L(a) = \{a\}$

3) A var, usually capitalized & italic such as L is a var, representing any language.

INDUCTION:
1) If E & F are R.Es, then E+F is a RE denoting the union of L(E) and L(F). ie
$L(E+F) = L(E) \cup L(F)$.





2) If E and F are REs then EF is a RE denoting the concatenation of L(E) and L(F). ie
$L(EF) = L(E)L(F)$.

3) If E is a RE, then $E^*$ is a RE, denoting the closure of L(E).
ie $L(E^*) = (L(E))^*$

4) If E is a RE, then (E), a parenthesized E, is also a RE. denoting the same language.
Formally, $L((E)) = L(E)$.

Precedence of Reg. Expn Operators
1) The star operator is of highest precedence.
2) Concatenation or dot operator.
3) All unions (+ operators) with their operands.

Finite Automata & Regular Expressions:-
W.K.t DFA & the two kinds of NFA (with & without ε-transitions) accept the same class of languages. T.S.T RE define the same class, we must show that
1) Every lang. defined by one of these automata is also defined by a RE.
2) Every lang. defined by a RE is defined by one of these automata. For this we show that There is an NFA with ε-transitions accepting the same language.

Platinum Date: / /200

pbms 1) Obtain a RE to accept a lang. consisting of strings of a's and b's of even length.

Soln $(aa+ab+ba+bb)^*$ or

$L(R) = \{aa+ab+ba+bb)^n \mid n \geq 0\}$

2) Obtain a RE to accept a lang. consisting of strings of a's and b's of odd length.

$(a+b)(aa+ab+ba+bb)^*$

3) Obtain a re to accept strings of a's & b's ending with 'b' and has no substring aa.

$(b+ab)^+$

using set notation,

$L(R) = \{(b+ab)^n \mid n \geq 1\}$

4) Obtain a RE to accept strings of 0's & 1's having no two consecutive zeros.

$(1+01)^*(0+1)$

5) Obtain a RE such that $L(R) = \{w \mid w \in \{0,1\}^*\}$ with atleast three consecutive zeros.

Soln $(0+1)^* 000 (0+1)^*$ or

$L(R) = \{(0+1)^m 000 (0+1)^n \mid m \geq 0 \text{ and } n \geq 0\}$

6) Obtain a RE to accept strings of a's and b's of length $\leq 10$

$(\epsilon+a+b)^{10}$

7) Obtain a RE to accept strings of a's and b's starting with 'a' and ending with 'b'.

$a(a+b)^*b$

## CONVERTING REG. EXPRNS TO AUTOMATA

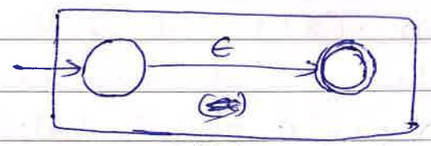
EMP

**Theorem**

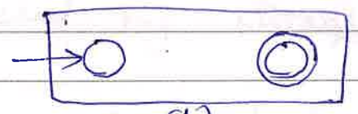
Every language defined by a RE is also defined by a finite automaton.

**Proof:** Suppose $L = L(R)$ for a RE. R. We show that $L = L(E)$ for some $\epsilon$-NFA E with:

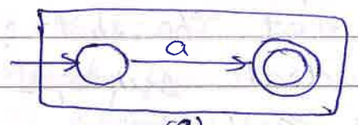
1) Exactly one accepting state,
2) No arcs into the initial state
3) No arcs out of the accepting state.




(a)



(b)




(c)

**BASIS:** There are three parts to the basis as shown in above fig.

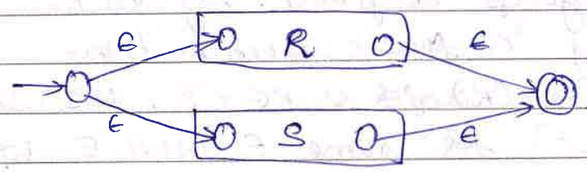
In part (a) we see how to handle the expr$^n$ $\epsilon$. The lang. of the automaton is easily seen to be $\{\epsilon\}$. Since the only path from the start state to an accepting state is labeled $\epsilon$.

Part (b) shows the construction for $\phi$. Since there are no paths from start state to accepting state, so $\phi$ is the lang. of this automaton.

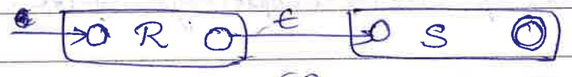
Part (c) gives the automaton for a RE a. The lang. of this automaton is $L(a)$, consists of one string a.

Platinum
Date: / /200

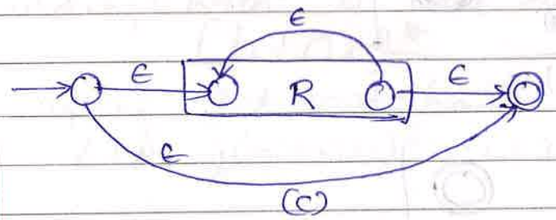
INDUCTION:-




(a)




(b)




(c)

The three parts of induction are shown above. We assume that the stmt of the thm is true for the immediate subexprns of a given RE. The four cases are:

1) The expr is R+S for some smaller exprns R and S. Then automaton (a) serves. Ie starting at the new start state, we can go to the start state of either the automaton for R or the automaton for S. We then reach the accepting state of one of these automata, following a path labeled by some string in L(R) or L(S), resply. From there, we follow one of the ϵ-arcs to the accepting state of the new automaton. Thus, the lang. of the automaton is L(R) ∪ L(S).

2) The expr. is RS for some smaller exprns R and S. The automaton is shown in (b). The start state of the first automaton becomes the start state of the whole, and the accepting state of the second automaton becomes the accepting state of the whole.
The only paths from start to accepting state go first through the automaton for R, where it must follow a path labeled by a string in L(R). and then through the automaton for S, where it follows a path labeled by a string in L(S). Thus the paths in the automaton of fig (b) are those labeled by strings in L(R)L(S).

3) The expr is $R^*$ for some smaller expr R. We use automaton (c). That allows us to go either:
(a) Directly from the start state to the accepting state along a path labeled ϵ. That path lets us accept ϵ, which is in $L(R^*)$ no matter what exprn R is.

(b) To the start state of the automaton for R, through that automaton one or more times and then to the accepting state. This set of paths allows us to accept strings in L(R), L(R)L(R) and so on thus covering all strings in $L(R^*)$ except ϵ.

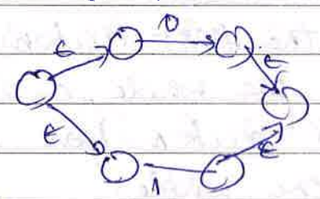
4) The expr is (R) for some smaller expr of R. The automaton for R also serves as the automaton for (R) since the parentheses do not change

Platinum
Date: / /200

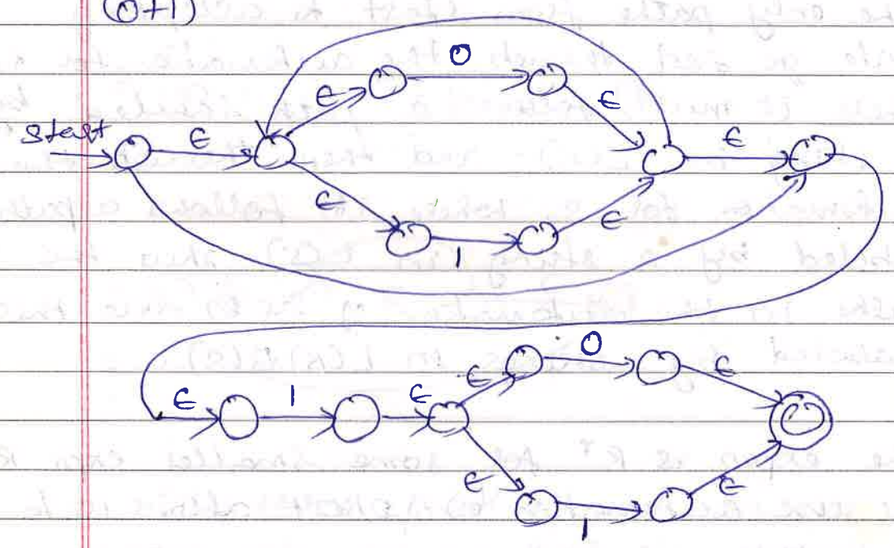
Prbm

1) Convert the RE $(0+1)^*1(0+1)$ to an $\epsilon$-NFA.

Soln Consider $(0+1)$

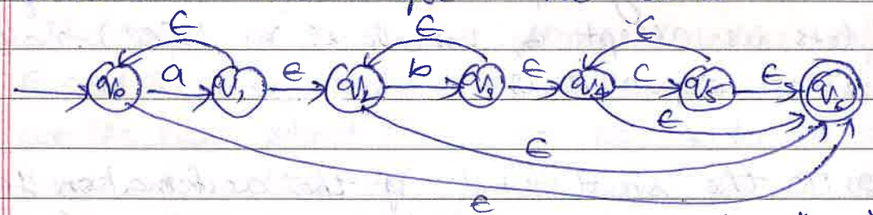



$(0+1)^*$

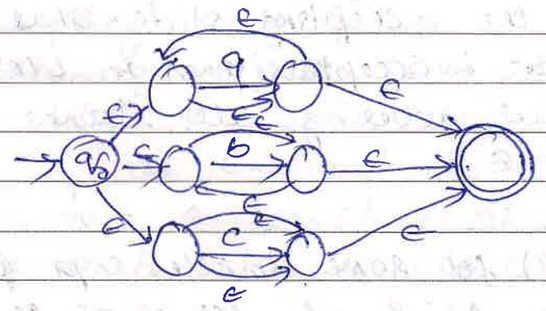



2) Obtain an NFA for the R.E $(a+b)^*aa(a+b)^*$

3) Obtain an NFA for the RE $a^*b^*c^*$




4) Obtain an NFA for the RE $a^*+b^*+c^*$




5) To accept strings of a's & b's starting with


Platinum
Date: / /200

Theorem: - Kleene's Theorem :-

If $L=L(A)$ for some DFA A, then there is a RE R such that $L=L(R)$.

