Camlin Page
Date / /

4/8/17

MODULE - I

FINITE STATE AUTOMATA

Formal language:

* Symbol - Any letter or any digit.
* Alphabet - A finite set of symbols.
  $\Sigma = \{a, b, 0\}$
* String - Sequence of symbols from alphabet.
* Language - Group of strings formed from alphabets.
  - Set of strings over a fixed alphabet
  $\{aa, aab, baa\}$

Empty String or Null String : String with length 0. Denoted as ε

* KLEENE STAR (Σ^*) : Set of string including ε.
  $\Sigma = \{0, 1\}$
  $\Sigma^* = \{\varepsilon, 00, 01..\}$
* KLEENE PLUS (Σ^+) : Set of strings without ε.
  $\Sigma^+ = \{a, b, ab, ba\}$

* $\Sigma^1 = \{a, b\}$
* $\Sigma^2 = \{ab, ba, aa\}$ $\boxed{ab \neq ba}$
* $\Sigma^0 = \{\varepsilon\}$

7/8/17

* $L(G) = \{ w \in \Sigma^* \mid S \xRightarrow[G]{*} w \}$

$L(G)$: Language defined on a grammar G
w : string of terminals

Eg : $S \rightarrow NPVP$ (Production Rule)
$NP \rightarrow$ noun
$VP \rightarrow$ verb , adverb.
noun $\rightarrow$ Geevan | Kavya
verb $\rightarrow$ run , ran
adverb $\rightarrow$ quickly

* Terminals
* Non-terminals
* Start Symbol

* Grammar is defined as a 4-tuple

$$G = (V, \Sigma, P, S)$$

(N) V : Finite non-empty set of non-terminals
(T) Σ : Finite non-empty set of terminals
P : Finite set of production rules.
S : Start symbol

* $V \cap \Sigma = \phi$

Valid production rules : $B \rightarrow Aba$
$AB \rightarrow aba$, $aS \rightarrow ASb$

Scanned by CamScanner

* α : Strings that has got both terminals & non-terminals.

$$\alpha \in V \cup \Sigma$$

Generally $\alpha \xRightarrow[G]{*} \beta$

$\Rightarrow$: derivation $\rightarrow$: production

$*$: multiple times / more than 1 substitution

We get β from α over a number of derivations on the grammar G.

* $\alpha \xRightarrow[G]{} \beta$: over a single derivation.

β is a string derived from α over a number of substitution defined on G.

* $L(G) = \{ w \in \Sigma^* \mid S \xRightarrow[G]{*} w \}$

Consider $S \rightarrow baaS$
$S \rightarrow baa$

$V = \{S\}$ (Non-terminal → capital)
$\Sigma = \{a, b\}$
$S = S$
$P = \{ S \rightarrow baaS, S \rightarrow baa \}$ ① ②

This is G

1) $S \Rightarrow baa$

2) $S \xRightarrow{1} baaS$
$\xRightarrow{1} baabaaS$
$\xRightarrow{1} baabaabaaS$
$\xRightarrow{2} baabaabaabaa$

3) $S \xRightarrow{1} baaS$
$\xRightarrow{2} baabaaS$
$\Rightarrow baabaabaa$

$L(G) = \{ baa, baabaabaabaa, baabaabaa \ldots \}$

Q1. $S \to aSb$
$S \to \varepsilon$

Q2. $S \to aCa$
$C \to aCa \mid b$

Q3. $S \to AB$
$A \to BB$
$B \to AA$

Q4. Following are the productions of a given grammar

$S \to aABa$
$A \to baABb$
$B \to Aab$
$aA \to baa$
$bBb \to abab$

Check whether $w = ba^2b^2aba^3b^2aba$ is in $L(G)$.

Ans 1. $S \to aSb$ ②
$S \to \varepsilon$ ①
$V = \{S\}$
$\Sigma = \{a, b\}$
$P = \{S \to aSb, S \to \varepsilon\}$

Scanned by CamScanner

$S = \{S\}$

1) $S \Longrightarrow \varepsilon$

2) $S \stackrel{1}{\Longrightarrow} aSb$
$\Longrightarrow ab$

3) $S \stackrel{2}{\Longrightarrow} aSb$
$\stackrel{1}{\Longrightarrow} aaSbb$
$\Longrightarrow aabb$

$L(G) = \{\varepsilon, ab, aabb \ldots\}$

2. $S \rightarrow aCa$ ①
$C \rightarrow aCa$ ②
$C \rightarrow b$ ③

$V = \{S, C\}$
$\Sigma = \{a, b\}$
$P = \{S \rightarrow aCa,\ C \rightarrow aCa,\ C \rightarrow b$
$S = \{S\}$

$\cancel{1)\ S \Longrightarrow aCa}$
$\cancel{2)\ C \Longrightarrow aCa}$
$\cancel{1)\ C \Longrightarrow b}$

$S \stackrel{1}{\Longrightarrow} aCa$ $\qquad S \stackrel{3}{\Longrightarrow} aCa$
$\stackrel{3}{\Longrightarrow} aaCaa$ $\qquad \Longrightarrow aba$
$\Longrightarrow aabaa$

$S \stackrel{1}{\Longrightarrow} aCa$
$\stackrel{1}{\Longrightarrow} aaCaa$
$\stackrel{3}{\Longrightarrow} aaaCaaa$
$\Longrightarrow aaabaaa$

$L(G) = \{aba, aabaa, aaabaaa \ldots\}$

3. $S \rightarrow AB$ ①
$A \rightarrow BB$ ②
$B \rightarrow AA$ ③

$S \xRightarrow{2} AB$
$\xRightarrow{3} BBB$
$\xRightarrow{2} AABB$
$\Rightarrow AAAAB$
$\Rightarrow AAAAAA$

4. $S \rightarrow aABa$ ①
$A \rightarrow baABb$ ②
$B \rightarrow Aab$ ③
$aA \rightarrow baa$ ④
$bBb \rightarrow abab$ ⑤

$S \xRightarrow{4} aABa$
$\xRightarrow{3} ba^2Ba$
$\xRightarrow{2} ba^2Aaba$
$\xRightarrow{3} ba^2baABbaba$
$\xRightarrow{4} ba^2baAAab^2aba$
$\xRightarrow{4} ba^2b^2aaAab^2aba$
$\xRightarrow{4} ba^2b^2aba^3b^2aba$
$\Rightarrow ba^2b^2aba^3b^2aba$

8/8/'17 **Finite State Automata :**

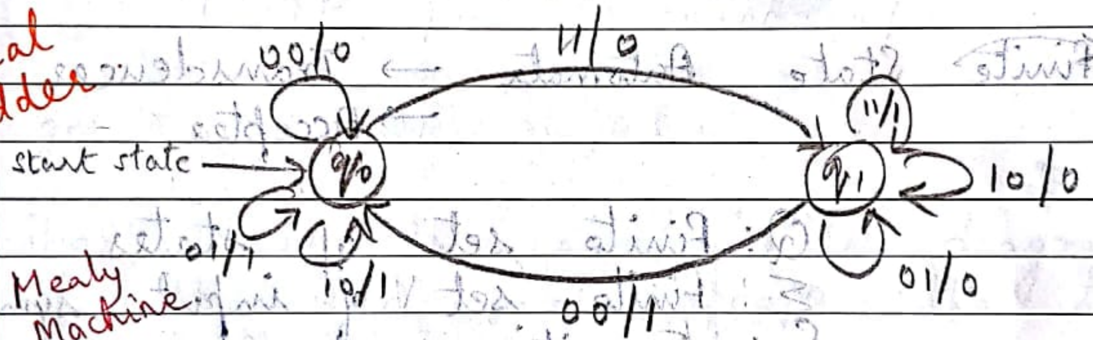
```
A   11011011 +
B   01101110
   101001001
```

Inputs = {00, 01, 10, 11} (Σ)

Output = {0, 1}

Serial Adder

1)



Mealy Machine

q_0 : state without carry
Change in state indicates carry.

```
 0100101 +
 0101110
 1010011
```

10		01		11		01		00		11	
q_0	1	q_0	1	q_0	0	q_1	0	q_1	1	q_0	0	q_1

(00) q_0

By reading it in reverse order, we get 1010011

This is one application - designing a sequential circuit.

* Finite State Automata can be defined as a 5-tuple

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

* finite State Automata → Transducer
→ Accepter

(K) Q : Finite set of states
Σ : Finite set of input symbols
δ : transition
q_0 : initial state
q_0 in Q is the initial state
F : Finite set of final states

'δ' is a mapping from $Q \times \Sigma \rightarrow Q$

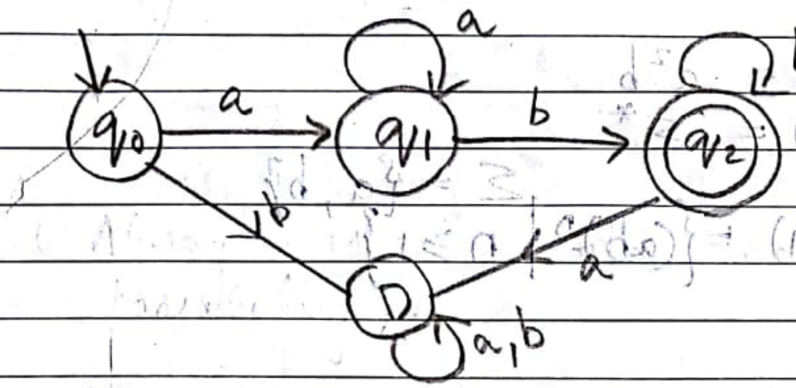
$$\delta(q_0, 00) = q_0$$

* Deterministic Finite State Automata.

$$L(G) = \{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

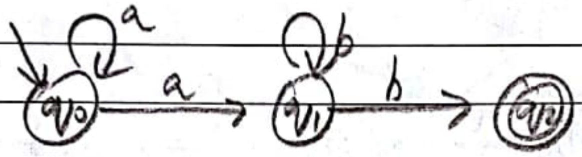
* $L(M) \Rightarrow$ language accepted by machine M.

$$L(M) = \{ a^n b^m \mid n, m \geq 1 \}$$



Non-deterministic FSA

Note



Here on q_0, if we give a, we get 2 states q_0, q_1

The above is called 'state diagram' of the FSA that accepts the language $a^n b^m$.

State Transition Table

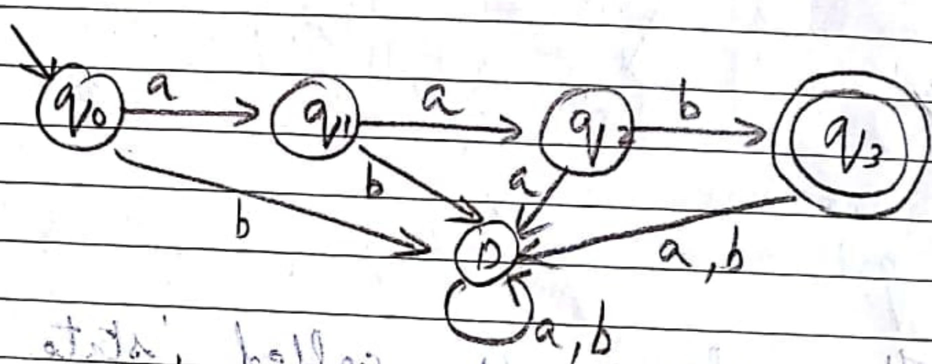
	a	b
$\rightarrow q_0$	q_1	D
q_1	q_1	q_2
q_2	D	q_2
D	D	D

Here, on q_0, if b is given, such an i/p is not allowed. Hence we trap it to a 'Dead' state.

Transitions on the 'D' state comes back to this state only

Q. 1. $L(M) = a^2b$
2. $L(M) = \Sigma^*$
3. $L(M) = \{(ab)^n \mid n \geq 1\}$ $\Sigma = \{a, b\}$

① $L(M) = a^2b$

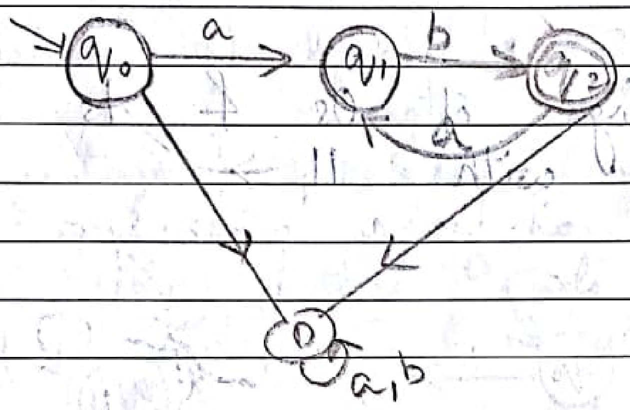


	a	b
→ q_0	q_1	D
q_1	q_2	D
q_2	D	q_3
q_3	D	D
D	D	D

②

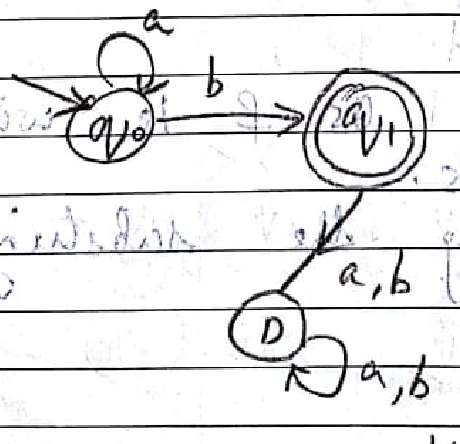
	a	b
q_0	q_0	q_0

(3)



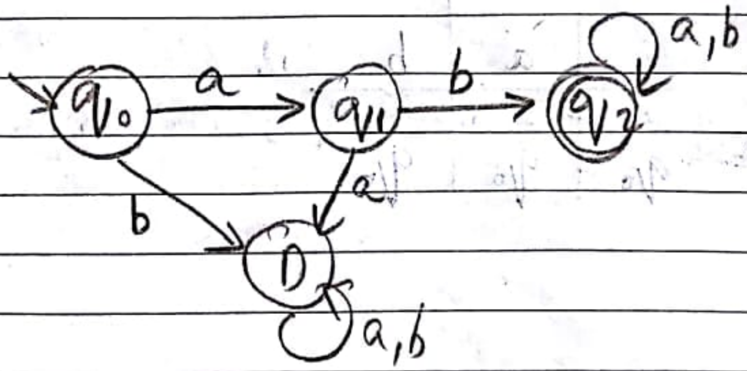
9/8/17

1) $L = \{a^n b \mid n \geq 0\}$



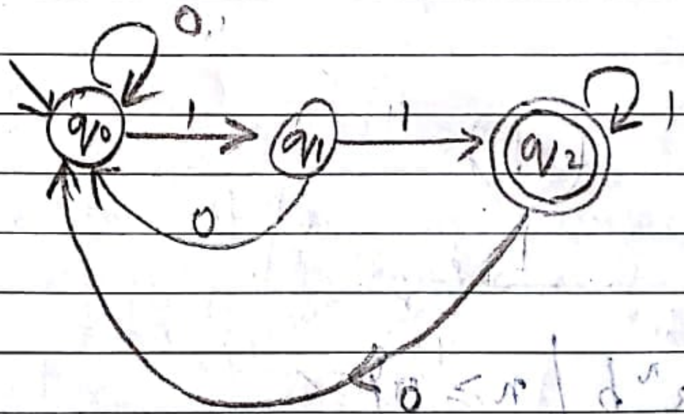
2) Design a DFA that will accept all strings on $\Sigma = \{a, b\}$ starting ~~on~~ with prefix ab.

ab
aba
abba



q. All strings of 0s & 1s that will end with 11

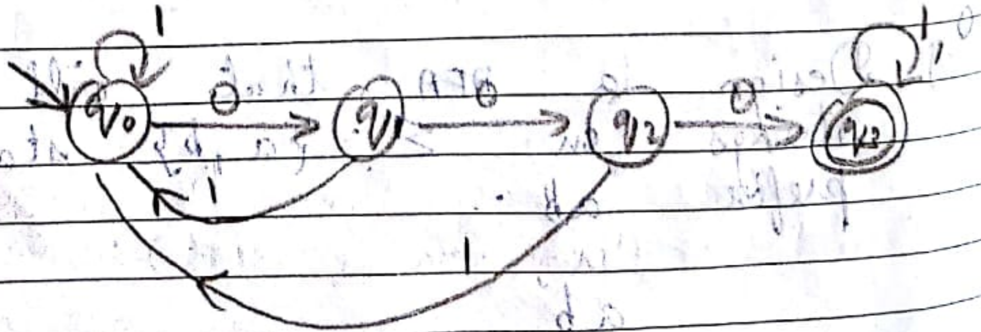
Eg: 1001



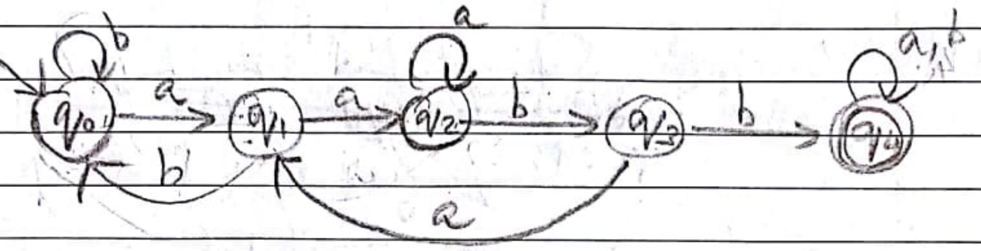
q1. Strings ~~of~~ of 0s & 1s with 3 consecutive 0s.

q2. String containing the substring aabb.

1.



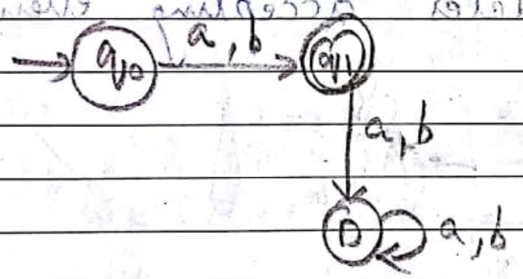
2.