Proof Let us suppose that A's states are $\{1, 2, \ldots, n\}$ for some integer n. We have to construct a collection of REs that describe progressively broader sets of paths in the DFA A.

Let us use $R_{ij}^{(k)}$ as the name of a RE whose lang. is the set of strings w such that w is the label of a path from state i to state j in A. and that path has no intermediate node whose number is $>k$. Note: The beginning & end pts of the path are not "intermediate." so there is no constraint that i and/or j be less than or equal to k.

To construct the expns $R_{ij}^{(k)}$, we use the following inductive defn starting at $k=0$ and finally reaching $k=n$. Notice that when $k=n$, there is no restriction at all on the paths represented, since there are no states greater than n.

BASIS: The basis is $k=0$. Since all states are numbered 1 or above, the restriction on paths is that the path must have no intermediate states at all. There are only two kinds of paths that meet such a condition:

1) An arc from node (state) i to node j.
2) A path of length 0 that consists of only some node i.

If $i \neq j$ then only case (c) is possible. We must examine the DFA A and find those input symbols a such that there is a transition from state i to state j on symbol a.

(a) If there is no such symbol a, then $R_{ij}^{(0)} = \phi$

(b) If there is exactly one such symbol a, then $R_{ij}^{(0)} = a$

(c) If there are symbols $a_1, a_2, \ldots a_k$ that label arcs from state i to state j, then $R_{ij}^{(0)} = a_1 + a_2 + \cdots + a_k$

However, if $i=j$ then the legal paths are the paths of length 0 and all loops from i to itself. The path of length 0 is represented by the RE $\epsilon$ since that path has no symbols along it. Thus we add $\epsilon$ to the various expns devised in (a) thro' (c) above.

INDUCTION:-

Suppose there is a path from state i to state j that goes through no state higher than k. There are two possible cases to consider:

1) The path does not go through state k at all. In this case, the label of the path is in the lang. of $R_{ij}^{(k-1)}$.

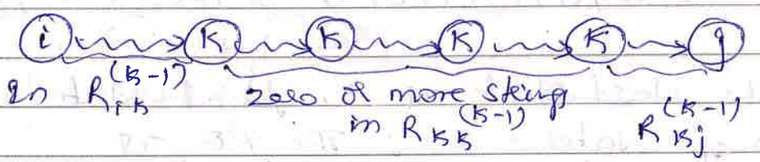
2) The path goes thro' state k atleast once. Then we can break the path into several pieces The first goes from state i to state k without passing through k, the last piece goes from k to j without passing thro' k, and all the pieces in the middle go from k to itself, without passing through k.

Note that if the path goes thro state k only once, then there are no "middle" pieces, just a path from i to k, and a path from k to j. The set of labels for all paths of this type is represented by a RE $R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$.

When we combine the expns for the paths of two types above, we have,

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

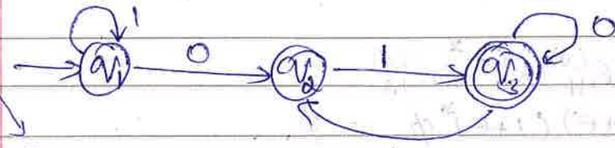
for the labels of all paths from state i to state j that go thro no state higher than k.



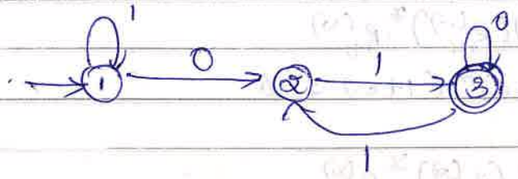

Ex: Convert the DFA to a RE.

The DFA accepts all strings corresponding to RE $1^* 01 (0+11)^*$.

However, lets use the form to identify the expn




Renamed as:



Platinum
Date: / /200

| Bases | | Indirection k=1 | | Indirection k=2 | |
|---|---|---|---|---|---|
| $R_{11}^{(0)}$ | $1+\epsilon$ | $R_{11}^{(1)}$ | $1^*$ | $R_{11}^{(2)}$ | $1^*$ |
| $R_{12}^{(0)}$ | $0$ | $R_{12}^{(1)}$ | $1^*0$ | $R_{12}^{(2)}$ | $1^*0$ |
| $R_{13}^{(0)}$ | $\phi$ | $R_{13}^{(1)}$ | $\phi$ | $R_{13}^{(2)}$ | $1^*01$ |
| $R_{21}^{(0)}$ | $\phi$ | $R_{21}^{(1)}$ | $\phi$ | $R_{21}^{(2)}$ | $\phi$ |
| $R_{22}^{(0)}$ | $\epsilon$ | $R_{22}^{(1)}$ | $\epsilon$ | $R_{22}^{(2)}$ | $\epsilon$ |
| $R_{23}^{(0)}$ | $1$ | $R_{23}^{(1)}$ | $1$ | $R_{23}^{(2)}$ | $1$ |
| $R_{31}^{(0)}$ | $\phi$ | $R_{31}^{(1)}$ | $\phi$ | $R_{31}^{(2)}$ | $\phi$ |
| $R_{32}^{(0)}$ | $1$ | $R_{32}^{(1)}$ | $1$ | $R_{32}^{(2)}$ | $1$ |
| $R_{33}^{(0)}$ | $0+\epsilon$ | $R_{33}^{(1)}$ | $0+\epsilon$ | $R_{33}^{(2)}$ | $0+\epsilon+11$ |
| (a) | | (b) | | (c) | |

REIs for paths that can go through
(a) no state (b) state 1 only (c) states 1 & 2 only.

Indirective post
$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}\left(R_{11}^{(0)}\right)^* R_{1j}^{(0)}$$

$$\therefore R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}\cdot\left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$
$$= 0 + (1+\epsilon)(1+\epsilon)^* 0 = 1^*0 \qquad \because (1+\epsilon)^* = 1^*$$

Le start state is 1, final state is 3,
no of states = 3. ∴ The RE is
$$R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)}\cdot\left(R_{33}^{(2)}\right)^* R_{33}^{(2)}$$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}\left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$
$$= (1+\epsilon) + (1+\epsilon)(1+\epsilon)^*(1+\epsilon)$$
$$= 1^*$$

$$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)}\left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$
$$= \phi + (1+\epsilon)(1+\epsilon)^*\phi = \phi.$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}\left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$
$$= \phi + \phi(1+\epsilon)^*(1+\epsilon) = \phi$$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}\cdot\left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$
$$= \epsilon + \phi(1+\epsilon)^*(0) = \epsilon$$

$$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)}\left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$
$$= 1 + \phi(1+\epsilon)^*\phi = 1$$

$$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)}\left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$
$$= \phi + \phi(\ )^*(1+\epsilon) = \phi$$

$$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}\left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$
$$= 1 + \phi(1+\epsilon)^* 0 = 1$$

$$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)}\left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$
$$= (0+\epsilon) + \phi(0+\epsilon)^*\phi = 0+\epsilon$$ same for other REIs

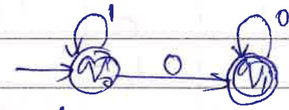
$$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)}\left(R_{22}^{(1)}\right)^* R_{23}^{(1)}$$
$$= \phi + 1^*0(\epsilon)^* 1 = 1^*01$$

$$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)}\left(R_{22}^{(1)}\right)^* R_{23}^{(1)}$$
$$= (0+\epsilon) + 1(\epsilon)^* 1$$
$$= 0+\epsilon+11$$

Now,
$$R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)}\cdot\left(R_{33}^{(2)}\right)^* R_{33}^{(2)}$$
$$= 1^*01 + 1^*01(0+\epsilon+11)^*(0+\epsilon+11)$$
$$= \underline{1^*01(0+11)^*}$$

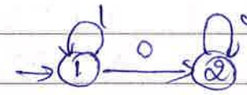
Platinum
Date: / /200

Pg 1) Obtain a RE for the FA shown below:




What is the lang. corresponding to the RE?

Soln Let $q_0 = 1$, $q_1 = 2$.




By Kleene's thm,

Basis: $k=0$

$R_{11}^{(0)} = \epsilon+1$ $\quad R_{12}^{(0)} = 0$ $\quad R_{21}^{(0)} = \phi$ $\quad R_{22}^{(0)} = \epsilon+0+1$

Induction:

The RE corresponding to the paths from state i to state j thro' a state which is not higher than k is given by:

$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$

When $k=1$,

$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$
$= (\epsilon+1) + (\epsilon+1)(\epsilon+1)^*(\epsilon+1)$
$= (\epsilon+1) + (\epsilon+1)1^*(\epsilon+1)$
$= (\epsilon+1) + 1^*$
$= 1^*$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)}$
$= 0 + (\epsilon+1)(\epsilon+1)^* 0$
$= 0 + 1^*0 = 1^*0$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$
$= \phi + \phi(\epsilon+1)^*(\epsilon+1) = \phi$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)}$
$= (\epsilon+0+1) + \phi(\epsilon+1)(0)$
$= \epsilon+0+1$

When $k=2$

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$
$= 1^* + 1^*0(\epsilon+0+1)^*\phi$
$= 1^*$

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$
$= 1^*0 + 1^*0(\epsilon+0+1)^*(\epsilon+0+1)$
$= 1^*0 + 1^*0(0+1)^*(\epsilon+0+1)$
$= 1^*0 + 1^*0(0+1)^*$
$= 1^*0(0+1)^*$

$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$
$= \phi + (\epsilon+0+1)(\epsilon+0+1)^*\phi$
$= \phi$

$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$
$= (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1)$
$= (\epsilon+0+1) + (0+1)^*$
$= (0+1)^*$

Since the total no of states is 2, the max val of $k=2$.