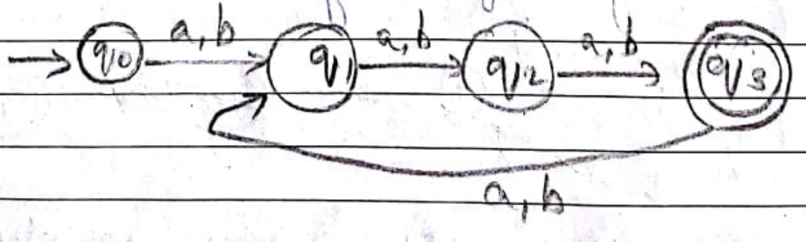
Q2. All strings of length which is a multiple of 3
Q1. All strings of length 1
Q3. Strings containing exactly four 1s.
Q4. Strings containing odd no. of 0s.
Q5. Set of strings that ends with either 00 or 11.

Ans

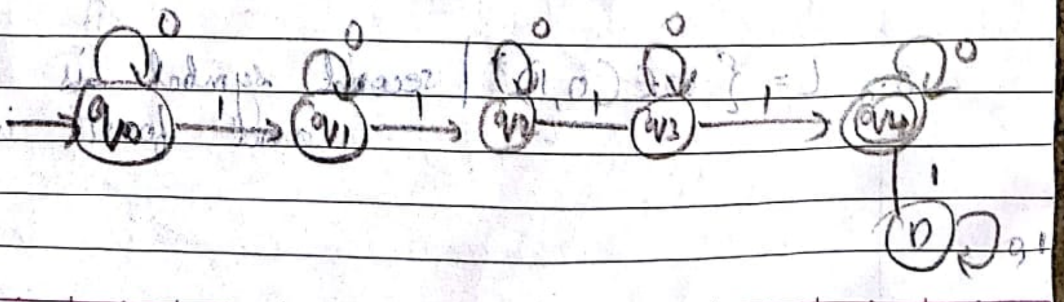
Q1.



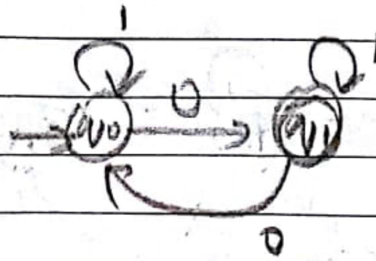
2.



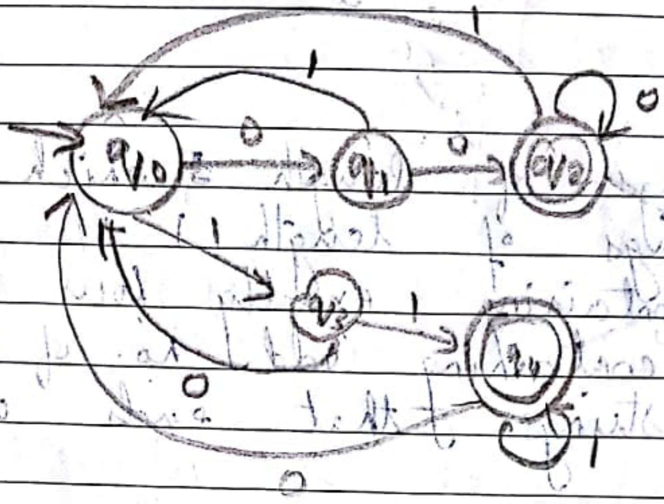
3.



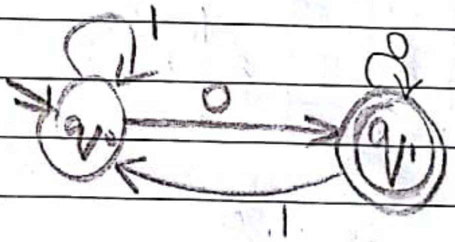
4.



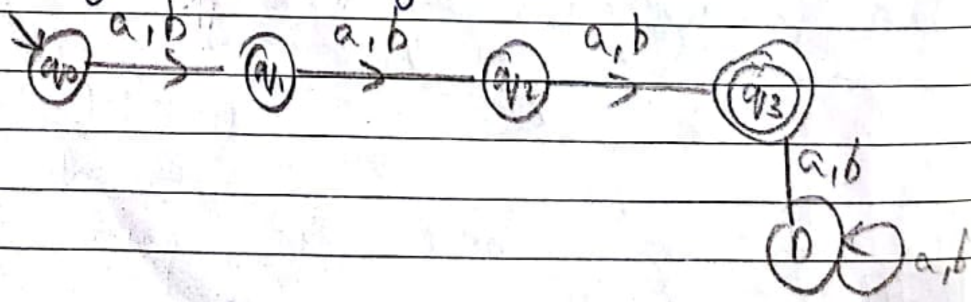
5.



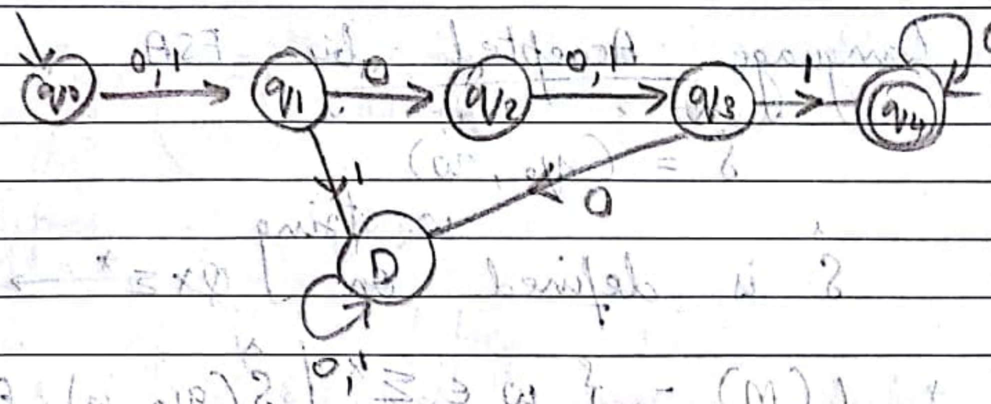
6. FSA with 2 states accepting even number



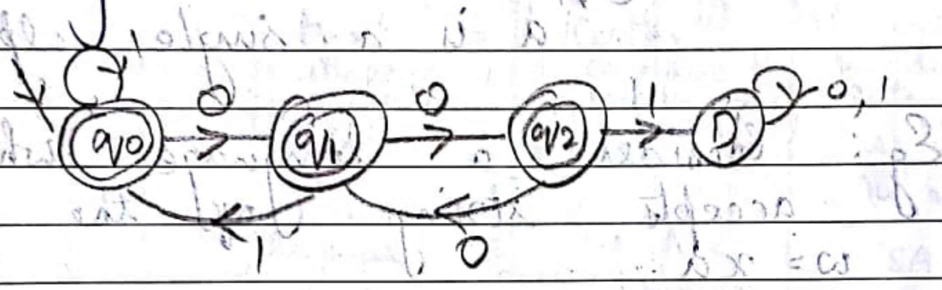
7. All strings of length 3.



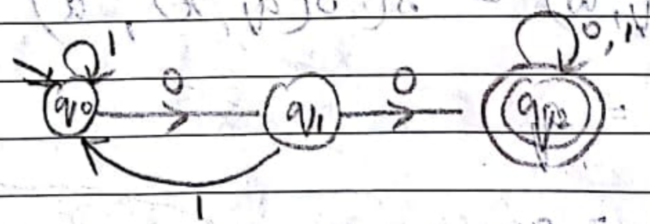
8. $L = \{ w \in (0,1)^* \mid$ second symbol is 0 and fourth input is 1 $\}$



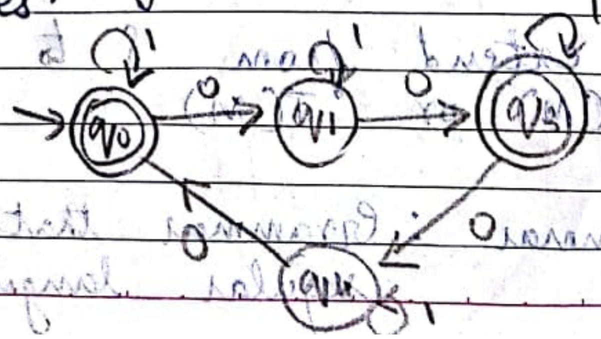
9. All strings except those containing the substring 001.



10. $L = \{ w \in (0,1)^* \mid$ every string w containing 00 as substring $\}$



11. All strings in $\{0,1\}^*$ having even number of zeroes.



14/08/'17

Language Accepted by FSA

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

w: string

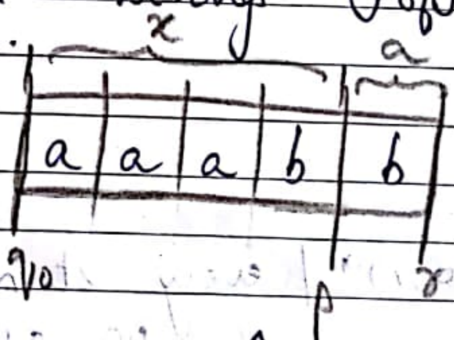
$\hat{\delta}$ is defined on $Q \times \Sigma^* \to Q$.

* $L(M) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$

Whereas $\delta(q, a) = p$

a is a single alphabet.

Eg: Consider a language which accepts strings of the form $w = xa$.



If $r \in F$, it is accepted. Otherwise not.

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

* $\delta(q, \epsilon) = q$

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

String of ϵs.

* We have extend from δ to $\hat{\delta}^*$.

L(M) or T(M)

* Regular Grammar : Grammar that supports regular language.

* Language accepted by FSA is called "regular languages". ($L(M)$)

Types of Grammar

It is of 4 types:

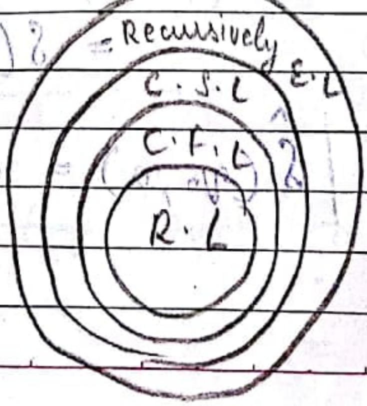
	TYPES	Grammar	Language	Machine	Production rules
1.	Type 0	Unrestricted Grammar	Recursively Enumerable Languages	Turing machine	$\alpha \to \beta$
2.	Type 1	Context sensitive grammar	Context sensitive languages	Linear bounded Automata	$\alpha A \beta \to \alpha \gamma \beta$
3.	Type 2	Context free grammar	Context free languages	Pushdown Automata	$A \to \alpha$
4.	Type 3	Regular Grammar	Regular Language	FSA	$A \to aB$ $A \to a$

It was defined by "Noam Chomsky".

* $\alpha A \beta \to \alpha \gamma \beta$ which means the replacement of A by γ can be done in the context of $\alpha\beta$

Length of α ie $|\alpha| > |\beta|$ for $\alpha \to \beta$

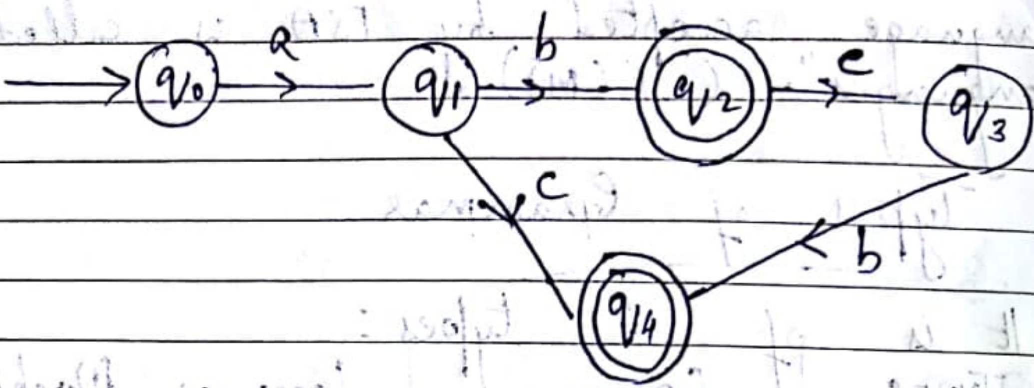
Non-Deterministic Finite State Automata (NDFA)



Most powerful → turing m/c
Least powerful → FSA

16/8/'17

Q.



$$Q = \{q_0, q_1, q_2, q_4, q_3\}$$

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

$$\delta(q_0, a) = q_1$$
$$\delta(q_1, b) = q_2$$
$$\vdots$$

$$F = \{q_2, q_4\}$$

$$\hat{\delta}(q_0, w) = \hat{\delta}(q_0, xa)$$
$$= \delta(\hat{\delta}(q_0, x), a)$$

Method 1:

$$\Rightarrow \hat{\delta}(q_0, abc) = \delta(\hat{\delta}(q_0, ab), c) \quad -1$$
$$\hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b) \quad -2$$

$$\hat{\delta}(q_0, a) = \hat{\delta}(q_0, \varepsilon a)$$
$$= \delta(\hat{\delta}(q_0, \varepsilon), a)$$
$$= \delta(q_0, a)$$

$$\Rightarrow \boxed{\hat{\delta}(q_0, a) = \delta(q_0, a)}$$

$\Downarrow$

(In FSA)

Camlin Page

Date / /

$$\hat{\delta}(q_0, a) = \delta(q_0, a) = q_1 \quad \text{--- } 3$$

Using 3 in 2

$$\hat{\delta}(q_0, ab) = \delta(q_1, b)$$
$$= q_2$$

Again using (i)

$$\hat{\delta}(q_0, abc) = \delta(q_2, c)$$
$$= q_3$$

$q_3 \notin F$.

$\therefore$ The string is not accepted.

Now consider abcb

$$\delta(q_0, abcb) = \delta(\delta(q_0, abc), b)$$
$$= \delta(q_3, b)$$
$$= q_4$$

$q_4 \in F$.

$\therefore$ The string is accepted.

Method 2:

$\hat{\delta}(q_0, abc)$

$$\hat{\delta}(q_0, a) = \hat{\delta}(q_0, \varepsilon a)$$
$$= \delta(\hat{\delta}(q_0, \varepsilon), a)$$
$$= \delta(q_0, a) = q_1$$

$$\hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b)$$
$$= \delta(q_1, b)$$
$$= q_2$$

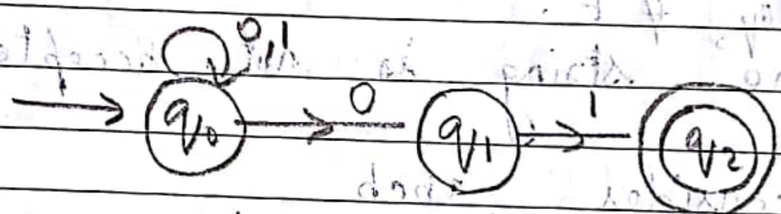
$$\hat{\delta}(q_0, abc) = \delta(\hat{\delta}(q_0, ab), c)$$
$$= \delta(q_2, c)$$
$$= q_3$$

$q_3 \notin F$

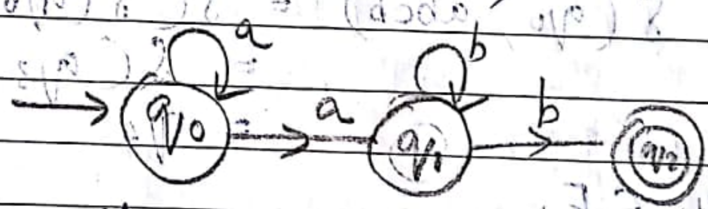
Non-Deterministic Finite State Automata

If there are 0, 1 or more transitions from same state for the same i/p it is an NFA.

Eg:

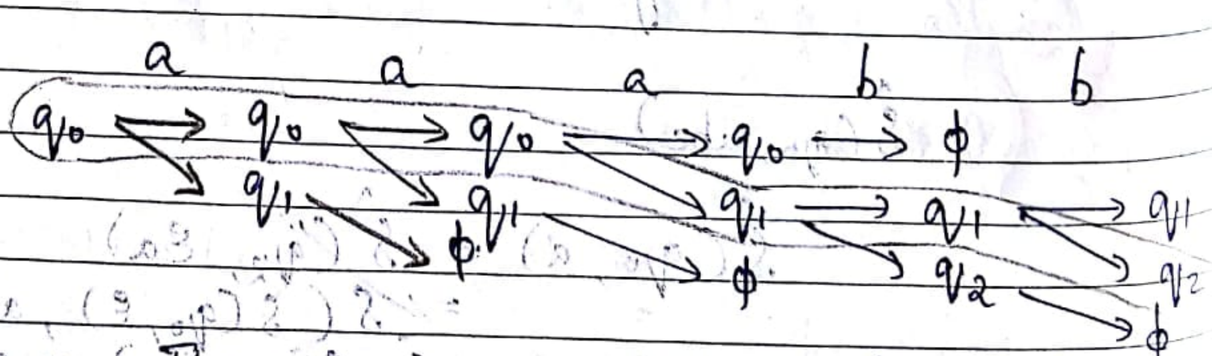


Consider $a^n b^m$, $n, m \geq 1$



Make both a & b NFA

i/p : a a a b b



There is 1 path which lands up in F.

Definition :

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

Here δ is a mapping from $Q \times \Sigma \rightarrow 2^Q$

Consider $Q = \{q_1, q_2, q_3\}$

Subsets : $\{\{q_1, q_2, q_3\}, \{q_1, q_2\}, \{q_2, q_3\}, \{q_1, q_3\}, \phi, \{q_1\}, \{q_2\}, \{q_3\}\}$ = Power Set

Here 2^Q is the set of all subsets of Q. The transition will always be to a set from the power set.

Q. Check whether 01101 is accepted by the example

$$\hat{\delta}(q_0, 01101) = \delta(\hat{\delta}(q_0, 0110), 1)$$

$$\hat{\delta}(q_0, 0110) = \delta(\hat{\delta}(q_0, 011), 0)$$

$$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$$

$$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$$

$$\hat{\delta}(q_0, 0) = \hat{\delta}(q_0, \varepsilon 0)$$

$$= \delta(\hat{\delta}(q_0, \varepsilon), 0)$$

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

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

$$\hat{\delta}(q_0, 01) = \delta(\{q_0, q_1\}, 1)$$

Note: $\delta(P, a) = \bigcup_{q \in P} \delta(q, a)$

$\hat{\delta}(q_0, 01) = \delta(\{q_0, q_1\}, 1)$
$= \delta(q_0, 1) \cup \delta(q_1, 1)$
$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 011) = \delta(\{q_0, q_2\}, 1)$
$= \delta(q_0, 1) \cup \delta(q_2, 1)$
$= \{q_0\} \cup \phi$
$= \{q_0\}$

$\hat{\delta}(q_0, 0110) = \delta(\{q_0\}, 0)$
$= \{q_0, q_1\}$

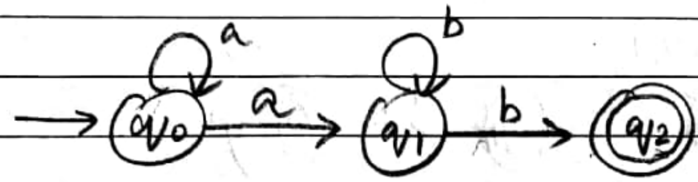
$\hat{\delta}(q_0, 01101) = \delta(\{q_0, q_1\}, 1)$
$= \delta(q_0, 1) \cup \delta(q_1, 1)$
$= \{q_0, q_2\}$

21/8/17

Both NFA & DFA are equally powerful.

CONVERSION OF NFA TO DFA

1. Construct an NFA for the language $a^n b^m$ for $n, m \geq 1$



The method for conversion is called "Subset Construction"

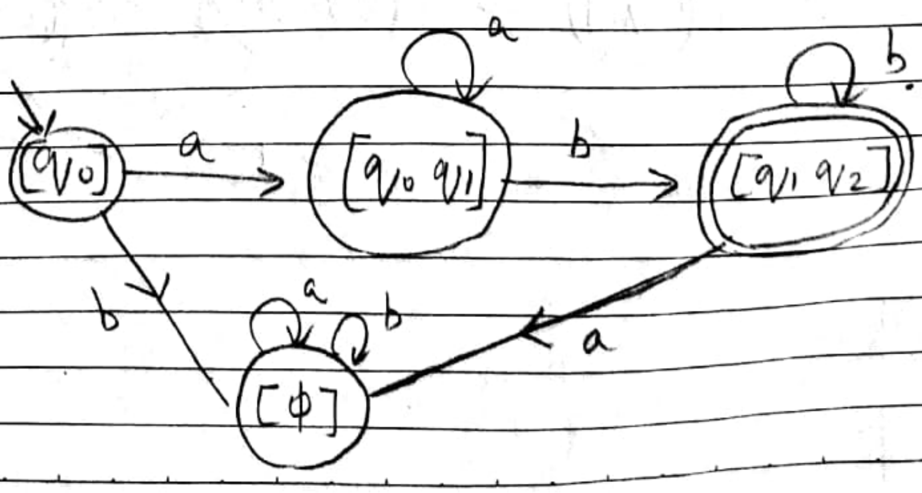
	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	ϕ
q_1	ϕ	$\{q_1, q_2\}$
q_2	ϕ	ϕ

* State table for NFA

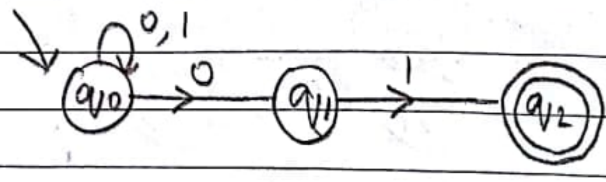
DFA

	a	b
$\rightarrow [q_0]$	$[q_0\, q_1]$	$[\phi]$
$[q_0\, q_1]$	$[q_0\, q_1]$	$[q_1, q_2]$
$[\phi]$	$[\phi]$	$[\phi]$
$[q_1\, q_2]$	$[\phi]$	$[q_1\, q_2]$

$\rightarrow \delta(q_0, a) \cup \delta(q_1, a)$
$\{q_0, q_1\} \cup \phi$
$\{q_0, q_1\}$



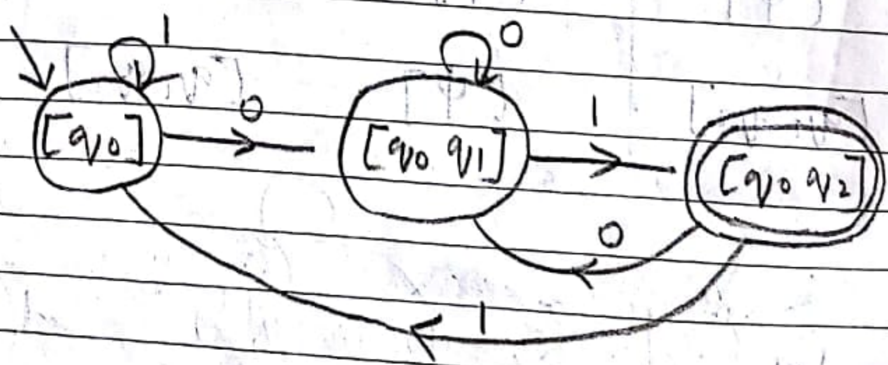
2.



	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
q_2	ϕ	ϕ

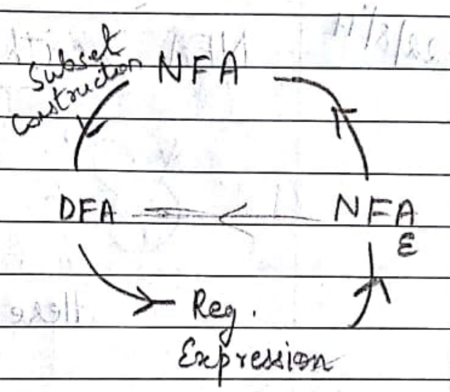
DFA

	0	1
→ $[q_0]$	$[q_0\ q_1]$	$[q_0]$
$[q_0\ q_1]$	$[q_0\ q_1]$	$[q_0\ q_2]$
$[q_0\ q_2]$	$[q_0\ q_1]$	$[q_0]$

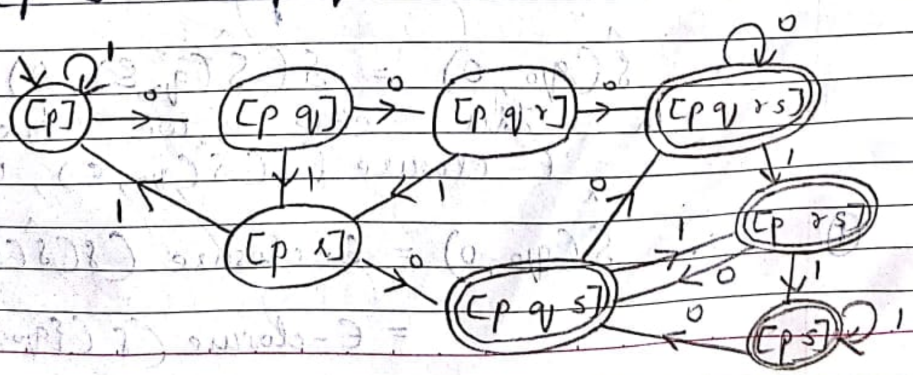


3.

	0	1
→ p	{p, q}	{p}
q	{r}	{r}
r	{s}	ϕ
S	{s}	{s}

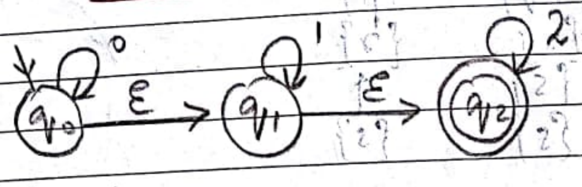


	0	1
→ [p]	[p q]	[p]
[p q]	[p q r]	[p r]
[p q r]	[p q r s]	[p r]
[p r]	[p q s]	[p]
✓ [p q r s]	[p q r s]	[p r s]
✓ [p q s]	[p q r s]	[p r s]
✓ [p r s]	[p q s]	[p ~~r~~ s]
✓ [ps]	[p q s]	[p s]



22/8/17

NFA with ε Transition:



Here
$$Q = \{q_0, q_1, q_2\}$$

* ε- Closure (q_0)

Set of all states reachable from it just by reading ε (through ε moves), including that state itself.

ε- closure $(q_0) = \{q_0, q_1, q_2\}$

ε-closure $(q_1) = \{q_1, q_2\}$

ε-closure $(q_2) = \{q_2\}$

$$\boxed{\hat{\delta}(q, \varepsilon) = \varepsilon\text{-Closure}(q)}$$

Consider the example

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \varepsilon), 0)$

In this case, we take
ε- closure $(\delta(\hat{\delta}(q_0, \varepsilon), 0))$

(1) $\therefore \hat{\delta}(q_0, 0) = \varepsilon$- closure $(\delta(\hat{\delta}(q_0, \varepsilon), 0))$

$= \varepsilon$-closure $(\delta(\{q_0, q_1, q_2\}, 0))$

$= \epsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$
$= \epsilon\text{-closure}(\{q_0\} \cup \phi \cup \phi)$
$= \epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$

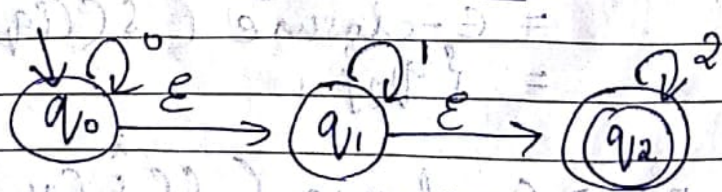
$\therefore q_2 \in F$, 0 is accepted by the language $0^m 1^n 2^p,\ m, n, p \geq 0$.

② $\hat{\delta}(q_0, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \varepsilon), 1))$
$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 1))$
$= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$
$= \epsilon\text{-closure}(\phi \cup \{q_1\} \cup \phi)$
$= \epsilon\text{-closure}(q_1)$
$= \{q_1, q_2\}$

$\because q_2 \in F$, 1 is accepted.

③ $\hat{\delta}(q_0, 2) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \varepsilon), 2))$
$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 2))$
$= \epsilon\text{-closure}(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$
$= \epsilon\text{-closure}(\phi \cup \phi \cup \{q_2\})$
$= \epsilon\text{-closure}(\{q_2\})$
$= \{q_2\}$

CONVERSION OF NFA WITH ε INTO NFA without ε



ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

	0	1	2	ϵ
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

(4) $\hat{\delta}(q_2, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 0))$
$= \epsilon\text{-closure}(\delta((\{q_2\}), 0))$
$= \epsilon\text{-closure}(\delta(q_2, 0))$
$= \epsilon\text{-closure}(\phi)$
$= \phi$

(5) $\hat{\delta}(q_2, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 1))$
$= \epsilon\text{-closure}(\delta(\{q_2\}, 1))$
$= \epsilon\text{-closure}(\phi)$
$= \phi$

(6) $\hat{\delta}(q_2, 2) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 2))$
$= \epsilon\text{-closure}(\delta((\{q_2\}), 2))$
$= \{q_2\}$

(7) $\hat{\delta}(q_1, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), 0))$
$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 0))$

$= \epsilon\text{-closure}\,(\delta(q_1, 0) \cup \delta(q_2, 0))$
$= \epsilon\text{-closure}\,(\phi \cup \phi)$
$= \phi$

⑧ $\hat{\delta}(q_1, 1) = \epsilon\text{-closure}\,(\delta(\hat{\delta}(q_1, \varepsilon), 1))$
$= \epsilon\text{-closure}\,(\delta(\{q_1, q_2\}, 1))$
$= \epsilon\text{-closure}\,(\{q_1\} \cup \phi)$
$= \{q_1, q_2\}$

⑨ $\hat{\delta}(q_1, 2) = \epsilon\text{-closure}\,(\delta(\hat{\delta}(q_1, \varepsilon), 2))$
$= \epsilon\text{-closure}\,(\delta(\{q_1, q_2\}, 2))$
$= \epsilon\text{-closure}\,(\phi \cup \{q_2\})$
$= \{q_2\}$

NFA without ε

	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

Q. Check if 012, 011 are accepted (NFA with ε)

① $\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

$\hat{\delta}(q_0, 0) = \epsilon\text{-closure}\,(\delta(\hat{\delta}(q_0, \varepsilon), 0))$
$= \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0, 01) = \delta(\{q_0, q_1, q_2\}, 1)$
$= \epsilon.c(\{q_1\})$

$= \{q_1, q_2\}$

$\hat{\delta}(q_0, 011) = E \cdot C \{ \delta(\{q_1, q_2\}, 1) \}$

$= E \cdot C \{q_1\}$

$= \{q_1, q_2\}$

② $\hat{\delta}(q_0, 012) = \delta(\hat{\delta}(q_0, 01), 2)$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

$\hat{\delta}(q_0, 0) = E \cdot C \{ \delta(\hat{\delta}(q_0, \varepsilon),$

$= E \cdot C \{ \delta(\{q_1, q_2, q_0$

$= E \cdot C \{q_0\}$

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

$\hat{\delta}(q_0, 01) = E \cdot C \{ \delta(\{q_0, q_1, q_2\}, 1$

$= E \cdot C \{q_1\}$

$= \{q_1, q_2\}$

$\hat{\delta}(q_0, 012) = E \cdot C (\delta(\{q_1, q_2\}, 2$

$= E \cdot C \{q_2\}$

$= \{q_2\}$

25

25/8/17

3 a)

δ	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_4	q_2
q_2	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

b)

δ	0	1
q_0	q_0	$\{q_1, q_2\}$
q_1	q_2	q_2
q_2	ϕ	ϕ

c)

δ	0	1
$\rightarrow p$	$\{p, r\}$	$\{q\}$
q	$\{r, s\}$	$\{p\}$
r	$\{p, s\}$	$\{r\}$
s	$\{q, r\}$	ϕ

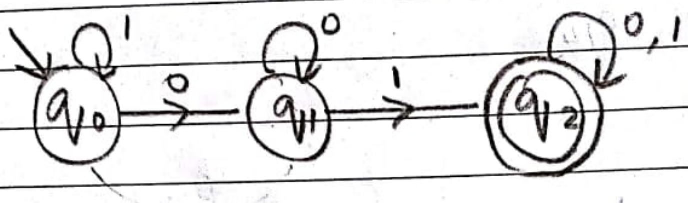
d)

δ	ε	a	b
p	$\{r\}$	$\{q\}$	$\{p, r\}$
q	ϕ	$\{p\}$	ϕ
r	$\{p, q\}$	$\{r\}$	$\{p\}$

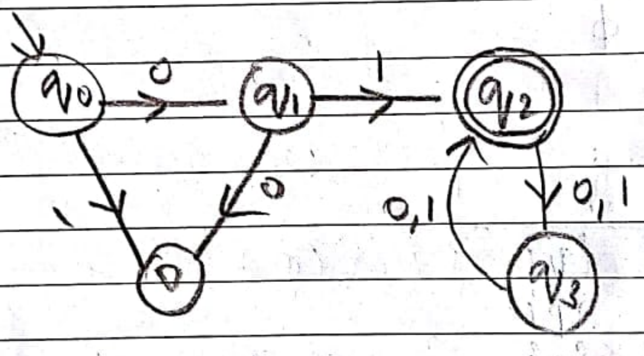
TUTORIAL

1) Design a DFA that accepts all strings

a) with substring 01.

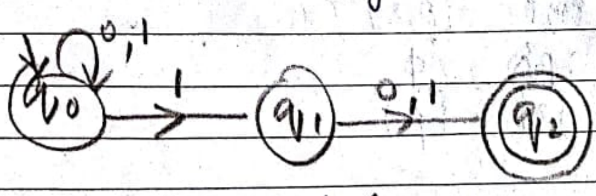


b) of even length & begins with 01.

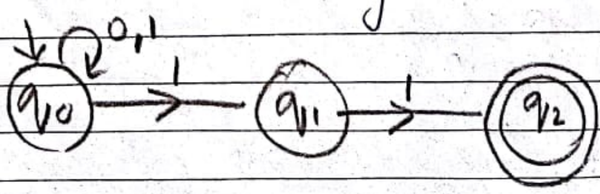


2) Design an NFA that accepts set of all strings

a) whose 2nd last symbol is 1.



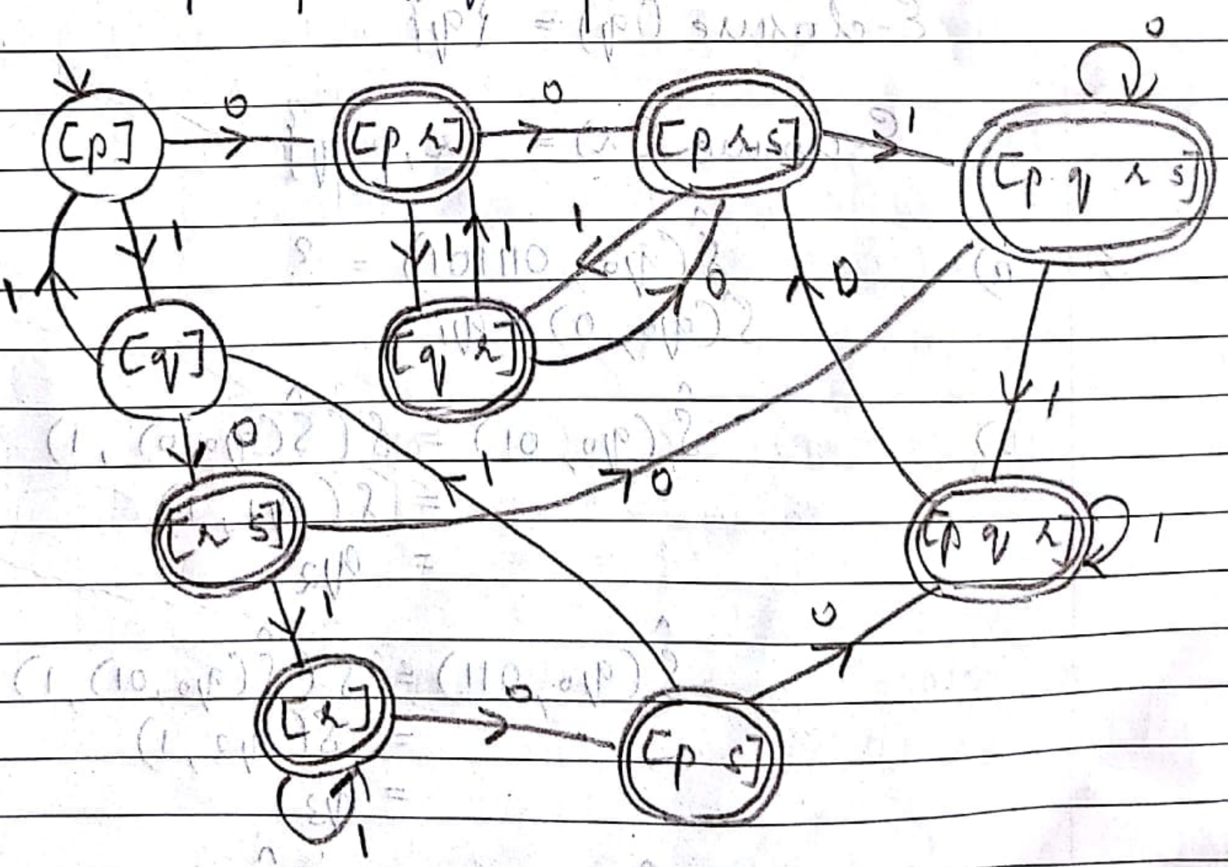
b) End with substring 11.