Since the start state is 1 & final state is 2, the RE is given by

$R_{12}^{(2)} = 1^*0(0+1)^*$.

i.e The RE for DFA is $1^*0(0+1)^*$ which is the lang. consisting of any no of 1's followed by a zero followed by strings of 0's & 1's.

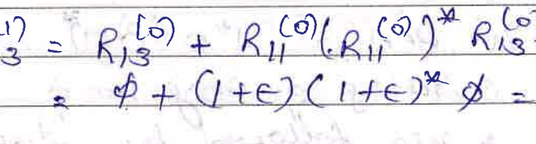
Platinum
Date: / /200

Consider the DFA shown below:

| | Σ | |
|---|---|---|
| states | 0 | 1 |
| →$q_1$ | $q_2$ | $q_1$ |
| $q_2$ | $q_3$ | $q_1$ |
| *$q_3$ | $q_3$ | $q_2$ |

Obtain the RE $R_{ij}^{(k)}$, $R_{ij}^{(n)}$ and simplify the RE as much as possible.

Soln By renaming the states & constructing FA,




By applying Kleene's thm:

Basis: When $k=0$,

$R_{11}^{(1)} = 1+\epsilon$ $\quad R_{21}^{(0)} = 1$ $\quad R_{31}^{(0)} = \phi$

$R_{12}^{(1)} = 0$ $\quad R_{22}^{(0)} = \epsilon$ $\quad R_{32}^{(0)} = 1$

$R_{13}^{(1)} = \phi$ $\quad R_{23}^{(0)} = 0$ $\quad R_{33}^{(0)} = 0+\epsilon$

Induction

When $k=1$,

$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)}$

$= (1+\epsilon)+(1+\epsilon)(1+\epsilon)^*(1+\epsilon)$

$= (1+\epsilon)+1^* = 1^*$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$

$= 0+(1+\epsilon)(1+\epsilon)^* 0$

$= 0+1^*0 = 1^*0$

$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}$

$= \phi + (1+\epsilon)(1+\epsilon)^* \phi = \phi$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)}$

$= 1+1(1+\epsilon)^*(1+\epsilon)$

$= 1+1.1^* = 11^*$

Platinum
Date: / /200

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$

$= \epsilon + 1(1+\epsilon)^* 0$

$= \epsilon + 11^*0$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}$

$= 0+1\cdot(1+\epsilon)^*\phi$

$= 0$

$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)}\cdot(R_{11}^{(0)})^* R_{11}^{(0)}$

$= \phi + \phi(\,)(\,) = \phi$

$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$

$= 1+\phi(\,)(\,) = 1$

$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)}\cdot(R_{11}^{(0)})^* R_{13}^{(0)}$

$= (0+\epsilon)+\phi(\,)(\,)$

$= (0+\epsilon)$

When $k=2$

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}\cdot(R_{22}^{(1)})^* R_{21}^{(1)}$

$= 1^* + 1^*0\cdot(\epsilon+11^*0)^* 11^*$

$= 1^* + 1^*0(11^*0)^* 11^*$

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$

$= 1^*0 + 1^*0(\epsilon+11^*0)^*(\epsilon+11^*0)$

$= 1^*0 + 1^*0(11^*0)^*$

$= 1^*0(11^*0)^*$

$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^*\cdot R_{23}^{(1)}$

$= \phi + 1^*0(\epsilon+11^*0)^*0$

$= 1^*0(11^*0)^*0$

Platinum Date: / /200

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)}$$
$$= 11^* + (\epsilon + 11^*0)(\epsilon + 11^*0)^* 11^*$$
$$= 11^* + (\epsilon + 11^*0)(11^*0)^* 11^*$$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$$
$$= (\epsilon + 11^*0) + (\epsilon + 11^*0)(\epsilon + 11^*0)^* (\epsilon + 11^*0)$$
$$= (\epsilon + 11^*0) + (\epsilon + 11^*0)(11^*0)^* (\epsilon + 11^*0)$$

$$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{23}^{(1)}$$
$$= 0 + (\epsilon + 11^*0)(\epsilon + 11^*0)^* 0$$
$$= 0 + (\epsilon + 11^*0)(11^*0)^* 0$$

$$R_{31}^{(2)} = R_{31}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)}$$
$$= \phi + 1 \cdot (\epsilon + 11^*0)^* 11^*$$
$$= 1(11^*0)^* 11^*$$

$$R_{32}^{(2)} = R_{32}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$$
$$= 1 + 1(\epsilon + 11^*0)^* (\epsilon + 11^*0)$$
$$= 1 + 1(11^*0)^* (\epsilon + 11^*0)$$

$$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^* R_{23}^{(1)}$$
$$= (0 + \epsilon) + 1(\epsilon + 11^*0)^* \cdot 0$$
$$= (0 + \epsilon) + 1(11^*0)^* 0$$

The RE is given by $R_{13}^{(3)}$ is calculated as

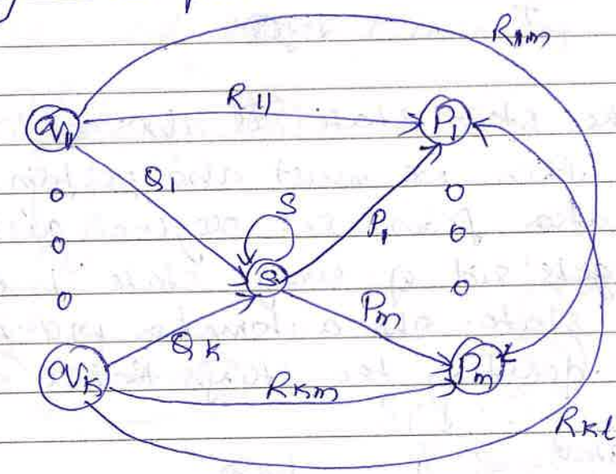
$$R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^* R_{33}^{(2)}$$
$$= 1^*0(11^*0)^*0 + 1^*0(11^*0)^*0[(0+\epsilon) + 1(11^*0)^*0]^*$$
$$[(0+\epsilon) + (1 \cdot (11^*0)^* 0)]$$

=

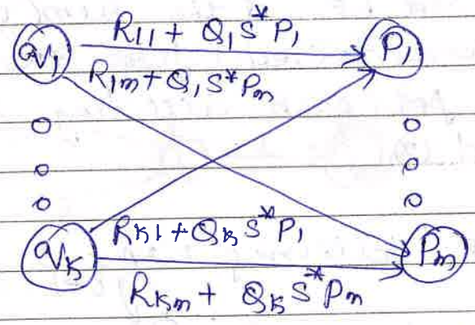
Platinum Date: / /200

## Converting DFA's to Reg. expns by Eliminating States :-

The lang. of the automaton is the union over all paths from the start state to an accepting state of the lang. formed by concatenating the languages of the RE's along that path.




fig① A state S about to be eliminated




fig② Result of eliminating state S from fig ①

The strategy for constructing a RE from FA is as follows :

1) For each accepting state $q_i$, apply the above reduction process to produce an equivalent automaton with RE labels on the arcs. Eliminate all states except $q_i$ and start state $q_0$.

2) If $q \neq q_0$, then we shall be left with a two-state automaton that looks like fig ③. The RE for the accepted strings can be described in various ways. One is $(R+SU^*T)^*SU^*$.

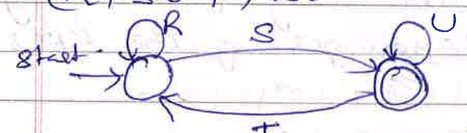


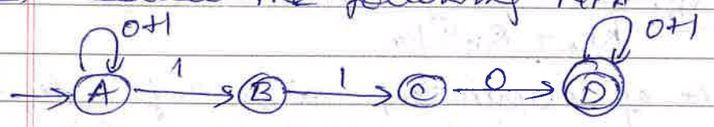

fig ③

3) If the start state is also an accepting state, then we must also perform a state-elimination from the original automaton that gets rid of every state but the start state. The automaton will be like fig ④. The RE denoting the strings that it accepts is $R^*$

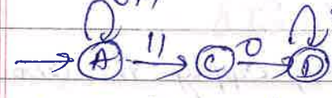

fig ④

4) The desired RE is the sum (union) of all the expressions derived from the reduced automata for each accepting state by rule ② and ③.

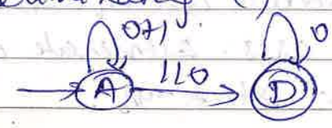
Prob Reduce the following NFA:



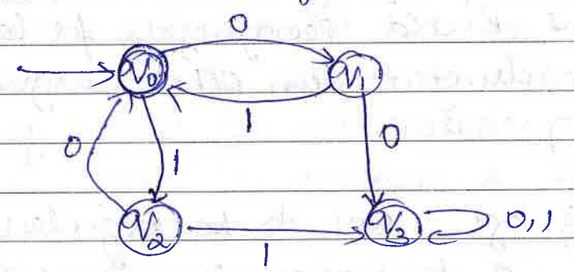

Soln By eliminating state B,



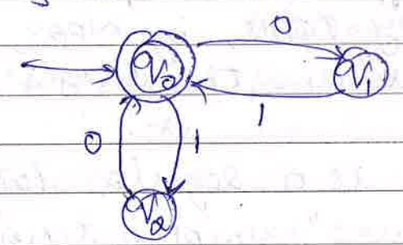

By eliminating C,

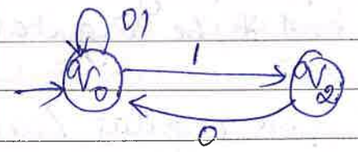


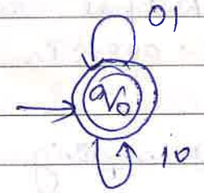

⇒ Obtain a RE for the FA shown below:




Soln $q_3$ is dead state. Eliminating $q_3$,

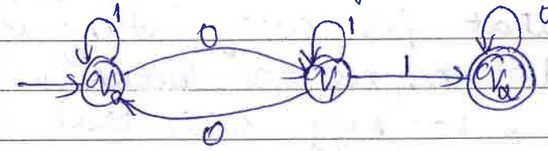


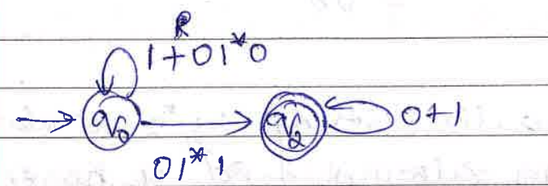




∴ The RE is $(01+10)^*$

6) Obtain a RE for the FA shown below.