4) Convert the following NFA to DFA

DFA :

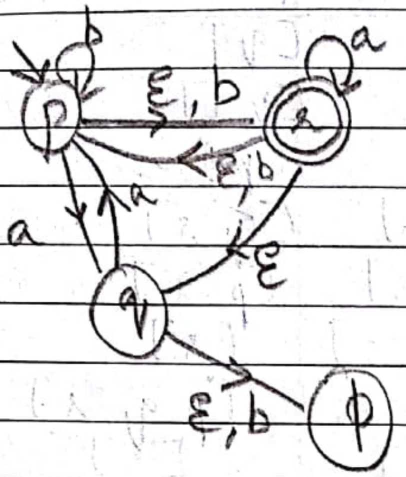
δ	0	1
→ [p]	[p r]	[q]
[p r]	[p r s]	[q r]
[q]	[r s]	[p]
[p r s]	[p q r s]	[q r]
[q r]	[p r s]	[p r]
[r s]	[p q r s]	[r]
[p q r s]	[p q r s]	[p q r]
[r]	[p s]	[r]
[p q r]	[p r s]	[p q r]
[p s]	[p q r]	[q]



Scanned by CamScanner

5.

δ	ε	a	b
$\rightarrow p$	$\{r\}$	$\{q\}$	$\{p, r\}$
q	ϕ	$\{p\}$	ϕ
r	$\{p, q\}$	$\{r\}$	$\{p\}$



a) $\varepsilon\text{-closure}(p) = \{p, r, q\}$

$\varepsilon\text{-closure}(q) = \{q\}$

$\varepsilon\text{-closure}(r) = \{r, p, q\}$

3. a) $\hat{\delta}(q_0, 011101) = ?$

$\delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$
$= \delta(q_1, 1)$
$= q_2$

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$
$= \delta(q_2, 1)$
$= q_3$

$\hat{\delta}(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1)$
$= \delta(q_3, 1) = q_2$

$$\hat{\delta}(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0)$$
$$= \delta(q_2, 0)$$
$$= q_3$$

$$\hat{\delta}(q_0, 011101) = \delta(\hat{\delta}(q_0, 01110), 1)$$
$$= \delta(q_3, 1)$$
$$= q_2$$

$q_2 \in F$.

$\therefore$ The string is accepted.

b) $\hat{\delta}(q_0, 01010) = ?$

$$\hat{\delta}(q_0, 0) = \delta(q_0, 0) = q_0$$

$$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$$
$$= \delta(q_0, 1)$$
$$= \{q_1, q_2\}$$

$$\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0)$$
$$= \delta(\{q_1, q_2\}, 0)$$
$$= q_2$$

$$\hat{\delta}(q_0, 0101) = \delta(\hat{\delta}(q_0, 010), 1)$$
$$= \delta(q_2, 1)$$
$$= \phi$$

$$\hat{\delta}(q_0, 01010) = \delta(\hat{\delta}(q_0, 0101), 0)$$
$$= \delta(\phi, 0)$$
$$= \phi$$

$\therefore$ The string is not accepted.

Camlin Page
Date / /

26/8/'17

REGULAR EXPRESSIONS

3 basic primitive expressions ϕ, ε, $a \in \Sigma$
These are used to form new regular expressions with following operators.
Operators +, ·, *

+ : union
· : concatenation
* : closure

Language L_1, L_2
Find $L_1 + L_2$, $L_1 \cdot L_2$, $L_1 *, L_2 *$
$L_1 = \{001, 10, 111\}$
$L_2 = \{\varepsilon, 001\}$
$L_1 + L_2 = \{001, 10, 111, \varepsilon\} = L_3$
$L_1 \cdot L_2 = \{001, 001001, 10, 10001, 111, 111001\}$

Note: * Consider $L_1 = \{0, 1\}$
$L_1^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$

Consider $L = \{0, 11\}$
$L^* = \{\varepsilon, 0, 11, 00, 000, 011, 110, 1111, 0011, 1100, 0000, 0110 \ldots\}$

* 01 is 0.1
Here there is only 1 string 01
* 0+1
Here the language accepts either 0 or 1.
* 0^* $\{\varepsilon, 0, 00, \ldots\}$
* $(0+1)^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 001, \ldots 110, 111 \ldots\}$

If $\Sigma = \{0, 1\}$, this is same as Σ^*

Scanned by CamScanner

* $0(0+1)^*$

Set of all strings that starts with zero.

* Find Regular expression that accepts strings that ends with 11.

$(0+1)^* 11$

* Set of strings which have atleast one consecutive 11

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

* Set of strings which have atleast 1 consecutive pair of 0s or 1s.

$(0+1)^* (00+11) (0+1)^*$

* Set of string that ends with either 00 or 11

$(0+1)^* (00+11)$

* Set of all strings which has no 2 consecutive 1s.

0^*.

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

Set of strings ~~starting~~ ending with 1 or ending with 0. $\{1, 01, 001, \ldots, 0, 10, 110$

* $(0+1)^* + 1$ ($L_1 = (0+1)^*$ $L_2 = 1$

$\{\varepsilon, 1, 0, 01, 11, 001, 111, \ldots\}$

29/8/'17

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

Language is 0^*1 or 1.

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

* $(0+1)^* \cdot 1$

$L_3 = \{1, 01, 11, 001, \ldots\}$

$L_1 = (0+1)^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$

$L_2 = 1$

$L_1 \cdot L_2 = L_3 = (0+1)^* \cdot 1$

* $((0+1)^* \cdot 1)^*$ $\quad L_4 = L_3^*$

$L_4 = \{\varepsilon, 1, 11, 01, 001, 101 \ldots\}$

Note: $L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3$

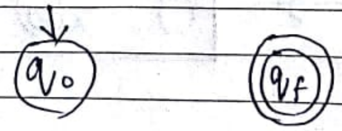
CONVERTING REGULAR EXPRESSION INTO NFA WITH ε

Let Σ be a given alphabet then,

1) ~~for~~ ϕ, ε, $a \in \Sigma$ are all regular expressions. These expressions are called primitive regular expressions.
2) If R_1 and R_2 are regular expressions, so are $R_1 + R_2$, $R_1 \cdot R_2$, $R_1{*}$ and (R_1)
3) A string is a regular expression iff it can be derived from the primitive regular expression by a finite no. of applications of the rules mentioned in 2.

FSA for:

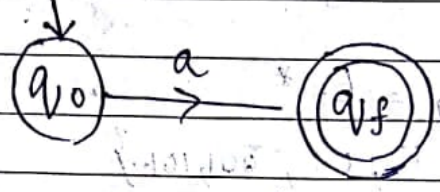
1) ϕ



2) ε

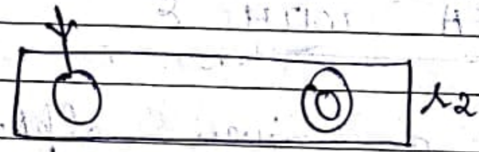
$L = \{\varepsilon\}$

3) $a \in \Sigma$

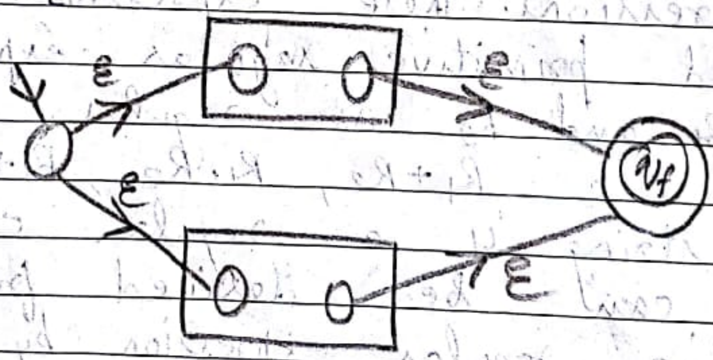


Consider r_1 and r_2 (two regular expression

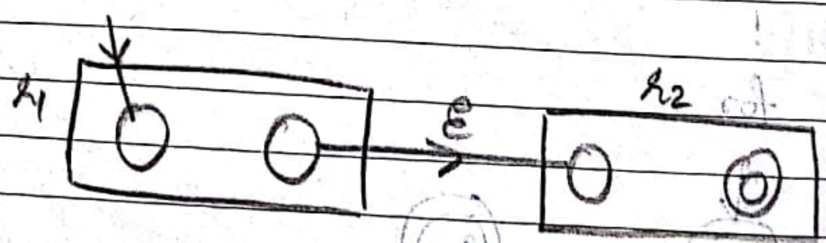
(capsule)



1) $r_1 + r_2$

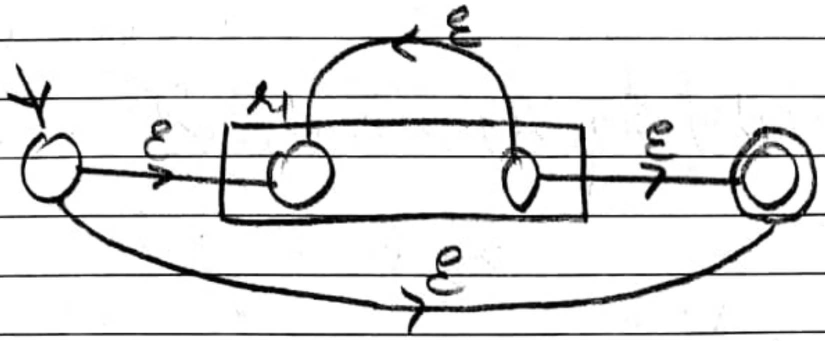


2) $r_1 r_2$



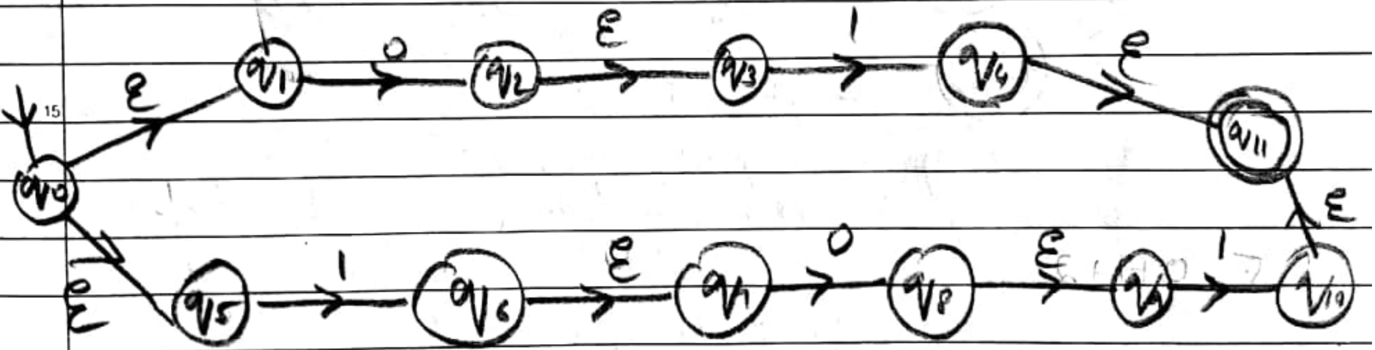
3) r_1^*

$r_1^* = r_1^0 \cup r_1^1 \cup \ldots$

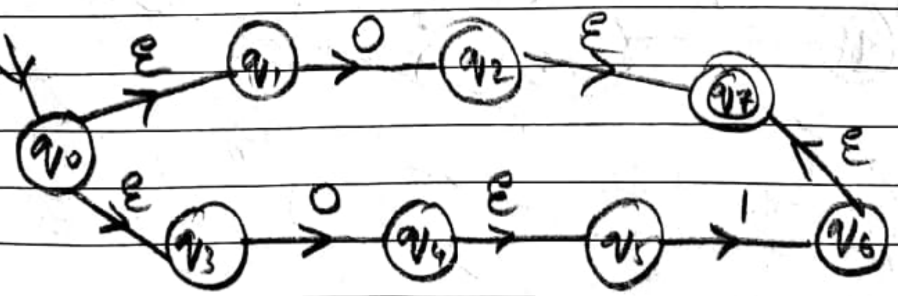


Q. Convert the following RE into NFA with εs.

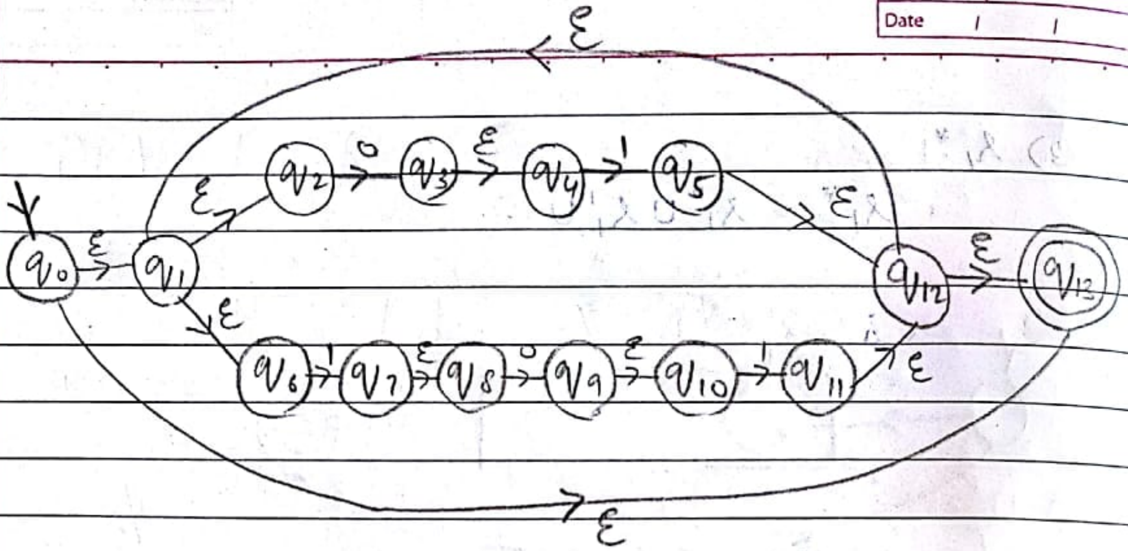
1) 01+101



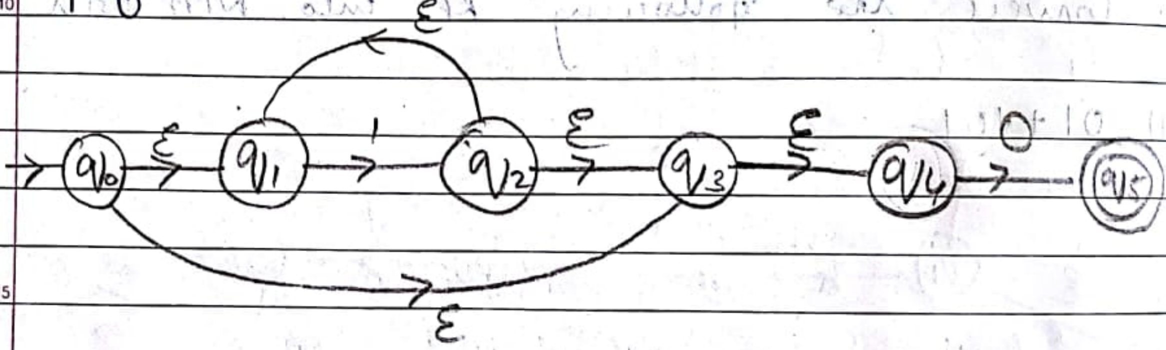
2) 0+01



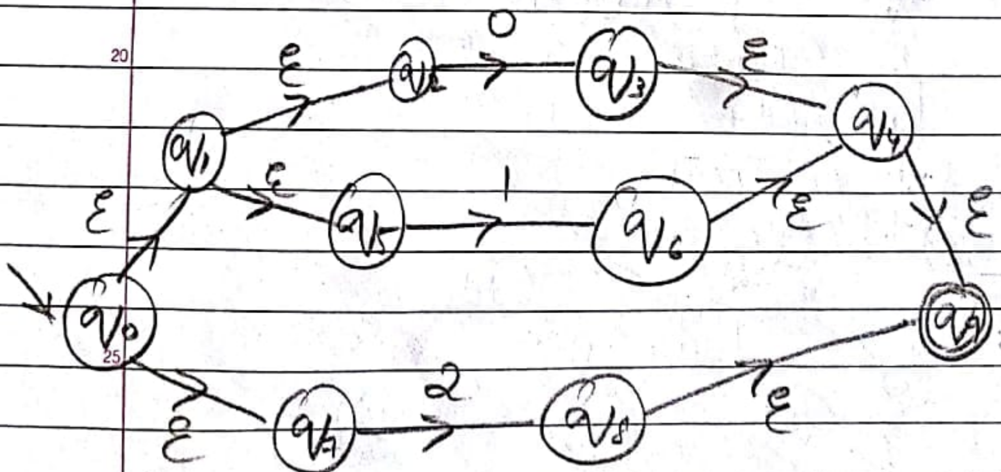
3) $(01+101)^*$



4. 1^*0



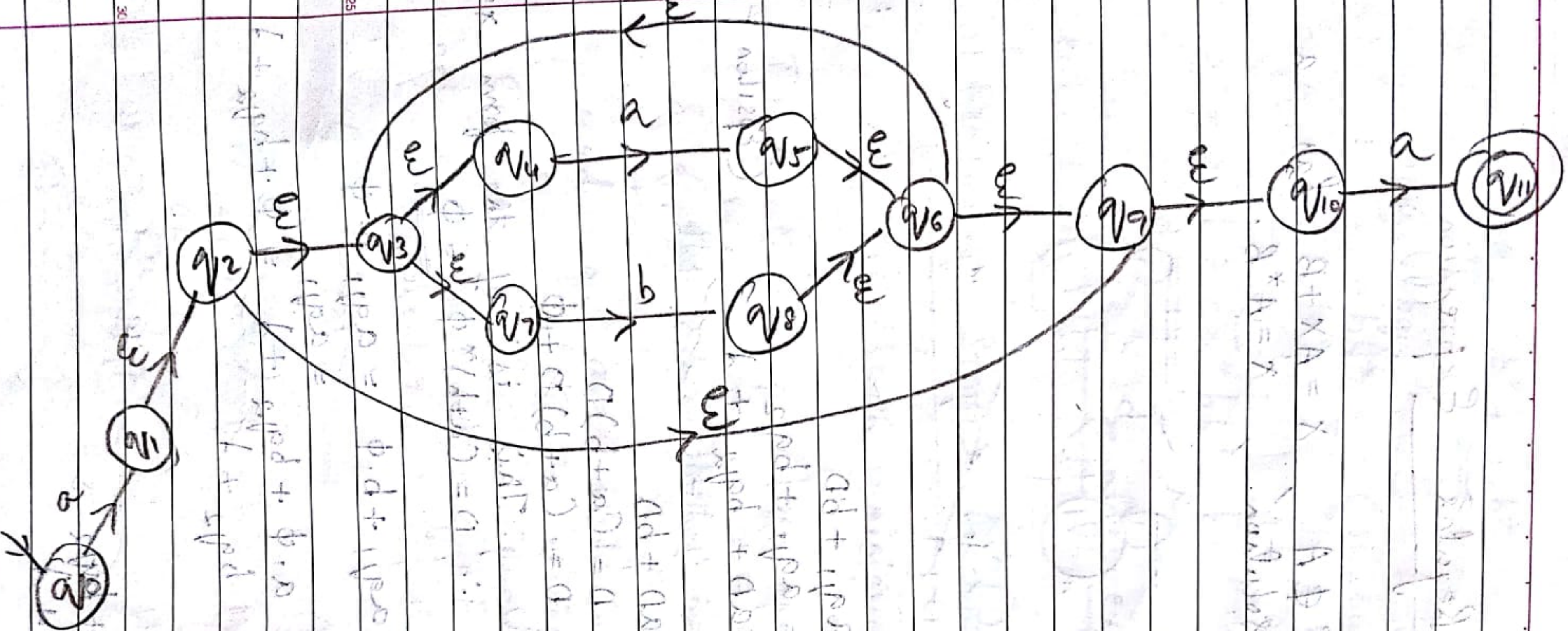
5. $0+1+2$



6. $(a(a+b)^*a)$

Camlin Page

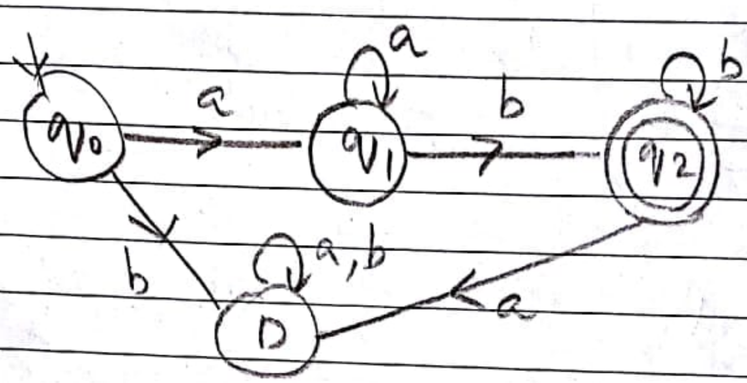
Date / /



DFA to Regular Expression

Lemma:

If $\varepsilon \notin A$, $X = AX + B$ has a unique solution, $X = A^*B$.



$$q_0 = aq_1 + bD$$
$$q_1 = aq_1 + bq_2$$
$$q_2 = aD + bq_2 + \lambda \qquad (\text{Epsilon} - \varepsilon/\lambda)$$
$\hookrightarrow$ (final state)

$$D = aD + bD$$
$$\Longrightarrow D = (a+b)D$$
$$\Longrightarrow D = (a+b)D + \phi$$

This is of the form $X = AX + B$

$$\therefore D = (a+b)^*\phi = \underline{\underline{\phi}}$$

$$\therefore q_0 = aq_1 + b.\phi = aq_1 + \phi$$
$$= aq_1$$

$$q_2 = a.\phi + bq_2 + \lambda = \phi + bq_2 + \lambda$$
$$= bq_2 + \lambda$$

$$q_0 = aq_1$$
$$q_1 = aq_1 + bq_2$$
$$q_2 = bq_2 + \lambda$$

$$q_2 = b^*\lambda = b^*$$

$$q_1 = aq_1 + bq_2$$
$$= aq_1 + bb^* \quad \text{(This is of the form } X = AX + B\text{)}$$
$$q_1 = a^*bb^*$$

$$q_0 = aq_1 = aa^*bb^*$$

This is the regular expression corresponding to the DFA.

15/09/17

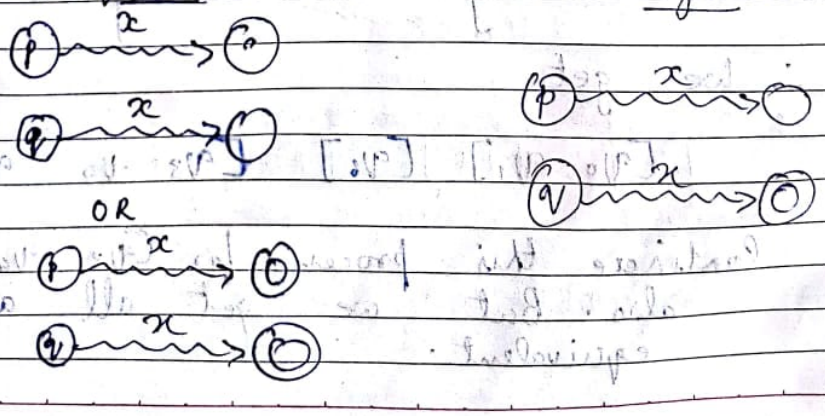
MINIMUM STATE AUTOMATA

An FSA with minimum number of states.

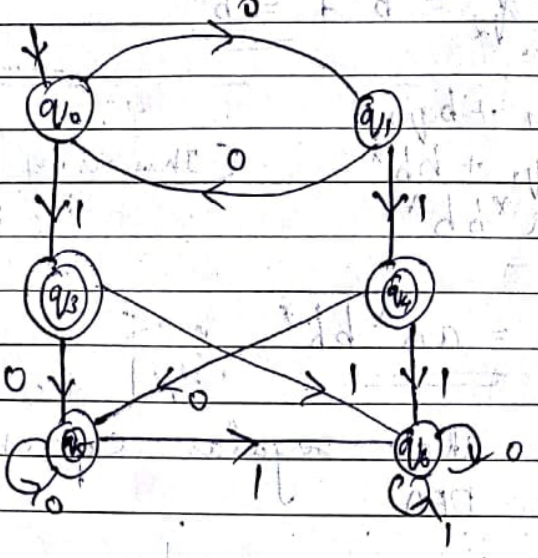
* Two states p & q are equivalent iff for any string $x \in \Sigma^*$, $\hat{\delta}(p, x)$ and $\hat{\delta}(q, x)$ are both in F or both not in F.

* p and q are said to be distinguishable if there exists a string x such that $\hat{\delta}(p, x)$ is an element of F & $\hat{\delta}(q, x)$ not an element of F and vice-versa.

Equivalent | Distinguishable



Eg:



FSA accepting strings having a single 1.

→ $q_0 \quad q_1 \quad q_3 \quad q_4 \quad q_5 \quad q_6$

Step 1: (First partitioning based on F or not).

→ $[q_0 \; q_1 \; q_6] \qquad [q_3 \; q_4 \; q_5]$

Step 2:

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

$\delta(q_0, 1) = q_3$ (F) $\quad \delta(q_1, 1) = q_4$ (F)

q_0 & q_1 are equivalent

$\delta(q_0, 0) = q_1$ NF $\quad \delta(q_6, 0) = q_6$ NF

$\delta(q_0, 1) = q_3$ F $\quad \delta(q_6, 1) = q_6$ NF

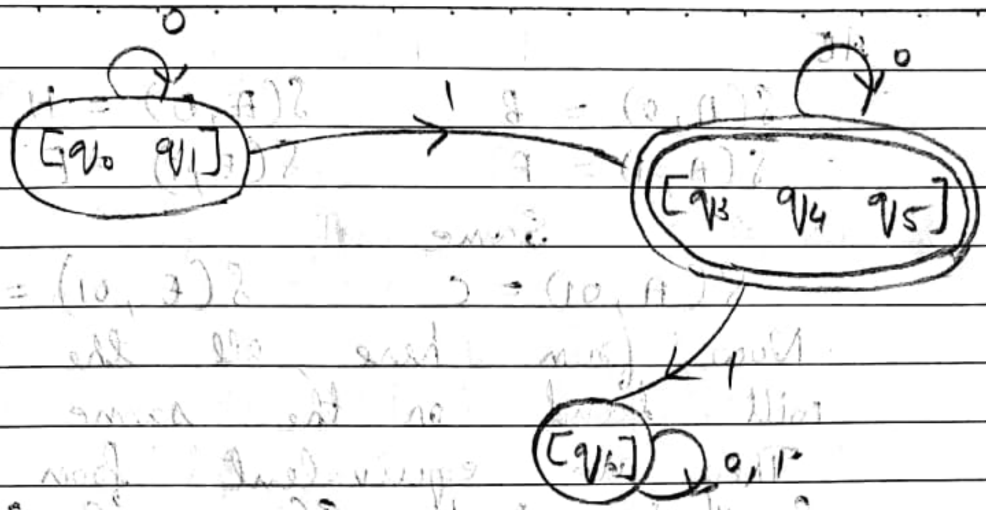
q_0 & q_6 are not equivalent.

Hence they cannot belong to 1 partition

∴ We get

$[q_0 \; q_1] \quad [q_6] \quad [q_3 \; q_4 \; q_5]$

Continue this process for $[q_3 \; q_4 \; q_5]$ also. But we get all are equivalent.



.	0	1
→ A	B	F
B	G	C
* C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

Minimize the given FSA.

A B C D E F G H

Step 1: [C] [A B D E F G H]

Step 2: AB

$\delta(A,0) = B$ NF $\delta(B,0) = G$ NF

$\delta(A,1) = F$ NF $\delta(B,1) = C$ F

A and B are not equivalent

AD

$\delta(A,0) = B$ NF $\delta(D,0) = C$ F

$\delta(A,1) = F$

Not equivalent

AE

$\delta(A,0) = B \qquad \delta(E,0) = H$

$\delta(A,1) = F \qquad \delta(E,1) = F$

Same

$\delta(A,01) = C \qquad \delta(E,01) = C$

Now, from here, all the transitions will land on the same state. They are equivalent from this point

Similarly check $\delta(A,01)$ & $\delta(E,01)$ and also for 00, 11.

They are equivalent

AF

$\delta(A,0) = B$ NF $\quad \delta(F,0) = C$ F

They are not equivalent

AG

$\delta(A,0) = B \qquad \delta(G,0) = G$

$\delta(A,1) = F \qquad \delta(G,1) = E$

$\delta(A,01) = C \qquad \delta(G,01) = E$

They are not equivalent.

AH

$\delta(A,0) = B \qquad \delta(H,0) = G$

$\delta(A,1) = F \qquad \delta(H,1) = C$

They are not equivalent

[C] [AE] [BDFGH]

BD

$\delta(B,0) = G \qquad \delta(D,0) = C$

Not equivalent

17/10/21

BF

$\delta(B,0) = G \quad \delta(F,0) = C$

They are not equivalent.

BG

$\delta(B,0) = G \quad \delta(G,0) = G$

$\delta(B,1) = C \quad \delta(G,1) = E$

Not equivalent

BH

$\delta(B,0) = G \quad \delta(H,0) = G$

$\delta(B,1) = C \quad \delta(H,1) = C$

Equivalent

[C] [AE] [BH] [DFG]

DF

$\delta(D,0) = C \quad \delta(D,1) = G$

$\delta(F,0) = C \quad \delta(F,1) = G$

Equivalent

[C] [AE] [BH] [DFG]

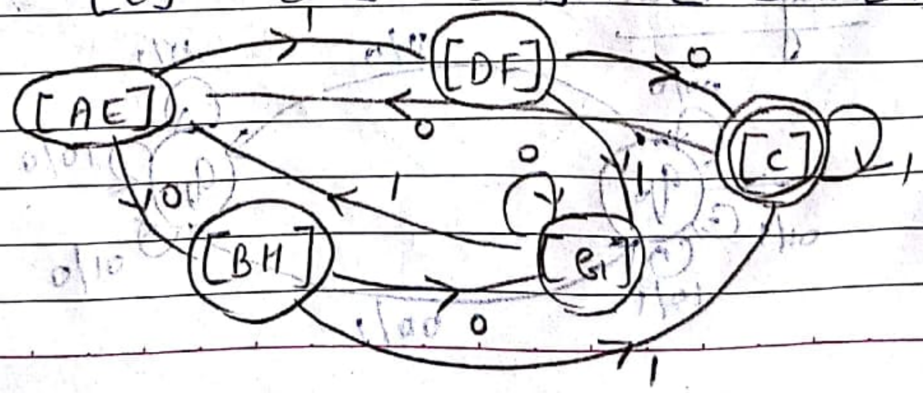
DG

$\delta(D,0) = C \quad \delta(G,0) = G$

$\delta(D,1) = G \quad \delta(G,1) = E$

Not equivalent

[C] [AE] [BH] [DF] [G]



16/09/17

	A	B	C	D	E	F	G
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	
H	X		X	X	X	X	X

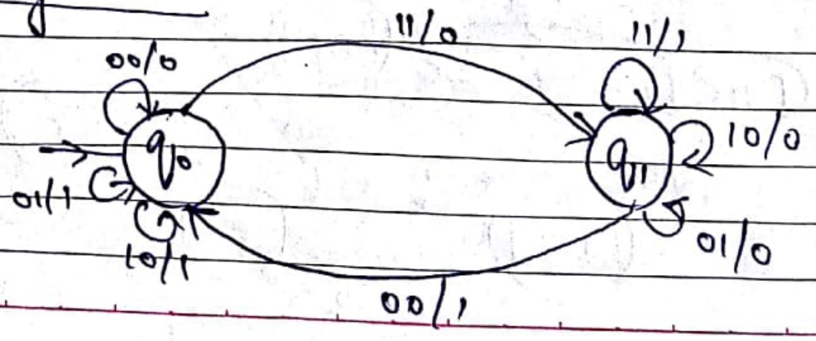
	0	1
[A, E]	[B, H]	[F]
[C]	[A]	[C]
[B, H]	[G]	[C]
[D, F]	[C]	[G]
[G]	[G]	[E]

FINITE STATE AUTOMATA WITH OUTPUT

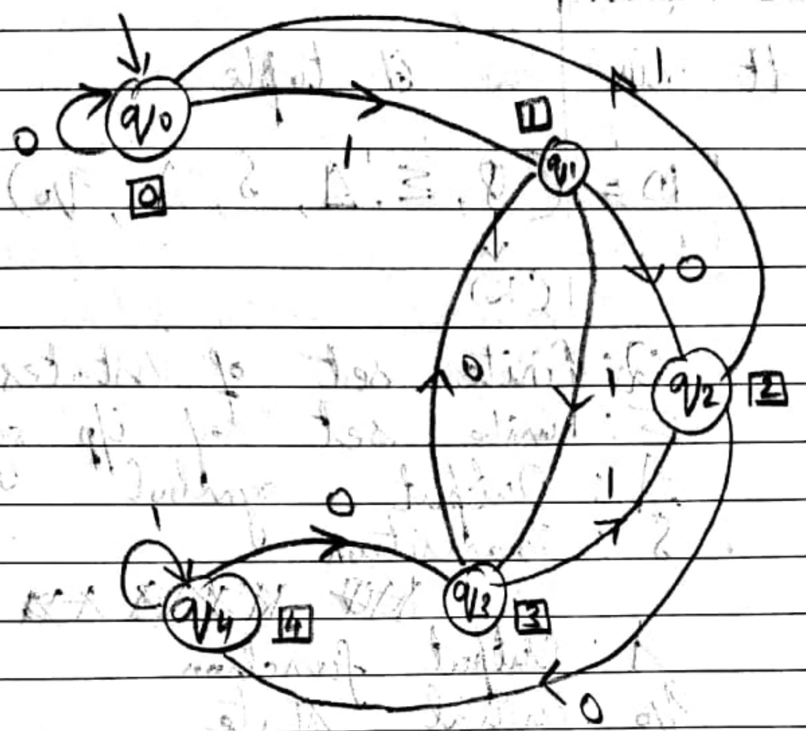
In a Mealy machine, transition depends on current state & input.

In Moore machine, the state alone is required to reach the output.

Mealy Machine



Moore Machine



Outputs ⇒ [0] [1] [2] [3] [4]

2	0010
5	0101
7	0111

Consider 2 0010 (i/p) ⇒ (o/p) [2]

5 0101 (i/p) ⇒ [0]

7 0111 (i/p) ⇒ [2]

12 1100 (i/p) ⇒ [2]

10 1010 (i/p) ⇒ [0]

(The operation is $n \% 5$).

Eg : 0101

q_0 → q_0 q_1 q_2 q_0

0 1 2 0

Formal description

It is a 6 tuple.

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

$Q \to (K)$

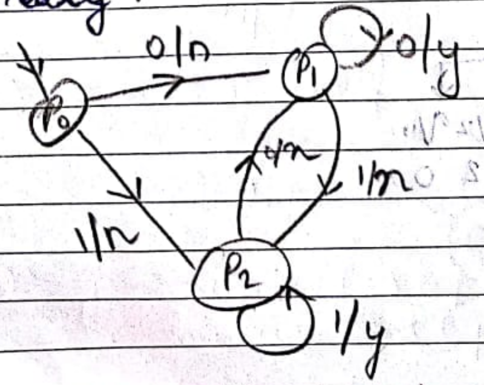
Q : Finite set of states
Σ : Finite set of i/p symbols
Δ : Output symbol
δ : Transition $\quad K \times \Sigma \to K$
λ : Output function
q_0 : initial state

* λ determines if it is a Mealy or Moore machine.