Soln




By comparing it with fig ③

$R = 1+01^*0$ $\quad T = \phi$

$S = 01^*1$ $\quad U = 0+1$

$(R+\phi S U^* T)^* SU^*$

$(1+01^*0)+$

$(R+SU^*T)^*SU^*$

$(1+01^*0)+(01^*1(0+1)^*\phi)^*.01^*1(0+1)^*$

$=(1+01^*0)+\phi^*01^*1(0+1)^*$

$=(1+01^*0)+01^*1(0+1)^*$

Platinum Date: / /200

# PROPERTIES OF REGULAR LANGUAGES

Regular languages exhibit closure properties which let us build recognizers for languages that are constructed from other languages by certain operations.

## Proving languages not to be Regular

The class of languages known as Reg. langs has atleast four different descriptions. They are lang's accepted by DFA's, by NFA's, and by ε-NFA's. they are also the lang's defined by RE's.

Not every lang. is a regular language. A powerful technique "pumping lemma" shows certain lang's not to be regular.

## The pumping lemma for Regular Languages

**Theorem:**

The pumping lemma

Let L be a regular language. Then there exists a constant n (which depends on L) such that for every string w in L such that $|w| \geq n$, we can break w into three strings, $w = xyz$ such that

1) $y \neq \epsilon$
2) $|xy| \leq n$
3) For all $k \geq 0$, the string $xy^k z$ is also in L.

That is, we can always find a nonempty string y not too far from the beginning of w that can be "pumped"; ie repeating y any no. of times or deleting it (the case k=0), keeps the resulting string in the lang. L.

**Proof:** Suppose L is regular. Then $L = L(A)$ for some DFA A. Suppose A has n states. Now, consider any string w of length n or more, say $w = a_1 a_2 \cdots a_m$, where $m \geq n$ and each $a_i$ is an input symbol. For $i = 0, 1, \cdots n$ define state $p_i$ to be $\hat{\delta}(q_0, a_1 a_2 \cdots a_i)$ where $\delta$ is the tf of A, and $q_0$ is the start state of A. That is $p_i$ is the state A is in after reading the first i symbols of w. Note that $p_0 = q_0$.

By the pigeonhole principle, it is not possible for the n+1 different $p_i$'s for $i = 0, 1, \cdots n$ to be distinct, since there are only n different states. Thus, we can find two dif integers i and j, with $0 \leq i < j \leq n$, such that $p_i = p_j$. Now, we can break $w = xyz$ as follows:

1) $x = a_1 a_2 \cdots a_i$
2) $y = a_{i+1} a_{i+2} \cdots a_j$
3) $z = a_{j+1} a_{j+2} \cdots a_m$

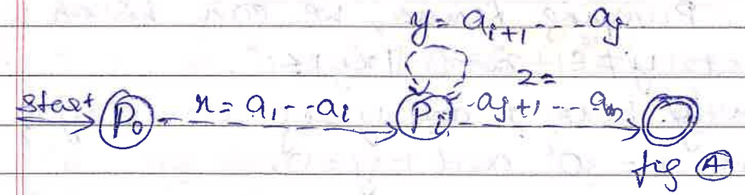



fig (A)

That is, x takes us to $p_i$ once; y takes us from $p_i$ back to $p_i$ (since $p_i$ is also $p_j$) and z is the balance of w. The relationships among the strings & states are suggested by fig (A)

Note that x may be empty, in case that i=0. Also, z may be empty if j=n=m. However, y cannot be empty, since i is strictly less than j.

Now, consider that automation A receives the input $xy^k z$ for any $k \geq 0$. If $k=0$ then the automation goes from the start state $q_0$ (which is also $p_0$) to $p_i$ on input x. Since $p_i$ is also $p_j$, it must be that A goes from $p_i$ to the accepting state shown in fig. on i/p z. Thus A accepts xz.

If $k>0$, then A goes from $q_0$ to $p_i$ on i/p x, circles from $p_i$ to $p_i$ k times on i/p $y^k$, and then goes to the accepting state on i/p z. Thus for any $k \geq 0$, $xy^k z$ is also accepted by A; ie $xy^k z$ is in L.

Applications of pumping lemma:

Ex1 S.T. the lang. L consisting of all palindromes over $(0+1)^*$ is not regular.

Soln Suppose L was regular.
Then there exists a constant n satisfying the conditions of pumping lemma.
Let $w = 0^n 1 0^n$
Using Pumping Lemma, we can break $w = xyz$ so that $y \neq \epsilon$ and $|xy| \leq n$.
Obviously both x and y consist only of 0's.
Suppose $x = 0^i$ and $y = 0^j$.
Then w.k.t by pumping lemma that $xy^0 z$ is in L.
Thus $0^{n-j} 1 0^n \in L$.
This is not possible.
Hence the lang. is not regular.

$n=8$
$i=3$
$j=4$
$n-j=4$

$\underbrace{000}_{x}\underbrace{0000}_{y}\underbrace{1}\underbrace{00000000}_{z}$

2) S.T. $L = \{a^n b^n \mid n \geq 0\}$ is not regular.
Let L is regular & n be the no of states in A. Consider the string $w = a^n b^n$
$|w| = 2n$ ie $|w| \geq n$
So, we can split w into xyz such that $|xy| \leq n$ & $|y| \geq 1$ as shown

$w = \underbrace{aaaaaa}_{n}\underbrace{bbbbbb}_{n}$
$\quad\quad x \quad y \quad z$

where
$|x| = n-1$ $\quad |xy| = |x| + |y|$ $\quad |z| = n$
$|y| = 1$ $\quad\quad = n-1+1 = n$
According to pumping lemma,
$xy^k z \in L$ for $k = 0, 1, 2, \ldots$
If $k=0$, y does not appear.
$\therefore$ No of a's < no of b's.
W.K.T if $k = 0, 3, \ldots$
no. of a's > no of b's
But acc to PL, equal no of a's should be followed by n no of b's which is a contradiction to the assumption that the string is regular.
$\therefore$ the lang $L = \{a^n b^n \mid n \geq 0\}$ is not regular

3) S.T. $L = \{ww^R \mid w \in (0+1)^*\}$ is not regular.
Let L is regular & n is no of states in FA.
Consider the string $w\,w^R$

$\underbrace{1\ldots1}_{n}\underbrace{0\ldots0}_{n}\underbrace{0\ldots0}_{n}\underbrace{1\ldots1}_{n}$

Since $|w| \geq n$, string w can be split into xyz such that $|xy| \leq n$ & $|y| \geq 1$ as shown

$\underbrace{1\ldots1}_{x}\underbrace{1}_{y}\underbrace{0\ldots00\ldots01\ldots1}_{z}$

where $|x| = n-1$ $\quad |uv| = n-1+1 = n$ is true.
$|y| = 1$

Platinum Date: / /200

Acc^g to P.L^m $xy^kz \in L$ for $k=0,1,2\ldots$
If $k=0$, y doesn't appear.
So, no of 0's in left of w will be less than no of 0's on right of w.
So the string is not of the form $ww^R$.
So $xy^kz \notin L$ when $k=0$.
This is a contradiction to the assumption that the lang is regular.
So the given lang. is not regular.

4) S.T $L=\{a^{n!} \mid n\geq 0\}$ is not regular.
Let L is regular & n be no of states in FA.
Let $w=a^{n!}$. It is clear that $|w|\geq n$
So, split w into xyz such that
$|xy|\leq n \quad |y|\geq 1$
$w=a^{n!} = \underbrace{a^j}_{u}\underbrace{a^k}_{v}\underbrace{a^{n!-j-k}}_{w}$ where $|u|=j$, $|v|=k$
$|uv|=j+k\leq n$
Acc^g to P.L^m $xy^kz \in L$ for $k=0,1,2\ldots$
i.e $a^j(a^k)^i a^{n!-j-k} \in L$
If we choose $i=0$, it means that
$a^j a^{n!-j-k} \in L$ i.e $xz \in L$
$\Rightarrow a^{n!-k} \in L$
$n! > n!-k$
When $k=1$, $n! > n!-1$
But acc^g to P.L^m $n!=n!-1$ which is false & is a contradiction. So, L cannot be regular.
So the lang $L=\{a^{n!} \mid n\geq 0\}$ is not regular.

# Equivalence & Minimization of Automata

## Testing Equivalence of states :-

**Basis :-**

If p is an accepting state & q is non accepting, then the pair {p, q} is distinguishable.

**Induction :-**

Let p & q be states such that for some input symbol a, r = δ(p, a) & s = δ(q, a) are a pair of states known to be distinguishable. Then {p, q} is a pair of distinguishable states. The reason this rule makes sense is that there must be some string w that distinguishes r from s; i.e. exactly one of δ(r, w) and δ(s, w) is accepting. Then string aw must distinguish p from q, since δ(p, aw) and δ(q, aw) is the same pair of states as δ(r, w) and δ(s, w).

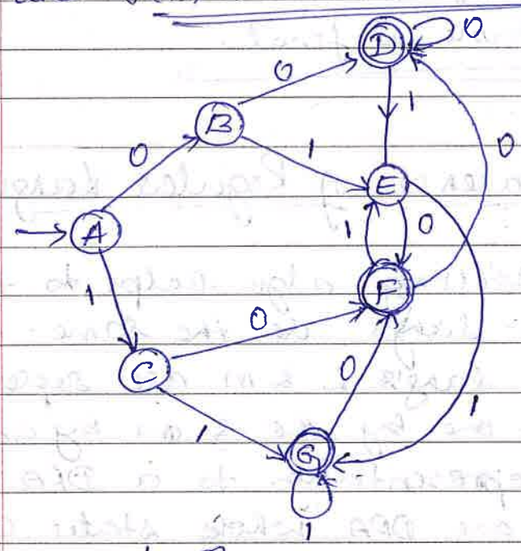



fig ①

An Automaton with Equivalent states

Non-Acc A B C E

Accepting D F G

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| B | X | | | | | |
| C | X | X | | | | |
| D | X | X | X | | | |
| E | X | X | · | X | | |
| F | X | X | X | · | X | |
| G | X | X | X | X | X | X |

Table of state equivalences

{B, E}

Refer pg 93 bot

Platinum Date: / /200

Initially the entire table is blank. Now, since states D, F and G are the only accepting states, they cannot be equivalent to A, B, C and E. Accordingly, we fill the corresponding squares with X. Now, we can use these distinguishable pairs to find others. For instance, since {E, G} is distinguishable & states B and E go to E and G on input 1, we find that {B, E} is also distinguishable. In a similar manner each of the blank squares is examined and distinguishable pairs identified whenever we find that the transition from a pair of states either on 0 or 1 goes to distinguishable states.

After one pass we find there are no more distinguishable pairs. The two remaining pairs {C, E} and {D, F} are equivalent since they have identified transitions on both 0 & 1. Thus the table is final.

## Testing Equivalence of Regular Languages

The table filling algo. helps to test if two regular langs are the same.

Suppose two langs L & M are represented in some way - one by R.E & one by an NFA. convert each representation to a DFA. Now, imagine one DFA whose states are the union of the states of the DFA's for L and M. Technically, this DFA has two start states, but it is irrevelant irrelevent as far as testing state equivalence is concerned, so make any state the lone start state.

Now, test if the start states of the two original DFA's are equivalent, using the table filling algo. If they are equivalent, then L = M and if not, then L ≠ M.

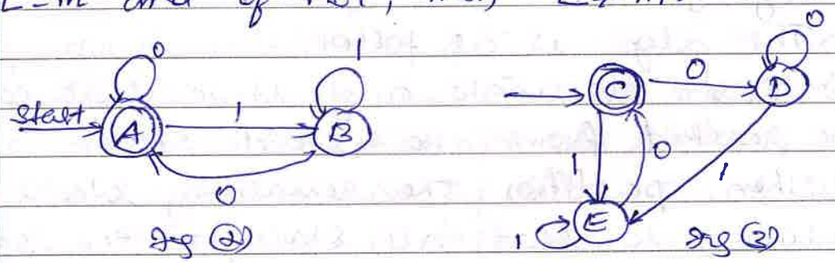



Fig (2) Fig (3)

Two equivalent DFA's

The lang of RE is $\epsilon + (0+1)^*0$

For fig (3) the partition of states into blocks is {{A}, {B}, {C, E}, {D, F}, {G}}

## Minimization of DFA's

It means for each DFA, we can find an equivalent DFA that has as few states as any DFA accepting the same language. This minimum-state DFA is unique for the language.

The algo is as follows:

1) First, eliminate any state that cannot be reached from the start state.
2) Then, partition the remaining states into blocks, so that all states in the same block are equivalent, and no pair of states from dif. blocks are equivalent.

**Thm**

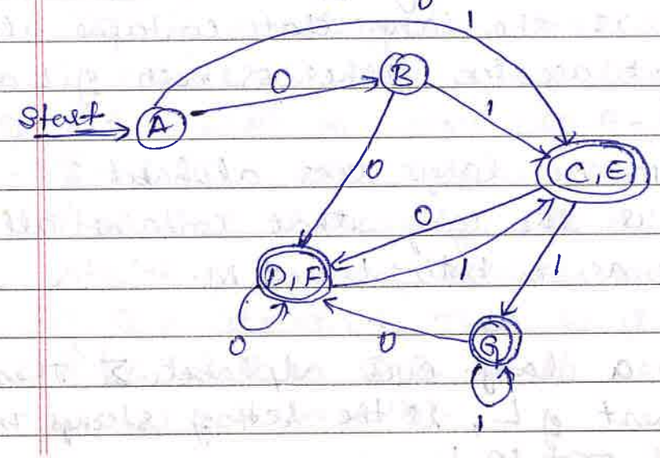
The equivalence of states is transitive. ie if in some DFA A, states p & q are equivalent & also q and r are eqv, then it must be that p and r are equivalent.

**Thm 4.24**

If we create for each state q of a DFA a block consisting of q and all the states equivalent to q, then the different blocks of states form a partition of the set of state. Ie each state is in exactly one block. All members of a block are equivalent. and no pair of states chosen from dif blocks are equivalent.

We state the algo. for minimizing a DFA $A = (Q, \Sigma, \delta, q_0, F)$

1) Use the table-filling algo. to find all the pairs of equivalent states.
2) Partition the set of states Q into blocks of mutually equivalent states, by the method described above.
3) Construct the minimum-state equivalent DFA B by using the blocks as its states. Let $\delta$ be the tr. fn of B. Suppose S is a set of equivalent states of A, and a is an i/p symbol. Then there must exist one block T of states such that for all states q in S, $\delta(q,a)$ is a member of block T. For if not, then i/p symbol a takes two states p and q of S to states in dif blocks, and those states are distinguishable by Thm 4.24. That fact lets us conclude that p & q are not equivalent and they did not both belong in S. As a consequence, we can set $\delta(S,a) = T$. In addition,
   (a) The start state of B is the block containing the start state of A.
   (b) The set of accepting states of B is the set of blocks containing accepting states of A. Note that if one state of a block is accepting, then all the states of that block must be accepting.



Platinum
Date: / /200

## Closure Properties of Regular Languages

Closure properties express the idea that when one (or several) lang's are regular, then certain related lang's are also regular.

Summary of the principal closure properties for regular languages:

1) The union of two regular languages is regular
2) The intersection —"— —"—
3) The complement of a —"— —"—
4) The difference of two —"— —"—
5) The reversal of a —"— —"—
6) The closure (star) of a —"— —"—
7) The concatenation of —"— —"—
8) A homomorphism (substitution of strings for symbols) of a reg. lang. is regular.
9) The inverse homomorphism of a reg. lang. is regular.

## Closure of Regular Lang's under Boolean Operation:-

The 3 boolean op are union, intersection & complementation.

1) Let L and M be languages over alphabet $\Sigma$. Then $L \cup M$ is the lang. that contains all strings that are in either or both of L and M.

2) Let L and M be langs over alphabet $\Sigma$. Then $L \cap M$ is the lang. that contains all strings that are in both L and M.

3) Let L be a lang. over alphabet $\Sigma$. Then $\bar{L}$, the complement of L, is the set of strings in $\Sigma^*$ that are not in L.

Platinum
Date: / /200

The reg. langs are closed under all three of the boolean operations.

## Closure under Union:-

Thm:

If L and M are reg. langs then so is $L \cup M$.

Proof:

Since L and M are regular, they have reg. exprns, say $L = L(R)$, and $M = L(S)$. Then $L \cup M = L(R+S)$ by the defn of the + operator for reg. exprns

## Closure under Complementation

Starting with a RE, we could find a RE for its complement as follows:

1) Convert the RE to an $\epsilon$-NFA.
2) Convert that $\epsilon$-NFA to a DFA by the subset construction.
3) Complement the accepting states of that DFA.
4) Turn the complement DFA back into a RE using the construction technique.

Thm

If L is a reg. lang over alphabet $\Sigma$, then $\bar{L} = \Sigma^* - L$ is also a reg. lang.

Proof: Let $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$. Then $\bar{L} = L(B)$, where B is the DFA $(Q, \Sigma, \delta, q_0, Q-F)$. That is, B is exactly like A, but the accepting states of A have become nonaccepting states of B & vice versa. Then w is in $L(B)$ if $\hat{\delta}(q_0, w)$ is in Q-F, which occurs if & only if w is not in $L(A)$.

Platinum
Date: / /200

## Closure under Intersection :-

The intersection of langs L & M is given by,

$L \cap M = \overline{\overline{L} \cup \overline{M}}$ - (De-morgan's)

In general, the intersection of two sets is the set of elements that are not in the complement of either set.

The other De-Morgan's Law,

$L \cup M = \overline{\overline{L} \cap \overline{M}}$

Refer Thm:

## Closure under Difference :-

In terms of languages, L-M, the difference of L and M is the set of strings that are in lang L but not in M.

Thm: If L & M are reg, langs then so is L-M.

Proof: Observe that $L - M = L \cap \overline{M}$. By thm 4.5, $\overline{M}$ is regular & by thm 4.8 $L \cap \overline{M}$ is reg. Therefore L-M is regular.

## Reversal

The reversal of a string $a_1 a_2 \cdots a_n$ is the string written backwards ie $a_n a_{n-1} \cdots a_1$. We use $w^R$ for the reversal of string w.

$\epsilon^R = \epsilon$.

The reversal of a lang L, written $L^R$ is the lang. consisting of the reversals of all its strings.

Ex If $L = \{001, 10, 111\}$ then

$L^R = \{100, 01, 111\}$

Platinum
Date: / /200

# CONTEXT-FREE GRAMMARS

Components of Grammar
- Variables
- Terminals
- Productions
- Start symbol

## Chomsky Hierarchy:-

Types of Grammar

| Type 0 (Phase structured) Gr. | Type 1 (context sensitive) gr | Type 2 (context free) gr | Type 3 Regular gr. |
|---|---|---|---|
| $\alpha \to \beta$ | $\lvert\beta\rvert \geq \lvert\alpha\rvert$ | $A \to \alpha$ | Rt linear or left linear |
| $\alpha \in (V \cup T)^+$ | $\alpha, \beta \in (V \cup T)^+$ | $\alpha \in (V \cup T)^*$ | Rt - $A \to wB$ <br> Rt - $A \to Bw$ |
| $\beta \in (V \cup T)^*$ | $\epsilon$-free gr. | | |
| $S \to aAb \mid \epsilon$ <br> $aA \to bAA$ <br> $bA \to a$ | $S \to aAb$ <br> $aA \to bAA$ <br> $bA \to aa$ | $S \to aB \mid bA \mid \epsilon$ <br> $A \to aA \mid b$ <br> $B \to bB \mid a \mid \epsilon$ | $S \to aB \mid bA \mid \epsilon$ <br> $A \to aA \mid b$ <br> $B \to bB \mid a \mid \epsilon$ |

$G = (V, T, P, S)$

The 4 imp components of grammar are:

1. Non-terminals or syntactic categories:
   Finite set of variables. Each var represents a language.
2. Terminals or Terminal Symbols: A finite set of symbols that form the strings of the language being defined.
3. Start Symbol: Represents the language being defined
4. Productions: A finite set of productions or rules represent the recursive definition of a language.
   Each production consists of:

Platinum
Date: / /200

# PUSH DOWN AUTOMATION

The formal defn. of PDA involves seven components.