For Moore machine $\quad \lambda : K \to \Delta$
For Mealy machine $\quad \lambda : K \times \Sigma \to \Delta$

Two Way Finite Automata

Eg for Mealy :



y : consecutive
n : otherwise

Camlin Page
Date / /

Two way FSA: It is an abstract machine, generalized version of DFA which can revisit characters already processed.
There are finite number of states with transition b/w them based on current i/p. But each transition is also labelled with a value indicating whether the machine will move its position in the i/p to the left, right or stay at the same position.

$$M = (Q, \Sigma, \vdash, \dashv, \delta, s, t, r)$$

left & right end markers. Not i/p symbol

$$\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \rightarrow Q \times \{L, R\}$$

L, R : Direction in which we wish to move
s: start state
t: end state
r: reject state
$r \neq t$

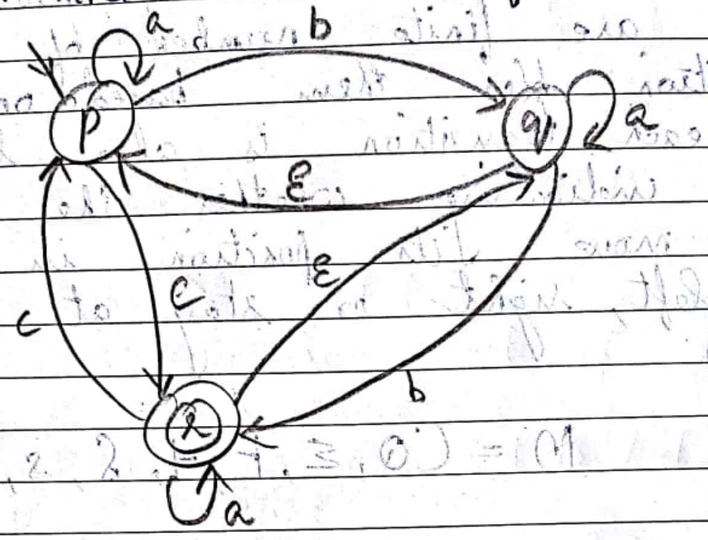
In addition to this, two conditions should also be satisfied.

$$\delta(q, \vdash) = (q', R)$$
$$\delta(q, \dashv) = (q', L)$$

for some $q \in Q$.

NFA with ε to NFA

Step 1: Find ε-closure of each state in FSA.



$\varepsilon\text{-closure}(p) = \{p\}$

$\varepsilon\text{-closure}(q) = \{q, p\}$

$\varepsilon\text{-closure}(r) = \{r, q, p\}$

$$\hat{\delta}(p, a) = \epsilon\text{-closure}\,[\hat{\delta}(p, a)]$$
$$= \epsilon\text{-closure}\,[\delta(\underbrace{\hat{\delta}(p, \varepsilon)}_{\varepsilon\text{ closure}}, a)]$$
$$= \epsilon\text{-closure}\,[\delta(\{p\}, a)]$$
$$= \epsilon\text{-closure}\,[\delta(p, a)]$$
$$= \{p\}$$

$$\hat{\delta}(p, b) = \epsilon\text{-closure}\,[\hat{\delta}(p, b)]$$
$$= \epsilon\text{-closure}\,[\delta(p, b)]$$
$$= \{q, p\}$$

$$\hat{\delta}(p, c) = \epsilon\text{-closure}\,[\hat{\delta}(p, c)]$$
$$= \epsilon\text{-closure}\,[\delta(p, c)]$$
$$= \{p, q, r\}$$

$$\hat{\delta}(q,a) = \epsilon\text{-closure}(\hat{\delta}(q,a)]$$
$$= \epsilon\text{-closure}[\delta(\{p,q\},a)]$$
$$= \epsilon\text{-closure}[\{p\} \cup \{q\}]$$
$$= \epsilon\text{-closure}\{p\} \cup \epsilon\text{-closure}\{q\}$$
$$= \{p,q\}$$

$$\hat{\delta}(q,b) = \epsilon\text{-closure}(\hat{\delta}(q,b))$$
$$= \epsilon\text{-closure}[\delta(\{q,p\},b)]$$
$$= \epsilon\text{-closure}[\{q\} \cup \{r\}]$$
$$= \{p,q,r\}$$

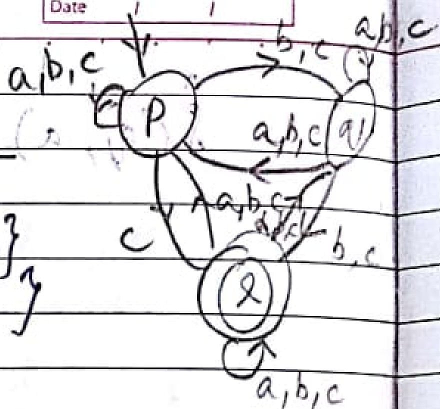
$$\hat{\delta}(q,c) = \epsilon\text{-closure}(\hat{\delta}(q,c))$$
$$= \epsilon\text{-closure}[\delta(\{p,q\},c)]$$
$$= \epsilon\text{-closure}[\{r\} \cup \{\ \}]$$
$$= \{p,q,r\}$$

$$\hat{\delta}(r,a) = \epsilon\text{-closure}[\hat{\delta}(r,a)]$$
$$= \epsilon\text{-closure}[\delta(\{p,q,r\},a)]$$
$$= \epsilon\text{-closure}[\{p,q,r\}]$$
$$= \{p,q,r\}$$

$$\hat{\delta}(r,b) = \epsilon\text{-closure}[\hat{\delta}(r,b)]$$
$$= \epsilon\text{-closure}[\delta(\{p,q,r\},b)]$$
$$= \epsilon\text{-closure}[\{q,r\}]$$
$$= \{p,q,r\}$$

$$\hat{\delta}(r,c) = \epsilon\text{-closure}[\hat{\delta}(r,c)]$$
$$= \epsilon\text{-closure}[\delta(\{p,q,r\},c)]$$
$$= \epsilon\text{-closure}\{r \cup \phi \cup p\}$$
$$= \{p\} \cup \{q,r\} = \{p,q,r\}$$

States	Inputs a	b	c
$\rightarrow p$	$\{p\}$	$\{p,q\}$	$\{p,q,r\}$
q	$\{p,q\}$	$\{p,q,r\}$	$\{p,q,r\}$
$*r$	$\{p,q,r\}$	$\{p,q,r\}$	$\{p,q,r\}$



NFA with ε to DFA

1) Take initial state.
2) Find ε-closure $\Rightarrow$ set of states (1st state in DFA)
3) Apply δ on the set of states for every i/p $\Rightarrow$ new states / state
4) Take ε-closure
   (This state is a state in DFA.)
   Continue (3) & (4) until all new states are formed.

States	Inputs ε	a	b	c
$\rightarrow p$	ϕ	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	ϕ
$*r$	$\{q\}$	$\{r\}$	ϕ	$\{p\}$

ε-closure $(p) = \{p\}$

Step 1: $\delta(p, a) = \varepsilon(\{p\}) = \{p\}$

$\delta(p, b) = \varepsilon(\{q\}) = \{q, p\}$

$\delta(p, c) = \varepsilon(\{r\}) = \{r, p, q\}$

Step 2: $\delta(\{p, q\}, a) = \varepsilon\text{-clos}(\{p, q\})$
$= \{p, q\}$

25/09/17

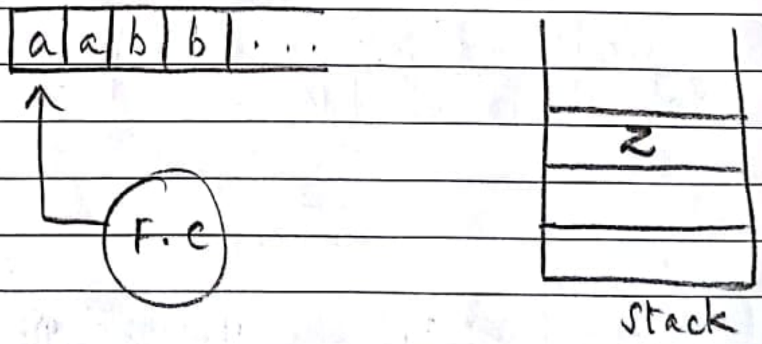
PUSHDOWN AUTOMATA

In this automata, we have a stack which is used to memorize certain things.

Eg: $a^n b^n$
wcw^R ⇒ Eg: 0110110

PDA has 3 components
1) input tape
2) finite control
3) stack

In PDA, there are 3 things which decides what comes next
→ current state, input, stack top element (q, a, z)



A PDA accepts a string in two methods:
1) Empty store (Empty i/p string, empty stack)
2) Final state (i/p string over, ends up in a final state, stack may have some elements)

Eg: wcw^R

Top plate	State	0		C
Blue	q_1	Add a blue plate remain in q_1	Add a green plate remain in q_1	go to q_2
	q_2	Remove blue plate. Remain in q_2	Halt	Halt
Green	q_1	Add a blue plate. q_1	Add a green plate. q_1	Go to q_2
	q_2	Halt	Remove green plate. q_2	Halt
Red	q_1	Add a blue plate. q_1	Add a green plate. q_1	Go to q_2.
	q_2	without waiting for input, remove red plate.		

Deterministic PDA

Initial stack top Z_0 = R

Possible stack elements R G B

Blue - 0
Green - 1

C

+
+
0

+
+
0

Tally

Eg i/p: 011 C 110

$q_1 q_1 q_1 q_2 q_2 q_2$

q_2 ~~q_1~~	~~G~~
q_1	~~G~~
q_1	~~B~~
	~~R~~

$\Rightarrow$ The i/p string is over, stack is empty

Stack becomes empty.

$\therefore$ The string is accepted using empty store method.

i/p $\Sigma = \{0, 1, c\}$

States $q_1 : w$ $q_2 : w^R$

Formal Definition of PDA:

It is a 7-tuple

$M = (K, \Sigma, \Gamma, \delta, q_0, z_0, F)$

K: finite set of states

z_0: initial stack symbol

For empty store method, F is empty

Γ: set of stack symbol

(In empty store acceptance, $F = \phi$)

$\delta : (K \times \Sigma \times \Gamma) \rightarrow (K \times \Gamma^*)$

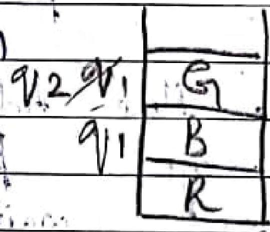
Q1. 01C01

Q2. 01C101

Q3. 011C11

Ans:

1. 01 C 01

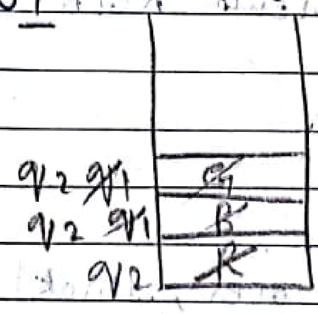


$\delta(q_1, 0, R) = \{(q_1, BR)\}$

$\delta(q_1, 1, B) = \{(q_1, GB)\}$

Halts. Not accepted.

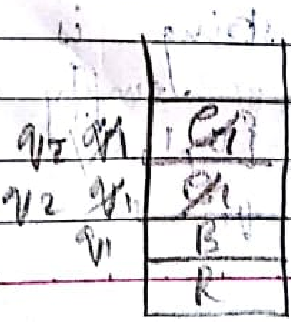
2. 01C101



1 is left to be processed

Hence it can't be accepted.

3. 011C11



I/P is over but stack is not empty. Hence, not accepted.

For wcw^R.

$$Q = \{q_1, q_2\}$$
$$\Sigma = \{0, 1, c\}$$
$$\Gamma = \{R, G, B\}$$
$$z_0 = R$$
$$F = \phi$$

$$\delta : K \times \Sigma \times \Gamma \rightarrow K \times \Gamma^*$$
$$(q, a, z) \longrightarrow (p, \underset{\downarrow}{\gamma})$$

string of stack symbols

PDA $\Rightarrow$ NPDA

1. $\delta(q_1, 0, R) = \{(q_1, BR)\}$
2. $\delta(q_1, 1, R) = \{(q_1, GR)\}$
3. $\delta(q_1, c, R) = \{(q_2, R)\}$
4. $\delta(q_1, 0, B) = \{(q_1, BB)\}$
5. $\delta(q_1, 1, B) = \{(q_1, GB)\}$
6. $\delta(q_1, c, B) = \{(q_2, B)\}$
7. $\delta(q_1, 0, G) = \{(q_1, BG)\}$
8. $\delta(q_1, 1, G) = \{(q_1, GG)\}$
9. $\delta(q_1, c, G) = \{(q_2, G)\}$

10. $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$

11. $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

12. $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$

26/09/17

Q. Design a PDA for $a^n b^n$

$$\Sigma = \{a, b\}$$
$$\Gamma = \{R, a\}$$
$$K = \{q_1, q_2\}$$
$$z_0 = R$$

1) $\delta(q_1, a, R) = \{(q_1, aR)\}$
2) $\delta(q_1, a, a) = \{(q_1, aa)\}$
3) $\delta(q_1, b, a) = \{(q_2, \varepsilon)\}$
4) $\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$
5) $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$

This is for acceptance by empty store.

For final state acceptance we have

$\delta(q_2, \varepsilon, R) = \{(q_f, \varepsilon)\}$

INSTANTANEOUS DESCRIPTION (ID)

- It is expressed as a triplet

(q, w, γ)

q: current state
w: portion of i/p to be read
γ: content of the stack

Camlin Page
Date / /

Suppose $(q, aw, z\gamma)$ is an ID   Eg: 011c110
If $\delta(q, a, z)$ contains (p, β), $\underset{a}{1}\underset{w}{c110}$
then the next ID will be

$$(q, aw, z\gamma) \vdash (p, w, \beta\gamma)$$

(moving to the next id)   a consumed ; β pushed over γ

$$ID_0 \vdash ID_1 \vdash \dots \vdash ID_n$$

Initial id $(q_0, w, z_0) = ID_0$

* To express the status of a PDA, we use ID

$$ID_0 \vdash^* ID_n$$

(This notation is used in language acceptance by a PDA).

The set of strings accepted by M by empty stake

$$N(M) = \{ w \mid w \in \Sigma^*, (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon) \}$$

$$L(M) = \{ w \mid w \in \Sigma^*, (q_0, w, z_0) \vdash^* (q_f, \varepsilon, \gamma) \}$$

$q_f \in F \quad \gamma \in \Gamma^*$
(Acceptance by final state)

Q. ww^R Eg: 011110

Non deterministic PDA

$\delta(q_1, 0, R) = \{ (q_1, BR) \}$

$\delta(q_1, 1, R) = \{ (q_1, GR) \}$

$\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$

$\delta(q_1, 1, B) = \{(q_1, GB)\}$

$\delta(q_1, 0, G) = \{(q_1, BG)\}$

$\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$

$\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$

$\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$

Q. $(q_1, 110011, R)$ ↓ This is ID.

$(q_1, 10011, GR)$ → ↓ $(q_1, 0011, GGR)$ and → $(q_2, 0011, R)$

$(q_2, 0011, R)$ ↓ $(q_2, 0011, -)$ (Halts)

$(q_1, 0011, GGR)$ ↓ $(q_1, 011, BGGR)$

$(q_1, 011, BGGR)$ ↙ $(q_1, 11, BBGGR)$ ↘ $(q_2, 11, GGR)$

$(q_1, 11, BBGGR)$ ↓ $(q_1, 1, GBBGGR)$

$(q_1, 1, GBBGGR)$ ↙ $(q_1, \varepsilon, GGBBGGR)$ ↓ $(q_2, \varepsilon, BBGGR)$

$(q_2, 11, GGR)$ ↓ $(q_2, 1, GR)$ ↓ (q_2, ε, R) ↓ $(q_2, \varepsilon, \varepsilon)$

It gets accepted.

Q1 1. $L = \{ wcw^R \mid w \in (a,b)^* \}$

2. $L = \{ ww^R \mid w \in (a,b)^* \}$

3. $L = \{ a^n b^n : n > 0 \}$

4. $L = \{ a^n b^{2n} : n > 0 \}$

5. $L = \{ a^n b^{n+1} \mid n = 1, 2, 3 \ldots \}$

0110

3/10/17

TUTORIAL

Design PDA's

1) $L = \{0^n 1^m 0^n \mid m, n \geq 1\}$

$$\delta(q_0, 0, R) = \{(q_0, 0R)\}$$
$$\delta(q_0, 0, 0) = \{(q_0, 00)\}$$
$$\delta(q_0, 1, 0) = \{(q_1, 0)\}$$
$$\delta(q_1, 1, 0) = \{(q_1, 0)\}$$
$$\delta(q_1, 0, 0) = \{(q_2, \varepsilon)\} \qquad \delta(q_2, 0, 0) = \{(q_2, \varepsilon$$
$$\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\} \quad \text{(Empty store)}$$
$$\text{OR} \quad \{(q_f, \varepsilon)\} \quad \text{(By final}$$

* check eg: 001110110

2) $L = \{a^n b^{n+2} \mid n = 1, 2, 3 \ldots\}$ $\qquad$ a^n b^n b^2

$$\delta(q_0, a, R) = \{(q_0, aR)\}$$
$$\delta(q_0, a, a) = \{(q_0, aa)\}$$
$$\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, b, a) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, b, R) = \{(q_2, R)\}$$
$$\delta(q_2, b, R) = \{(q_3, \varepsilon)\}$$

3) $L = \{a^n b^m c^{m+n} \mid n, m \geq 1\}$

$$\delta(q_0, a, R) = \{(q_0, aR)\}$$
$$\delta(q_0, a, a) = \{(q_0, aa)\}$$
$$\delta(q_0, b, a) = \{(q_0, ba)\}$$
$$\delta(q_0, b, b) = \{(q_0, bb)\}$$
$$\delta(q_0, c, b) = \{(q_1, \varepsilon)\}$$

$\delta(q_1, c, b) = \{(q_1, \varepsilon)\}$
$\delta(q_1, c, a) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, R) = \{(q_1, \varepsilon)\}$

4) $L = \{a^3 b^n c^n \mid n \geq 0\}$

$\delta(q_0, a, R) = \{(q_1, R)\}$
$\delta(q_1, a, R) = \{(q_2, R)\}$
$\delta(q_2, a, R) = \{(q_3, R)\}$ $\quad \delta(q_3, \varepsilon, R) = \{(q_4, \varepsilon)\}$
$\delta(q_3, b, R) = \{(q_3, bR)\}$
$\delta(q_3, b, b) = \{(q_3, bb)\}$
$\delta(q_3, c, b) = \{(q_4, \varepsilon)\}$
$\delta(q_4, c, b) = \{(q_4, \varepsilon)\}$
$\delta(q_4, \varepsilon, R) = \{(q_4, \varepsilon)\}$

5) $L = \{a^n b^{3n} \mid n \geq 0\}$ $\quad F = \{q_f\}$
$\delta(q_0, \varepsilon, R) = \{(q_f, \varepsilon)\}$
$\delta(q_0, a, R) = \{(q_0, aR)\}$
$\delta(q_0, a, a) = \{(q_0, aa)\}$
$\delta(q_0, b, a) = \{(q_1, a)\}$
$\delta(q_1, b, a) = \{(q_2, a)\}$
$\delta(q_2, b, a) = \{(q_3, \varepsilon)\}$
$\delta(q_3, b, a) = \{(q_1, a)\}$
$\delta(q_3, \varepsilon, R) = \{(q_f, \varepsilon)\}$

6) $L = \{0^m 1^m 2^n \mid m, n \geq 1\}$

$\delta(q_0, 0, R) = \{(q_0, 0R)\}$
$\delta(q_0, 0, 0) = \{(q_0, 00)\}$
$\delta(q_0, 1, 0) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, 0) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 2, R) = \{(q_1, R)\}$

Camlin Page
Date / /

$\delta(q_1, \varepsilon, R) = \{(q_1, \varepsilon)\}$

7) $L = \{a^m b^n \mid n < m\}$

$\delta(q_0, a, R) = \{(q_0, aR), (q_f, \varepsilon)\}$
$\delta(q_0, a, a) = \{(q_0, aa)\}$
$\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$
$\delta(q_1, b, a) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, a) = \{(q_f, a)\}$

* When there is order, change states to ensure the order.