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

The components have the following meanings:

Q: A finite set of states.

$\Sigma$: A finite set of input symbols

$\Gamma$: A finite stack alphabet. It is the set of symbols allowed to push on to the stack.

$\delta$: The transition function. $\delta$ takes as argument a triple $\delta(q, a, X)$ where:

1) q is a state in Q
2) a is either an input symbol in $\Sigma$ or $a = \epsilon$, the empty string which is assumed not to be an input symbol.
3) X is a stack symbol, i.e. a member of $\Gamma$.

The output of $\delta$ is a finite set of pairs $(p, \gamma)$ where p is the new state and $\gamma$ is the string of stack symbols that replaces X at the top of the stack.

For instance, if $\gamma = \epsilon$ then the stack is popped.

if $\gamma = X$ then stack is ~~intersecka~~ unchanged

3 if $\gamma = YZ$, then X is replaced by Z and Y is pushed on to the stack.

$q_0$: The start state. The PDA is in this state before making any transitions

$Z_0$: The stack start symbol. Initially, the PDA's stack consists of one instance of this symbol and nothing else.

F: The set of accepting states or final states.

Platinum
Date: / /200

Design a PDA to accept the language
$L = \{a^i b^j c^k \mid i+j=k\}$

Soln
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, b) = (q_1, bb)$
$\delta(q_1, b, Z_0) = (q_1, bZ_0)$
$\delta(q_1, c, b) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$.

Platinum
Date: / /200

# TURING MACHINES

Design a TM to accept the language consisting of all palindromes of 0's and 1's.

BabaB
BXbaB
:

| | Symbol | | | | |
|---|---|---|---|---|---|
| State | a | b | X | Y | B |
| $q_0$ | $(q_1, X, R)$ | $(q_2, Y, R)$ | $(q_6, X, R)$ | $(q_6, Y, R)$ | $(q_6, B, R)$ |
| $q_1$ | $(q_1, a, R)$ | $(q_1, b, R)$ | $(q_3, X, L)$ | $(q_3, Y, L)$ | $(q_3, B, L)$ |
| $q_2$ | $(q_2, a, R)$ | $(q_2, b, R)$ | $(q_4, X, L)$ | $(q_4, Y, L)$ | $(q_4, B, L)$ |
| $q_3$ | $(q_5, X, L)$ | – | $(q_6, X, R)$ | $(q_6, Y, R)$ | – |
| $q_4$ | – | $(q_5, Y, L)$ | $(q_6, X, R)$ | $(q_6, Y, R)$ | – |
| $q_5$ | $(q_5, a, L)$ | $(q_5, b, L)$ | $(q_0, X, R)$ | $(q_0, Y, R)$ | – |

# PDA

* Define PDA and construct a PDA that accepts the following languages. $L = \{w : w \in (a+b)^* \text{ and } n_a(w) = n_b(w)\}$. Write the instantaneous description for the string aababb. (Jan-09) (12 marks)

The PDA is

aababb

$Q = \{q_0, q_1\}$

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

$\Gamma = \{a, b, Z_0\}$


$\delta$:

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \epsilon)$

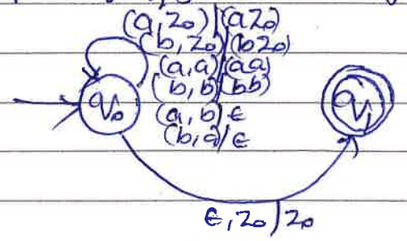
$\delta(q_0, b, a) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$

$q_0 \in Q$ is the start state

$Z_0 \in \Gamma$ - Initial symbol on stack

$F = \{q_1\}$ is the final state.




I.D

$(q_0, aababb, Z_0) \vdash$

$\vdash (q_0, ababb, aZ_0)$

$\vdash (q_0, babb, aaZ_0)$

$\vdash (q_0, abb, aZ_0)$

$\vdash (q_0, bb, aaZ_0)$

$\vdash (q_0, b, aZ_0)$

$\vdash (q_0, \epsilon, Z_0)$

$\vdash (q_1, \epsilon, Z_0)$

* Discuss the languages accepted by a PDA. Design a PDA for the language that accepts the strings with $n_a(w) < n_b(w)$ where $w \in (a+b)^*$ & show the IDs of the PDA on input abbab. (14) marks: Jun-July-09

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_0, \epsilon)$

$\delta(q_0, a, b) = (q_0, \epsilon)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, \epsilon, b) = (q_1, b)$ - Final state

Platinum Date: / /200

④ Obtain a PDA to accept the language $L = \{a^n b^n \mid n \geq 0\}$ by a final state. Give the graphical representation for PDA obtained. Show the moves made by the PDA for the string aaabbb. (10) M J/Feb 06

Soln The PDA is

$Q = \{\}$

$\Sigma =$

$\Gamma =$ a a a b b b

$\delta$: $\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$ - Accepted by final state

$\delta(q_1, \epsilon, \epsilon) = (q_2, \epsilon)$ - Accepted by empty stack.

## Equivalence of PDA's and CFG's

### From Grammars to PDA

Let $G = (V, T, Q, S)$ be a CFG. Construct the PDA P that accepts $L(G)$ by empty stack as follows:

$P = (\{q\}, T, V \cup T, \delta, q, S)$

where transition function $\delta$ is defined by:

1) For each variable A,

$\delta(q, \epsilon, A) = \{(q, \beta) \mid A \to \beta$ is a prodn of $P\}$

2) For each terminal a,

$\delta(q, a, a) = \{(q, \epsilon)\}$

Ex Convert the given grammar to PDA

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$E \to I \mid E*E \mid E+E \mid (E)$

The set of terminals for the PDA are

$\{a, b, 0, 1, (, ), +, *\}$

The transition function for the PDA is

a) $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$

b) $\delta(q, \epsilon, E) = \{(q, I), (q, E*E), (q, E+E), (q, (E))\}$

c) $\delta(q, a, a) = \{(q, \epsilon)\}$ $\quad \delta(q, (, () = \{(q, \epsilon)\}$

$\delta(q, b, b) = \{(q, \epsilon)\}$ $\quad \delta(q, ), )) = \{(q, \epsilon)\}$

$\delta(q, 0, 0) = \{(q, \epsilon)\}$ $\quad \delta(q, +, +) = \{(q, \epsilon)\}$

$\delta(q, 1, 1) = \{(q, \epsilon)\}$ $\quad \delta(q, *, *) = \{(q, \epsilon)\}$

Note (a) & (b) come from rule (1) & c transitions come from rule (2). Also, $\delta$ is empty except as defined by (a) through (c).

Platinum
Date: / /200

(*) Design a TM to accept all set of palindromes over $\{0,1\}^*$. Also write its transition diagram and ID on the string 10101. (14 Marks)

| | 0 | 1 | X | Y | B |
|---|---|---|---|---|---|
| $q_0$ | $(q_1, X, R)$ | $(q_2, Y, R)$ | $(q_7, X, R)$ | $(q_7, Y, R)$ | |
| $q_1$ | $(q_1, 0, R)$ | $(q_1, 1, R)$ | $(q_3, X, L)$ | $(q_3, Y, L)$ | $(q_3, B, L)$ |
| $q_2$ | $(q_2, 0, R)$ | $(q_2, 1, R)$ | $(q_4, X, L)$ | $(q_4, Y, L)$ | $(q_4, B, L)$ |
| $q_3$ | $(q_5, X, L)$ | - | $(q_7, X, R)$ | - | |
| $q_4$ | - | $(q_6, Y, L)$ | - | $(q_7, X, R)$ | |
| $q_5$ | $(q_5, 0, L)$ | $(q_5, 1, L)$ | $(q_0, X, L)$ | $(q_0, Y, L)$ | |
| $q_6$ | $(q_6, 0, L)$ | $(q_6, 1, L)$ | $(q_0, X, R)$ | $(q_0, Y, R)$ | |
| $q_7$ | | | $(q_7, X, R)$ | $(q_7, Y, R)$ | $(q_8, B, R)$ |
| $q_8$ | | | | | |

(*) Obtain a TM to accept the language $L = \{ww^R / w \in (a+b)^*\}$

⇒ It is a palindrome of even length.

| $\delta$ | a | b | B |
|---|---|---|---|
| $q_0$ | - | - | $q_1, B, R$ |
| $q_1$ | $q_2, B, R$ | $q_5, B, R$ | $q_7, B, R$ |
| $q_2$ | $q_2, a, R$ | $q_2, b, R$ | $q_3, B, L$ |
| $q_3$ | $q_4, B, L$ | - | $q_7, B, R$ |
| $q_4$ | $q_4, a, L$ | $q_4, b, L$ | $q_1, B, R$ |
| $q_5$ | $q_5, a, R$ | $q_5, b, R$ | $q_6, B, L$ |
| $q_6$ | - | $q_4, B, L$ | $q_7, B, R$ |
| $q_7$ | - | - | - |

palindrome even length entries

Platinum
Date: / /200

(*) Design a TM to accept the language $L = \{a^n b^n c^n / n \geq 1\}$. Give the graphical representation for the TM obtained.