8) $L = \{w : n_a(w) + n_b(w) = n_c(w)\}$

$\delta(q_0, a, R) = \{(q_0, aR)\}$
$\delta(q_0, b, R) = \{(q_0, bR)\}$
$\delta(q_0, c, R) = \{(q_0, cR)\}$

$\delta(q_0, a, a) = \{(q_0, aa)\}$
$\delta(q_0, b, a) = \{(q_0, ba)\}$
$\delta(q_0, a, b) = \{(q_0, ab)\}$
$\delta(q_0, b, b) = \{(q_0, bb)\}$
$\delta(q_0, c, a) = \{(q_0, \varepsilon)\}$
$\delta(q_0, c, b) = \{(q_0, \varepsilon)\}$
$\delta(q_0, c, c) = \{(q_0, cc)\}$

$\delta(q_0, a, c) = \{(q_0, \varepsilon)\}$
$\delta(q_0, b, c) = \{(q_0, \varepsilon)\}$
$\delta(q_0, \varepsilon, R) = \{(q_0, \varepsilon)\}$

2 0011

Camlin Page
Date 11/0/11

06/10/17

Assignment Test

Push Down Automata

Design a PDA for

1) $L = \{x \in \{a, b, c\}^* \mid |x_a| + |x_b| = |x_c|\}$

2) $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$

3) $L = \{x0y1 \mid x, y \in \{0, 1\}^*, |x| = |y|\}$

Answer:

2)
$\delta(q_0, a, R) = \{(q_0, aR)\}$
$\delta(q_0, a, a) = \{(q_0, aa)\}$
$\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$
$\delta(q_1, b, a) = \{(q_1, \varepsilon)\}$
$\delta(q_1, c, R) = \{(q_2, cR)\}$
$\delta(q_2, c, c) = \{(q_2, cc)\}$
$\delta(q_2, d, c) = \{(q_3, \varepsilon)\}$
$\delta(q_3, d, c) = \{(q_3, \varepsilon)\}$
$\delta(q_3, \varepsilon, R) = \{(q_f, \varepsilon)\}$

$K = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b, c, d\}$
$\Gamma = \{a, c, R\}$
$z_0 = \{R\}$
$F = \{q_f\}$

Camlin Page
Date / /

3) $\delta(q_0, 0, R) = \{(q_0, xR)\}$
$\delta(q_0, 1, R) = \{(q_0, xR)\}$
$\delta(q_0, 0, x) = \{(q_0, xx), (q_1, x)\}$
$\delta(q_0, 1, x) = \{(q_0, xx)\}$
$\delta(q_1, 0, x) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, x) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, R) = \{(q_2, \varepsilon)\}$

07/10/17 $L = \{a^n b^n c^n \mid n \geq 1\}$ cannot be designed using a PDA. This is a CSL.
PDA accepts only CFG.

Grammars

Type 0: Unrestricted Grammar
$\alpha \to \beta$
Type 1: Context sensitive Grammar
$\alpha \to \beta$ β longer than

$\alpha A \beta \to \alpha \gamma \beta$
Type 2: Context Free grammar
$A \to \alpha$
Type 3: Regular Grammar
$A \to aB$
$A \to a$

* $L(G) = \{w \in \Sigma^* \mid S \xrightarrow[G]{*} w\}$

Scanned by CamScanner

Identify the Language :

Q1. $N = \{S\}$ $T = \{a, b\}$

$$P = \begin{cases} 1. & S \rightarrow aSb \\ 2. & S \rightarrow ab \end{cases}$$

CFG
* Linear Grammar
* Only 1 non-terminal on the right side

$S \xRightarrow{2} ab$ $ab \in L(G)$

$S \xRightarrow{1} aSb$

$\Rightarrow aabb$ $a^2b^2 \in L(G)$

$S \xRightarrow{1} aSb$

$\xRightarrow{1} aaSbb$

$\xRightarrow{2} aaabbb$ $a^3b^3 \in L(G)$

$L(G) = \{a^n b^n \mid n \geq 1\}$

If $P = \{1.\ S \rightarrow aSb \quad 2.\ S \rightarrow \varepsilon\}$ then $L = \{a^n b^n \mid n \geq 0\}$

Q2. $N = \{S\}$ $T = \{a, b, c\}$

$$P = \begin{cases} 1. & S \rightarrow aSa \\ 2. & S \rightarrow bSb \\ 3. & S \rightarrow c \end{cases}$$

CFG
Linear Grammar

$S \xRightarrow{3} c$

$S \xRightarrow{1} aSa$

$\xRightarrow{3} aca$

$S \xRightarrow{2} bSb$

$\xRightarrow{3} bcb$

$S \xRightarrow{2} bSb$

$\xRightarrow{1} baSab$

$\xRightarrow{3} bacab$

$L(G) = \{wcw^R \mid wcw^R \in \Sigma^*\}$

Camlin Page
Date / /

Q3. $N = \{S\}$ $T = \{a, b, c\}$

$P = \begin{cases} 1.\ S \to aSb \\ 2.\ S \to ab \\ 3.\ S \to SS \end{cases}$ CFG

$S \xRightarrow{2} ab$

$S \xRightarrow{1} aSb$
$\xRightarrow{2} aabb$

$S \xRightarrow{1} aSb$
$\xRightarrow{3} aSSb$
$\xRightarrow{2} aSabb \xRightarrow{2} aababb$

$S \xRightarrow{3} SS$
$\xRightarrow{1} aSbS$
$\xRightarrow{1} aaSbbS$
$\xRightarrow{2} aaabbbS$
$\xRightarrow{2} aaabbbab$

L(G) consists of well balanced string of a and b

Q4. $N = \{S\}$ $T = \{(,)\}$

$P = \begin{cases} 1.\ S \to (S) \\ 2.\ S \to () \\ 3.\ S \to SS \end{cases}$

$S \xRightarrow{3} SS$
$\xRightarrow{1} (S)S$
$\xRightarrow{2} (())S$
$\xRightarrow{2} (())()$

This Language of well formed string of paranthesis is called Dyck set

Q5. $N = \{S, S_1\}$ $T = \{a, b\}$

Start symbol : S_1

$$P = \begin{cases} 1.\ S \rightarrow aSb \\ 2.\ S \rightarrow ab \\ 3.\ S_1 \rightarrow SS_1 \\ 4.\ S_1 \rightarrow \varepsilon \end{cases}$$

$$S_1 \xRightarrow{4} \varepsilon$$

$$S_1 \xRightarrow{3} SS_1$$
$$\xRightarrow{2} abS_1$$
$$\xRightarrow{4} ab$$

$$S_1 \xRightarrow{3} SS_1$$
$$\xRightarrow{1} aSbS_1$$
$$\xRightarrow{4} aabbS_1$$
$$\xRightarrow{3} aabbSS_1$$
$$\xRightarrow{1} aabbaSbS_1$$
$$\xRightarrow{4} aabbaSb$$
$$\xRightarrow{2} aabbaabb$$

$a^{n_1}b^{n_1}a^{n_2}b^{n_2}$

$L(G) = \{a^n b^n \mid n \geq 0\}$

$L(G_{new}) = L^*$

Generate Grammar Rules from the Language

1. $L = \{a^n b^n \mid n \geq 1\}$

Minimum string ab

$S \rightarrow ab$

Next $a\boxed{ab}b$

aSb

$S \rightarrow aSb$

Then $aa\boxed{ab}bb$

$a\boxed{aSb}b \Rightarrow aSb$

Again $aaa\boxed{ab}bbb$
$aa\boxed{aS}bbb$
$a\boxed{aSb}b$
aSb :

$\therefore P = \begin{cases} 1.\ S \to ab \\ 2.\ S \to aSb \end{cases}$

Q2. $L(G) = \{a^n b^m c^n \mid n, m \geq 1\}$

~~Minimum~~ abc

$a^n \boxed{b^m} c^n$

$a \boxed{\ } c$

1) $S \to aAc$

$aa \boxed{\ } cc$

$aaAcc$

aSc

2) $S \to aSc$

$aaaAccc$

$aaScc$

aSc

⋮

3) $A \to b$

4) $A \to bA$

a^n RG
$a^n b^n$ CFG
$a^n b^n c^n$ CSG

RG – Right linear

a^n
$S \to a$
$S \to aS$

Q3. $a^n b^n c^m$.

$\boxed{ab}\ \boxed{c}$ AB

$S \to AB$

$A \to aAb$

$A \to ab$

$B \to cB$

$B \to c$

09/10/17

Q. Generate the grammar for the language of all well formed strings of the alphabet {(,), [,]}

- Both nesting and repetition allowed.

Nesting $S \to (S)$ Repetition $S \to SS$

$S \to ()$ () [ ]

$S \to [\,]$ (S) [S]

$S \to (S)$ ([ ]) [( )]

$S \to [S]$

$S \to SS$

Q 1. $S \to aSBc$

2. $S \to abc$

3. $cB \to Bc$ (Purpose: Brings b's together)

4. $bB \to bb$

This is of type 1. (CSG)

$S \xRightarrow{2} abc$

$S \xRightarrow{1} aSBc$

$\xRightarrow{2} aabcBc$

$\xRightarrow{3} aabBcc$

$\xRightarrow{4} aabbcc$

$S \xRightarrow{1} aSBc$

$\xRightarrow{1} aaSBcBc$

$\xRightarrow{2} aaabcBcBc$

$\xRightarrow{3} aaabBccBc$

$\xRightarrow{4} aaabbccBc$

$\xRightarrow{3} aaabbcBcc$

$\xRightarrow{3} aaabbBccc$

$\xRightarrow{4} aaabbbccc$

All RG is CFG.

Note: CFG without $A \to \varepsilon$ is contained in CSG.

$A \to \varepsilon$ violates the rule length increasing grammar.

$L = \{a^n b^n c^n \mid n \geq 1\}$

Note: RG a^n
CFG $a^n b^n$
CSG $a^n b^n c^n$

Simplification of Context Free Grammar

I. Removal of useless productions

⇒ Generate the grammar for Fortran identifier
upto six symbols are allowed
First one should be a letter
The rest will be a digit or letter.

Letter = {26 characters of English}

$S \to$ Letter S_1
$S_1 \to \varepsilon$
$S_1 \to$ (digit / Letter) S_2 (36 rules (26+10))
$S_2 \to \varepsilon$
$S_2 \to$ (digit / letter) S_3
$S_3 \to \varepsilon$
$S_3 \to$ (digit / letter) S_4
$S_4 \to \varepsilon$
$S_4 \to$ (digit / letter) S_5
$S_5 \to \varepsilon$
$S_5 \to$ (digit / letter) S_6
$S_6 \to \varepsilon$

II. Find out the unit productions

III. ε-productions to be removed

10/10/17

1) Removing useless productions
2) Removing ϵ - productions
3) Removing unit productions

I. Removing Useless Productions

* Let $G = (N, T, P, S)$ be a context free grammar. A variable X in L is said to be useful iff there is atleast a string $\alpha \in L(G)$ such that $S \overset{*}{\Rightarrow} \alpha_1 X \alpha_2 \overset{*}{\Rightarrow} \alpha$ where $\alpha_1, \alpha_2 \in (N \cup T)^*$. That is, X is useful because it appears atleast in one derivation from S to a word α in $L(G)$. Otherwise, X is useless.

Two Lemmas

1) Whether X is generating w/α.
2) Whether X is reachable from S.

Necessary conditions:

1) It should be possible to derive a terminal string from X.

2) It should be possible to derive a sentential form X from S

Q.
$S \to AB$
$S \to AC$
$B \to b$
$S \to BC$
$C \to cEd$

Camlin Page
Date / /

$E \rightarrow cEd$
$E \rightarrow cd$

$GEN = \{b, c, d\}$ (Set of terminals)

Look at the right side; pick the non-terminals that give ~~non~~ terminals & add it to GEN.

$B \rightarrow b$
$GEN = \{c, d, b, B\}$
$E \rightarrow cd$
$GEN = \{c, b, d, B, E\}$

2nd iteration
$C \rightarrow cEd$
$GEN = \{c, b, d, B, E, C\}$

Again
$S \rightarrow BC$
$GEN = \{c, b, d, B, E, C, S\}$

∴ The production rules P' for G' are
$S \rightarrow BC$
$B \rightarrow b$
$C \rightarrow cEd$
$E \rightarrow cEd$
$E \rightarrow cd$

(Till here, we use lemma 1).

Lemma 2

$Reach = \{S\}$
Scan for productions that starts with S.

Reach = {S, B, C}

Reach = {S, B, C, b, c, E, d}

Here, all the production rules are accepted as Reach contains all the non-terminals.

Q. $S \rightarrow AC$ Remove useless symbols

$S \rightarrow BA$

$C \rightarrow CB$

$C \rightarrow AC$

$A \rightarrow a$

$B \rightarrow aC/b$

lemma 1:

GEN = {a, b}

1st iteration GEN = {a, b, A, B}

2nd iteration GEN = {a, b, A, B, S}

∴ Production rules are

$S \rightarrow BA$

$A \rightarrow a$

$B \rightarrow b$

lemma 2:

Reach = {S}

Reach = {S, A, B}

∴ All three are selected.

Q. $S \to aB \mid bX$
$A \to BAd \mid bSX \mid a$
$B \to aSB \mid bBX$
$X \to SBD \mid aBx \mid ad$

lemma 1:

$GEN = \{a, b, d, x\}$

1st : $GEN = \{a, b, d, x, A, X\}$

2nd : $GEN = \{a, b, d, x, A, X, S, \cancel{B}\}$

~~Lemma 2~~:

~~Reach = {S}~~

~~1st : Reach = {S, A, X}~~

Discard all the rules that involves B.

$S \to bX$
$A \to bSX$
$A \to a$
$X \to ad$

lemma 2:

$Reach = \{S\}$

1st : $Reach = \{S, X\}$

2nd : $Reach = \{S, X\}$

$\therefore$ We have

$S \to bX$
$X \to ad$

Q. $S \to AB$
$S \to a$
$A \to a$

Applying lemma 2 followed by 1

lemma 2:
Reach = {S}
1st : Reach = {S, A, B, a}
$S \to AB$, $S \to a$, $A \to a$

lemma 1:
GEN = {a}
1st : GEN = {a, S, A}
2nd : GEN = {a, S, A}

$S \to a$ $A \to a$

Applying lemma 1 followed by 2.

lemma 1:
GEN = {a}
1st : GEN = {a, S, A}
$S \to a$ $A \to a$

lemma 2:
Reach = {S}
1st : Reach = {S, a}

∴ We get $S \to a$

II Remove the $\mathcal{E}$ production

Step 1: Identify the nullable non-terminal. A terminal is said to be nullable if $A \overset{*}{\Rightarrow} \mathcal{E}$

Step 2: If $A \rightarrow B_1 B_2 B_3 \dots B_M$ is a rule in P, $A \rightarrow C_1 C_2 \dots C_M$ will be in P'.

where $\begin{cases} C_i = B_i \text{ if } B_i \text{ is not nullable} \\ C_i = \mathcal{E} \text{ or } B_i \text{ if } B_i \text{ is nullable} \\ \text{Not all } C_i \text{ can be } \mathcal{E} \end{cases}$

Eg: $S \rightarrow SaSbS$
$S \rightarrow \mathcal{E}$

Here, S is a nullable non terminal.

In P

$A \rightarrow B_1 B_2 B_3 \dots A_m$

$\Longrightarrow S \rightarrow Sa\,S\,b\,S$

In P'

$S \rightarrow SaSbS$
$S \rightarrow aSbS$
$S \rightarrow SaSb$
$S \rightarrow SabS$
$S \rightarrow Sab$
$S \rightarrow abS$
$S \rightarrow aSb$
$S \rightarrow ab$

Camlin Page
Date / /

Q. $S \to AB0C$
$A \to AB$
$B \to 1/\varepsilon$
$C \to D/\varepsilon$
$D \to \varepsilon$

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

Nullable non terminals B, C, D

$S \to AB0C \,/\, A0C \,/\, AB0 \,/\, A0$

$A \to AB \,/\, A$

$B \to 1$
$C \to D$

Q. $S \to aA$
$A \to b/\varepsilon$

Nullable non terminal is A

$S \to aA/a$
$A \to b$

Q. $S \to aX/bX$
$X \to a/b/\varepsilon$

Nullable non-terminal : X

$S \to aX/a/bX/b$
$X \to a/b$

Q. $S \to ABaC$
$A \to BC$
$B \to b / \varepsilon$
$C \to D / \varepsilon$
$D \to d$

Nullable Non-terminals : B, C, A

$S \to ABaC / AaC / ABa / Aa / BaC / aC / Ba / a$
$A \to BC / B / C$

$B \to b$
$C \to D$
$D \to d$

III Removal of Unit Production

Eg : $NT \to NT$ $(A \to B)$

$S \stackrel{*}{\Rightarrow} \alpha A \gamma \Rightarrow \alpha B \gamma \Rightarrow \alpha \beta_i \gamma$
we can remove $\alpha B \gamma$
and set $A \to \beta_i$

* Divide P into P_1, P_2
P_2 : Set of unit productions
P_1 : The other productions

Q. $S \to aSb$
$A \to cAd$
$A \to cd$
$S \to A$

$A \to B \to \beta i$
$A \to \beta i$

Camlin Page
Date / /

P_1
$S \rightarrow aSb$
$A \rightarrow cAd$
$A \rightarrow cd$

P_2
$S \rightarrow A$

After eliminating unit production we get,

P_1
$S \rightarrow aSb$
$A \rightarrow cAd$
$A \rightarrow cd$

P_2
$S \rightarrow cAd$
$S \rightarrow cd$

Q. $S \rightarrow aX / Yb / Y$
$X \rightarrow S$
$Y \rightarrow Yb / b$

P_1
$S \rightarrow aX$
$S \rightarrow Yb$
$Y \rightarrow Yb$
$Y \rightarrow b$

P_2
$S \rightarrow Y$ (1)
$X \rightarrow S$ (2)

Here choose the order carefully as they are inter-related.