| | a | b | c | X | Y | Z | B |
|---|---|---|---|---|---|---|---|
| $q_0$ | $q_1, X, R$ | | | | | | |
| $q_1$ | | | | | | | |
| $q_2$ | | | | | | | |
| $q_3$ | | | | | | | |
| $q_4$ | | | | | | | |
| $q_5$ | | | | | | | |
| $q_6$ | | | | | | | |

Platinum
Date: / /200

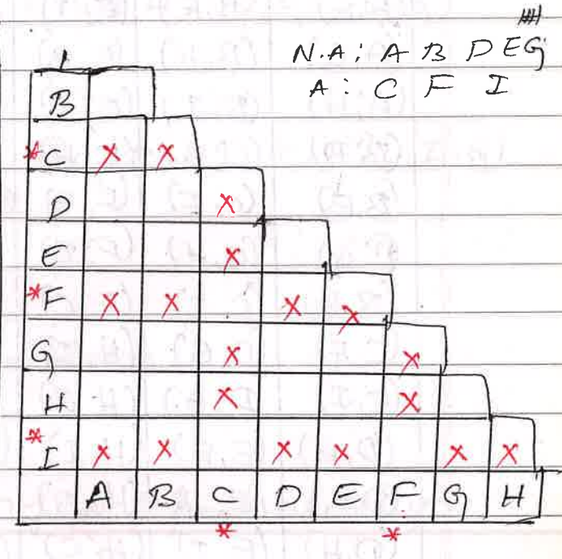
Design a TM for the language to accept the set of strings with equal no of 0's and 1's and also give the ID for the i/p 110100.

Platinum
Date: / /200

Minimize the following DFA using Table-filling algorithm where A is the start state. The states C, F and I are final states.

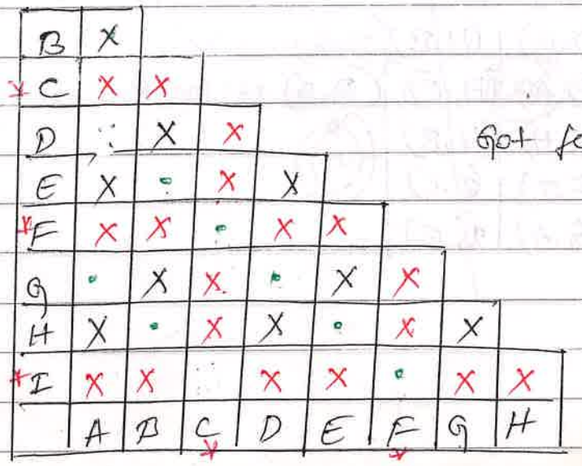
N.A: A B D E G
A: C F I

| | 0 | 1 |
|---|---|---|
| → A | B | E |
| B | C | F |
| C* | D | H |
| D | E | H |
| E | F | I |
| F* | G | B |
| G | H | B |
| H | I | C |
| I* | A | E |




Initial Horizontal marking:-
C, F, I are final states, The pairs (C, A), (F, A), (F, B), (F, D), etc have one final state & one non-final state. So mark them with X

Initial vertical marking:
The pairs (C, E), (C, G)(C H) etc are distinguishable So mark them with X in the table.




Got fragmented in next page.

Platinum
Date: / /200

| δ | 0 | 1 | |
|---|---|---|---|
| (A,B) | (B,C) | (E,F) | (B,C) is marked · so mark (A,B) |
| (A,D) | (B,E) | (E,H) | |
| (A,E) | (B,F) | (E,I) | (B,F) " " (A,E) |
| (A,G) | (B,H) | (E,B) | |
| (A,H) | (B,I) | (E,C) | (B,I) - - (A,H) |
| (B,D) | (C,E) | (F,H) | (C,E) " (B,D) |
| (B,E) | (C,F) | (F,I) | |
| (B,G) | (C,H) | (F,B) | (B,F) - - (B,G) |
| (B,H) | (C,I) | (F,C) | |
| (C,F) | (D,G) | (H,B) | |
| (C,I) | (D,A) | (H,E) | |
| (D,E) | (E,F) | (H,I) | (E,F) (D,E) |
| (D,G) | (E,H) | (H,B) | |
| (D,H) | (E,I) | (H,C) | (D,H) (C,H) |
| (E,G) | (F,H) | (I,B) | (B,I) (E,G) |
| (E,H) | (E,I) | (I,C) | |
| (F,I) | (G,A) | (B,E) | |
| (G,H) | (H,I) | (B,C) | (G,H) - (B,C) |

Table ←

| δ | 0 | 1 | |
|---|---|---|---|
| (A,D) | (B,E) | (E,H) | |
| (A,G) | (B,H) | (E,B) | |
| (B,E) | (C,F) | (F,I) | |
| (B,H) | (C,I) | (F,C) | |
| (C,F) | (D,G) | (H,B) | No more merges |
| (C,I) | (D,A) | (H,E) | (A,D) is marked · so mark (C,I) |
| (D,G) | (E,H) | (H,B) | |
| (E,H) | (F,I) | (I,C) | |
| (F,I) | (G,A) | (B,E) | |

Platinum
Date: / /200

Minimizing DFA:

1) Find the distinguishable & indistinguishable pairs.

(A,B), (A,G), (B,E), (B,H), (C,F), (D,G), (E,H), (F,I)

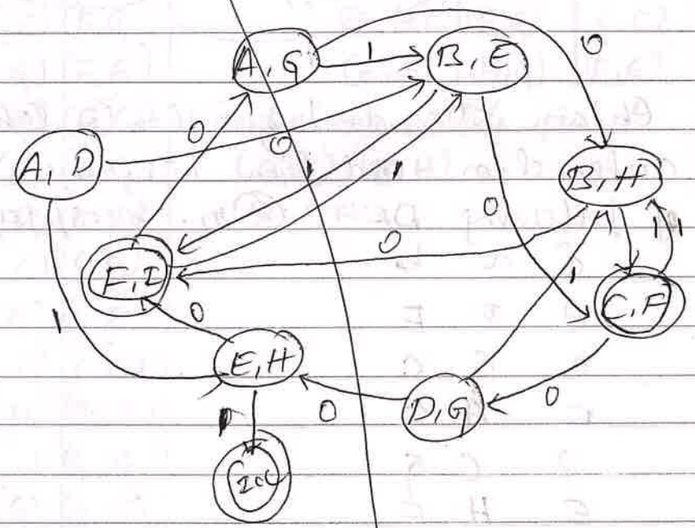
indistinguishable:

(A,D), (A,G), (B,E), (B,H), (C,F), (D,G), (E,H), (F,I), (C,I) :

distinguishable:

Compute the transition table:

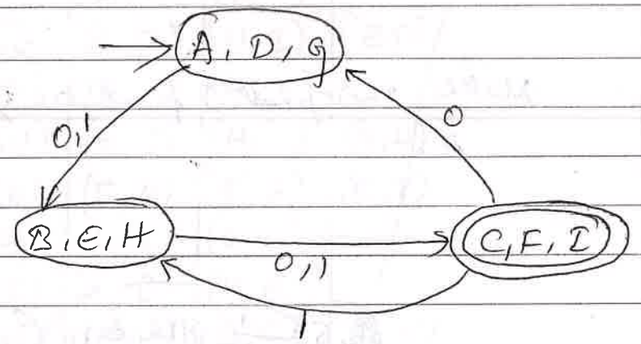
| δ | | |
|---|---|---|
| (A,D) | Make | non-final & final as groups |
| (A,G) | | A, |
| (B,E) | | |
| (B,H) | | |
| (C,F) | | |



Indistinguishable pairs:
(A,D), (A,G), (D,G) = (A,D,G)
(B,E), (B,H), (E,H) = (B,E,H)
(C,F), (C,I), (F,I) = (C,F,I)

Distinguishable state: does not exist.

| | 0 | 1 |
|---|---|---|
| → (A,D,G) | (B,E,H) | (B,E,H) |
| (B,E,H) | (C,F,I) | (C,F,I) |
| * (C,F,I) | (A,D,H) | (B,E,H) |

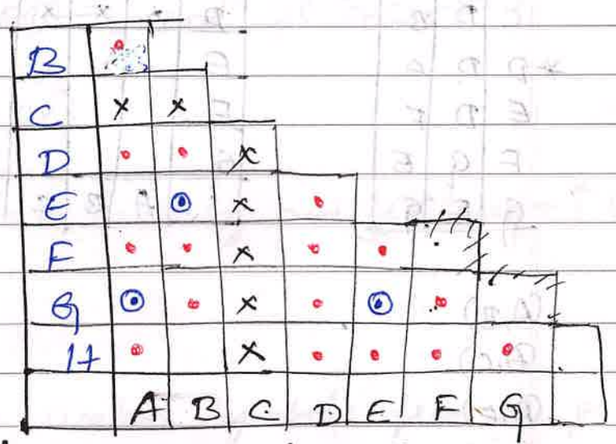



Qn 2 Obtain the distinguishable table for the automation and then minimize the states of following DFA. (Sm. Dec 06/Jan 07/July 07)

| δ | a | b |
|---|---|---|
| A | B | F |
| B | G | C |
| C | A | C |
| D | C | G |
| E | H | F |
| F | C | G |
| G | G | E |
| H | G | C |

Obtain the distinguishable table for the automation and then minimize the states of following DFA.

| δ | a | b |
|---|---|---|
| → A | B | F |
| B | G | C |
| * C | A | C |
| D | C | G |
| E | H | F |
| F | C | G |
| G | G | E |
| H | G | C |




| | a | b |
|---|---|---|
| (A,B) | (B,G) | (F,C) |
| (A,D) | (B,C) | (F,G) |
| (A,E) | (B,H) | (F,F) |
| (A,F) | (B,C) | (F,G) |
| (A,G) | (B,G) | (F,E) |
| (A,H) | (B,G) | (F,C) |
| (B,D) | (G,C) | (C,G) |
| (B,E) | (G,H) | (C,F) |
| (B,F) | (G,C) | (C,G) |
| (B,G) | (G,G) | (C,E) |
| (B,H) | (G,G) | (C,C) |
| (D,E) | (C,H) | (G,F) |
| (D,F) | (C,C) | (G,G) |
| (D,G) | (C,G) | (G,E) |
| (D,H) | (C,G) | (G,C) |
| (E,F) | (H,C) | (F,G) |
| (E,G) | (H,G) | (F,E) |
| (E,H) | (H,G) | (F,C) |
| (F,G) | (C,G) | (G,E) |
| (F,H) | (C,G) | (G,C) |

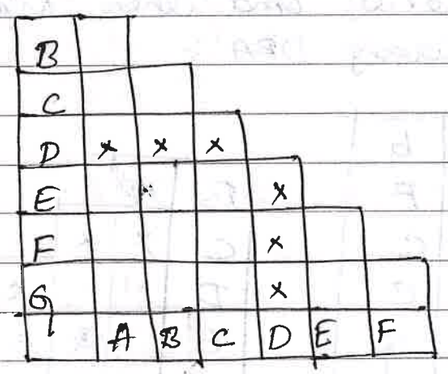
| | a | b |
|---|---|---|
| (A,E) | (B,H) | (F,F) |
| (A,G) | (B,G) | (F,E) |
| (B,E) | (G,H) | (C,F) |
| (B,H) | (G,G) | (C,C) |
| (E,G) | (H,G) | (F,E) |

Indistinguishable pairs
(A,E), (B,H)

Platinum
Date: / /200

Minimize the following DFA using Table filling algo.

| | 0 | 1 |
|---|---|---|
| → A | B | A |
| B | A | C |
| C | D | B |
| * D | D | A |
| E | D | F |
| F | G | E |
| G | F | G |




(A,B)
(A,C)
(A,E)
(A,F)
(E,G)

Platinum
Date: / /200

# CFG's

## Simplification of CFG's

### Elimination of useless symbols :-

Algorithm

$OV = \phi$

$NV = OV \cup \{A / A \to y \text{ and } y \in (OV \cup T)^*\}$

While $(OV_1 = NV)$
{ $OV = NV$;
$NV = OV \cup \{A / A \to y \text{ and } y \in (OV \cup T)^*\}$
}

$V_1 = OV$.

Problem

1) Eliminate the useless symbols from the gr.

$S \to aA | bB$
$A \to aA | a$
$B \to bB$
$D \to ab | Ea$
$E \to ac | d$.

| OV | NV | P. |
|---|---|---|
| $\phi$ | A, D, E | $A \to a$<br>$D \to ab$<br>$E \to d$ |
| A, D, E | A, D, E, S | $S \to aA$<br>$A \to aA$<br>$D \to Ea$ |
| A, D, E, S | A, D, E, S | – |

The gr is
$S \to aA$
$A \to a | aA$
$D \to Ea | ab$
$E \to d$.

The symbols D S E are not reachable from start state. ∴ The resulting gr is

$S \to aA$
$A \to aA | a$

Platinum
Date: / /200

2) Eliminate useless prodns from the following grammar

$A \to bA/Bba/aa$
$B \to aBa/b/D$
$C \to CA/AC/B$
$D \to a/\epsilon$

Soln

| OV | nV | P |
|---|---|---|
| $\phi$ | A,B,D | $A \to aa$<br>$B \to b$<br>$D \to a$ |
| A,B,D | A,B,D,C | $A \to bA$<br>$A \to Bba$<br>$B \to aBa$<br>$D \to D$<br>$C \to B$ |
| A,B,D,C | A,B,D,C | $C \to CA$<br>$C \to AC$ |

The gr obtained is

$A \to aa/bA/Bba$
$B \to b/aBa/D$
$C \to CA/AC/B$
$D \to a$

Step 2: The symbols c ~~are~~ not reachable from the start state.

∴ The resultant grammar is

$A \to Bba/bA/aa$
$B \to b/aBa/D$
$D \to a$

Platinum
Date: / /200

Elimination of Unit productions

Any prodn in G of the form $A \to B$ where $A, B \in V$ is a unit prodn.

It is undesirable ∵ one var simply replaces other var.

A unit prodn can be ~~replaced~~ eliminated by

1) Removing all prodns of the form $A \to A$
2) Add all non-unit prodns to $P_1$
3) For each var A find all var B such that $A \overset{*}{\Rightarrow} B$
4) Obtain the dependency graph.

for ex: $A \to B$
$B \to C$
$C \to B$

the graph will be

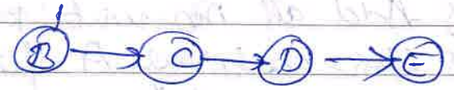

5) Note from graph
   a. $A \overset{*}{\Rightarrow} B$. B can be obtained from A. So all non-unit prodns generated from B can also be generated from A.
   x/y for other variables.
6) Finally, the unit prodns can be deleted from the grammar G.
7) The resulting gr generates the same language as accepted by G.

Prob Eliminate Unit prodns from the gr. ⇒

1) $S \to AB$
$A \to a$
$B \to C/b$
$C \to D$
$D \to \epsilon / bC$
$E \to d/Ab$

The unit prodns are
$B \to C$
$C \to D$
$D \to E$

The non unit prodns
$S \to AB$
$A \to a$
$B \to b$
$D \to bC$
$E \to d/Ab$.




From dependency graph
$D \overset{*}{\Rightarrow} E$
∴ The non unit prodns generated from E can also be generated from D.
$E \to d/Ab$ can also be obtained from D
∴ $D \to d/Ab$

The resulting D prodns are
$D \to bC$
$D \to d/Ab$

From graph $C \overset{*}{\Rightarrow} D$
∴ $C \to d/Ab$ · $C \to bC$
$C \to d/Ab$

From graph $B \overset{*}{\Rightarrow} C$
$B \to b$
$B \to d/Ab$
$B \to bC$

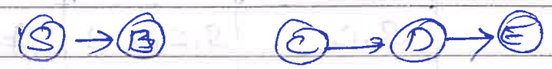
The final grammar is

$V' = \{B, A, B, C, D, E, S\}$
$T' = \{a, b, d\}$
$P' = \{ S \to AB$
$A \to a$
$B \to b/d/Ab/bC$
$C \to bC/d/Ab$
$D \to bC/d/Ab$
$E \to d/Ab$

2) 

| | Non unit prodns |
|---|---|
| $S \to Aa/B/Ca$ | $S \to Aa/Ca$ |
| $B \to aB/b$ | $B \to aB/b$ |
| $C \to Db/D$ | $C \to Db$ |
| $D \to E/d$ | $D \to d$, $E \to ab$ |
| $E \to ab$. | |

The unit prodns are
$S \to B$
$C \to D$
$D \to E$.




$S \overset{*}{\Rightarrow} B$ $\quad B \to aB/b$
∴ $S \to aB/b$

$D \overset{*}{\Rightarrow} E$. $\quad E \to ab$
∴ $D \to ab$.

$C \overset{*}{\Rightarrow} D$.
$C \to ab/d$.

The final gr is
$S \to aA/Ca/aB/b$
$B \to aB/b$
$C \to Db/ab/d$
$D \to d/ab$
$E \to ab$.

3) $S \to aAa/bBb/\epsilon$
$A \to C/a$
$B \to C/b$
$C \to CDE/\epsilon$
$D \to A/B/ab$

Jan 15
(10)

i) Eliminate $\epsilon$-prodns
(10) -"- any unit prodns in resulting gr
(10) (iii) -"- any useless symbols in -"-

(10)

PTO.

To eliminate $\epsilon$-prodns

$S \to \epsilon$ } C & S are nullable variables.
$C \to \epsilon$ }

Step1

| OV | NV | P |
|---|---|---|
| $\phi$ | S, C | $S \to \epsilon$<br>$C \to \epsilon$ |
| S, C | S, C, A, B | $A \to C$<br>$B \to C$ |
| S, C, A, B | S, C, A, B, D | $S \to aAa$<br>$S \to bBb$<br>$D \to A$<br>$D \to B$ |
| S, C, A, B, D | S, C, A, B, D | - |

The nullable vars are S, C, A, B, D.

Step2: Make each nullable var as null & find the grammar.

$S \to aAa \mid aa \mid bBb \mid bb$
$A \to C \mid a$
$B \to C \mid b$
$C \to CDE \mid DE \mid CE \mid E$
$D \to A \mid B \mid ab$

In this gs, the unit prodns are

$A \to C$ (A) → (C) → (E)
$B \to C$ (B) → (C)
$C \to E$
$D \to A$ (D) → (A)
$D \to B$ (D) → (B)

The non-unit prodns are

$S \to aAa \mid aa \mid bBb \mid bb$
$A \to a$, $B \to b$
$C \to CDE \mid DE \mid CE$

$A \overset{*}{\Rightarrow} C$
$\therefore A \to CDE \mid DE \mid CE$

$B \overset{*}{\Rightarrow} C$
$\therefore B \to CDE \mid DE \mid CE$

$D \overset{*}{\Rightarrow} A$
$\therefore D \to CDE \mid DE \mid CE$

$D \overset{*}{\Rightarrow} B$
$D \to CDE \mid DE \mid CE$

The gs is

$S \to aAa \mid bBb \mid aa \mid bb$
$A \to CDE \mid DE \mid CE \mid a$
$B \to CDE \mid DE \mid CE$
$C \to CDE \mid DE \mid CE$
$D \to CDE \mid BE \mid CE \mid ab$

Step3: By eliminating useless prodns,

$S \to aAa \mid bBb \mid aa \mid bb$

$A \to$

| OV | NV | P |
|---|---|---|
| $\phi$ | S, A, D | $S \to aa \mid bb$<br>$A \to a$<br>$D \to ab$ |
| S, A, D | S, A, D | $S \to aAa$ |

The gs is

$S \to aa \mid bb \mid aAa$
$A \to CDE \mid DE \mid CE \mid a$
$D \to ab$.

The final gs is $S \to aAa \mid bBb$