After eliminating unit productions we get,

P_1
$S \rightarrow aX$
$S \rightarrow Yb$
$Y \rightarrow Yb$
$Y \rightarrow b$

P_2
$S \rightarrow b$
$X \rightarrow aX / Yb / b$

In the other order
$X \rightarrow aX / Yb$
$S \rightarrow Yb / b$
Here $X \rightarrow b$ is missed.
Hence choose (1) followed by (2).

ie, $S \rightarrow aX / Yb / b$
$Y \rightarrow Yb / b$
$X \rightarrow aX / Yb / b$.

Scanned by CamScanner

Q. $S \to AA$
$A \to B / BB$
$B \to abB / b / bb$

P_1	P_2
$S \to AA$	$A \to B$
$A \to BB$	
$B \to abB$	
$B \to b$	
$B \to bb$	

$S \to AA$
$A \to BB / abB / b / bb$
$B \to abB / b / bb$

Q. $X \to aX / Y / b$
$Y \to bK / K / b$
$K \to a$

P_1	P_2
$X \to aX$	$X \to Y$
$X \to b$	$Y \to K$
$Y \to bK$	
$\cancel{Y \to K}$	
$Y \to b$	
$K \to a$	

$$P = \{ X \to aX / b / bK / \cancel{b} a,\quad Y \to bK / b / a,\quad K \to a \}$$

Q. $I \rightarrow a / b / Ia / Ib / I0 / I1$
$F \rightarrow I / (E)$
$T \rightarrow F / T * F$
$E \rightarrow T / E + T$

P_1

$I \rightarrow a / b / Ia / Ib / I0 / I1$
$F \rightarrow (E)$
$T \rightarrow T * F$
$E \rightarrow E + T$

P_2

$F \rightarrow I$
$T \rightarrow F$
$E \rightarrow T$

P

$F \rightarrow (E) / a / b / Ia / Ib / I0 / I1$
$T \rightarrow T * F / (E) / a / b / Ia / Ib / I0 / I1$
$E \rightarrow E + T / T * F / (E) / a / b / Ia / Ib / I0 / I1$
$I \rightarrow a / b / Ia / Ib / I0 / I1$

Q. $S \rightarrow AAA / B$
$A \rightarrow aA / B$
$B \rightarrow \epsilon$

* Note : Order to remove
1) ϵ production
2) Unit production
3) Useless symbols (lemma 1 followed by 2)

Remove ϵ production
Nullable non-terminal = B, A, S
$S \rightarrow AAA / B / AA / A$
$A \rightarrow aA / a / B$

Removing Unit Productions

P_1	P_2
$S \to AAA$	$S \to A$
$S \to AA$	$A \to B$
$A \to aA$	$S \to B$
$A \to a$	

P

$A \to aA \mid a \mid B$

$S \to AAA \mid AA \mid aA \mid a \mid B$

Removing Useless productions.

$GEN = \{a, A, S\}$

$A \to aA \mid a$

$S \to AAA \mid AA \mid aA \mid a$

$Reach = \{S\}$

$Reach = \{S, A, a\}$

Final P

$S \to AAA$

$S \to AA$

$S \to aA$

$S \to a$

$A \to aA$

$A \to a$

Camlin Page
Date / /

Q. $S \to AaB \mid aaB$
$A \to D$
$B \to bbA \mid \epsilon$
$D \to E$
$E \to F$
$F \to aS$

Remove ϵ production
Nullable non terminals = B

$S \to AaB \mid Aa \mid aaB \mid aa$
$A \to D$
$B \to bbA$
$D \to E$
$E \to F$
$F \to aS$

Remove unit production

P_1	P_2
$S \to AaB$	$A \to D$
$S \to Aa$	$D \to E$
$S \to aaB$	$E \to F$
$S \to aa$	
$B \to bbA$	
$F \to aS$	

P

$E \to aS$
$S \to AaB \mid Aa \mid aaB \mid aa$
$F \to aS$
$B \to bbA$
$D \to aS$
$A \to aS$

Remove Useless production

GEN = {a, b, S, E, F, D, A, B}

E → aS
S → aa / AaB / Aa / aaB
F → aS
B → bbA   D → aS   A → aS

Reach = {S, a, A, B, b}

∴ S → aa / AaB / Aa / aaB
B → bbA
A → aS

11/10/'17 Conversion of CFG to CNF

* There are two ways in which we can represent CFG
  1) CNF
  2) GNF

CNF

Every CFL without ε can be generated by a context free grammar with rules of the form $A \rightarrow BC$

$A \rightarrow a$

where A, B, C are non-terminals and a is a terminal.

Q. Write the grammar for $a^n b^n$, $n \geq 1$

$S \rightarrow ab$

$S \rightarrow aSb$

Step 1: For every terminal symbol, introduce a new non-terminal.

ie. $a \quad b$

$\downarrow \quad \downarrow$

$A \quad B$

$\therefore S \rightarrow AB$ CNF

$S \rightarrow ASB$ Not

$A \rightarrow a$ CNF

$B \rightarrow b$ CNF

Step 2: If $A \rightarrow B_1 B_2 B_3 \ldots B_n$ is a rule then,

$A \rightarrow B_1 D_1$

$B_1 \rightarrow B_2 D_2$

$B_2 \rightarrow B_3 D_3$

$\vdots$

$B_{n-2} \rightarrow B_{n-1} D_n$

Here, $S \rightarrow ASB$

$S \rightarrow AD_1$

$D_1 \rightarrow SB$

$\therefore$ The final rules are

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

$S \rightarrow AD_1$

$D_1 \rightarrow SB$.

Q. Write grammar for wcw^R

$S \rightarrow c$

$S \rightarrow aSa$

$S \rightarrow bSb$

Convert terminals into non-terminals

$a \downarrow A$ $\quad b \downarrow B$ $\quad c \downarrow C$

$\therefore$ $S \rightarrow c$

$S \rightarrow ASA$

$S \rightarrow BSB$

$A \rightarrow a \quad B \rightarrow b$

Step 2:

$S \rightarrow ASA$

$S \rightarrow AD_1$

$D_1 \rightarrow SA$

$S \rightarrow BSB$

$S \rightarrow BD_2$
$D_2 \rightarrow SB$

$\therefore$ Final rules are

$S \rightarrow c$
$S \rightarrow AD_1$
$D_1 \rightarrow SA$
$S \rightarrow BD_2$
$D_2 \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$

① Convert to CNF

$S \rightarrow ABb \mid a$
$A \rightarrow aaA \mid B$
$B \rightarrow bAb \mid b$

(Simplify first)

No ε productions.

Unit productions:

P_1	P_2
$S \rightarrow ABb \mid a$	$A \rightarrow B$
$A \rightarrow aaA$	
$B \rightarrow bAb \mid b$	

P

$A \rightarrow aaA \mid bAb \mid b$
$S \rightarrow ABb \mid a$
$B \rightarrow bAb \mid b$

Remove useless productions:

$GEN = \{a, b, S, B, A\}$

Camlin Page
Date / /

$A \to aaA \mid bAb \mid b$
$S \to ABb \mid a$
$B \to bAb \mid b$

Reach = $\{S, A, B, b, a\}$

$A \to aaA \mid bAb \mid b$
$S \to ABb \mid a$
$B \to bAb \mid b$

To CNF:
Convert terminals to non terminals

$a \downarrow C$ $b \downarrow D$

$A \to CCA \mid DAD \mid b$ $C \to a$
$S \to ABD \mid a$ $D \to b$
$B \to DAD \mid b$

$A \to CCA$
$A \to CD_1$
$D_1 \to CA$

$A \to DAD$
$A \to DD_2$
$D_2 \to AD$

$S \to ABD$
$S \to AD_3$
$D_3 \to BD$

Scanned by CamScanner

$B \to DAD$

$B \to CD_2$

$D_2 \to AD$

$\therefore$ Final rules are

$A \to b$

$S \to a$

$B \to b$

$C \to a$

$D \to b$

$A \to CD_1$

$D_1 \to CA$

$A \to DD_2$

$D_2 \to AD$

$S \to AD_3$

$D_3 \to BD$

$B \to CD_2$

Q. $E \to E + T / T$

$T \to (E) / a$

No ϵ productions

Unit Productions

P_1 | P_2

$E \to E + T$ | $E \to T$

$T \to (E) / a$

P $E \to E + T / (E) / a$.

$T \to (E) / a$.

Useless Productions

GEN = {a, (,), +, E, T}

Reach = $\{E, T, (,), +, a\}$

$$\begin{array}{ccc} (&) & + \\ \downarrow & \downarrow & \downarrow \\ X & Y & Z \end{array}$$

$E \rightarrow EZT \mid XEY \mid a$

$T \rightarrow XEY \mid a$

$X \rightarrow ($

$Y \rightarrow)$

$Z \rightarrow +$

$E \rightarrow ED_1$

$D_1 \rightarrow ZT$

$E \rightarrow XD_2$

$D_2 \rightarrow EY$

$T \rightarrow XD_2$

$E \rightarrow a$

$T \rightarrow a$

$Y \rightarrow)$

$X \rightarrow ($

$Z \rightarrow +$

13|10|17

AMBIGUITY

Derivation and Parse Trees

Left Most Derivation : A derivation is called leftmost derivation if we replace only the leftmost non-terminal by some production rule at each step of generating process of the language from the grammar.

Right Most Derivation : A derivation is called right most derivation if we replace only the right most non-terminal by some production rule at each step of generating process of the language from the grammar.

Q. Construct the string 0100110 from the grammar given below by using a) leftmost b) rightmost derivation

$$S \rightarrow 0S \,/\, 1AA \quad (1) \,|\, (2)$$
$$A \rightarrow 0 \,/\, 1A \,/\, 0B \quad (3) \,|\, (4) \,|\, (5)$$
$$B \rightarrow 1 \,/\, 0BB \quad (6) \,|\, (7)$$

a)
$$S \xRightarrow{1} 0S$$
$$S \xRightarrow{2} 01AA$$
$$S \xRightarrow{5} 010BA$$
$$S \xRightarrow{7} 0100BBA$$
$$S \xRightarrow{6} 01001BA$$
$$S \xRightarrow{6} 010011A$$
$$S \xRightarrow{3} 0100110$$

b) $S \xRightarrow{1} 0S$

$S \xRightarrow{2} 01AA$

$S \xRightarrow{3} 01A0$ 01A1A

$S \xRightarrow{5} 010B0$ 0B

$S \xRightarrow{7} 0100BB0$

$S \xRightarrow{6} 0100B10$

$S \xRightarrow{6} 0100110$

Q. $S \rightarrow aAB$ 1

$A \rightarrow bBb$ 2

$B \rightarrow A / \varepsilon$ 3 / 4

abbbb

LMD:

$S \xRightarrow{1} aAB$

$S \xRightarrow{2} abBbB$

$S \xRightarrow{3} abAbB$

$S \xRightarrow{2} abbBbbB$

$S \xRightarrow{4} abbbbB$

$S \xRightarrow{4} abbbb$

RMD:

$S \xRightarrow{1} aAB$

$S \xRightarrow{3} aAA$

$S \xRightarrow{2} aAbBb$

$S \xRightarrow{4} aAbb$

$S \xRightarrow{2} abBbbb$

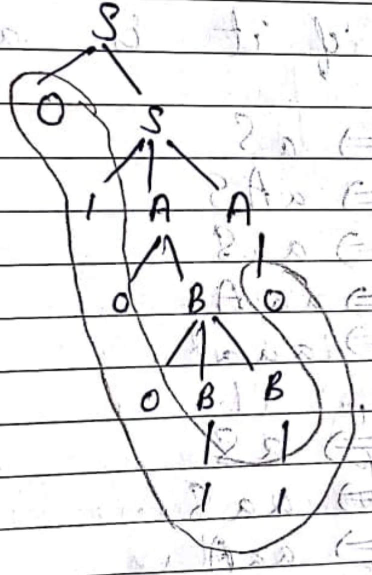
$S \xRightarrow{4} abbbb$

Q. $S \rightarrow 0S / 1AA$
$A \rightarrow 0 / 1A / 0B$
$B \rightarrow 1 / 0BB$

Find the parse tree for the derivations.

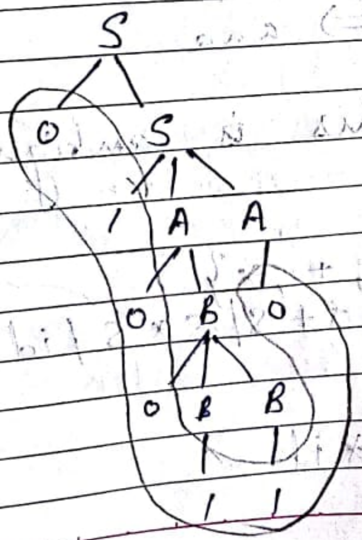
* Parsing of string is finding a derivation for that string from a given grammar

LMD:



Parse tree (Derivation tree)

RMD:



This is not an ambiguous grammar.

A grammar of a language is called ambiguous if any of the cases for generating a particular string, more than 1 leftmost derivation or more than 1 rightmost derivation or more than 1 parse tree can be generated.

Q. $S \rightarrow aS \mid AS \mid A$ 1/2/3
$A \rightarrow AS \mid a$ 4/5

Check if it is ambiguous.

$$S \overset{1}{\Longrightarrow} aS \overset{2}{\Longrightarrow} aAS \overset{5}{\Longrightarrow} aaS \overset{3}{\Longrightarrow} aaA \overset{5}{\Longrightarrow} aaa$$

$$S \overset{1}{\Longrightarrow} aS \overset{1}{\Longrightarrow} aaS \overset{3}{\Longrightarrow} aaA \overset{5}{\Longrightarrow} aaa$$

This is ambiguous grammar.

Q. $V_n : \{S\}$
$\Sigma = \{id, +, *\}$
$P : S \rightarrow S+S \mid S*S \mid id$ (1 / 2 / 3)
$S : \{S\}$

Generate id+id*id

LMD:

$$S \xRightarrow{1} S+S$$
$$S \xRightarrow{3} id+S$$
$$S \xRightarrow{2} id+S*S$$
$$S \xRightarrow{3} id+id*S$$
$$S \xRightarrow{3} id+id*id$$

$$S \xRightarrow{2} S*S$$
$$S \xRightarrow{1} S+S*S$$
$$S \xRightarrow{3} id+S*S$$
$$S \xRightarrow{3} id+id*S$$
$$S \xRightarrow{3} id+id*id$$

Conversion of CFG to Greibach Normal Form

The production rules are of the form

$$A \to aB_1 B_2 \ldots B_n$$
$$A \to a$$

Lemma 1:

Define an A production to be a production with a variable A on the left. Let $G = (N\,T\,P\,S)$ be a CFG. Let $A \to \alpha_1 B \alpha_2 B$ be a production in P and $B \to \beta_1 | \beta_2 | \cdots | \beta_r$ be the set of all B production. Let $G_1 = (N\,T\,P\,S)$ be obtained from G by deleting the production $A \to \alpha_1 B \alpha_2$ from P and adding the productions

$$A \to \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \cdots | \alpha_1 \beta_r \alpha_2$$

Then $L(G) = L(G_1)$

Lemma 2:

Let $G=(N,T,P,S)$ be a CFG.

Let $A \rightarrow A\alpha_1 / A\alpha_2 / \ldots / A\alpha_r$ be the set of A productions for which A is the leftmost symbol of the RHS. Let $A \rightarrow \beta_1 / \beta_2 \ldots / \beta_s$ be the remaining A productions. Let $G_1 = (N \cup \{Z\}, T, P_1, S)$ be the CFG formed by adding the variable Z to N and replacing all the A productions by the production

1) $A \rightarrow \beta_i$
   $A \rightarrow \beta_i Z$
   $1 \leq i \leq s$

2) $Z \rightarrow \alpha_i$
   $Z \rightarrow \alpha_i Z$
   $1 \leq i \leq r$

Then $L(G) = L(G_1)$

Step 1: For every terminal symbol introduce a new non-terminal symbol.

Q. $S \rightarrow aSb$
   $S \rightarrow ab$

Step 1: $S \rightarrow ASB$
$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

Step 2: Introduce an order among non-terminals by renaming them

* In conversion to GNF
Simplify → CNF → GNF.

$S = A_1$ $\quad$ $A_1 \to A_2 A_1 A_3$

$A = A_2$ $\quad$ $A_1 \to A_2 A_3$

$B = A_3$ $\quad$ $A_2 \to a$

$A_3 \to b$

Step 3: Use lemma 1 & lemma 2 and convert the rules in such a way that at the end of this step, the rules are in the form of GNF or in the form $A_i \to A_j \gamma$, $j > i$.

Order is $A_1 \to A_n$

✓ $A_1 \to A_2 A_1 A_3$ — Not GNF but in $A_i \to A_j \gamma$

✓ $A_1 \to A_2 A_3$ — Not GNF

✓ $A_2 \to a$ — GNF

✓ $A_3 \to b$ — GNF

Here, there is no need to apply lemma 1 and lemma 2.

Step 4: Convert the A_i rules into GNF.

Order is $A_n \to A_1$

✓ $A_3 \to b$

✓ $A_2 \to a$

$A_1 \to A_2 A_1 A_3$

✓ $A_1 \to a A_1 A_3$

$A_1 \to A_2 A_3$

✓ $A_1 \to a A_3$

$\therefore$ The rules are

$A_2 \rightarrow a \qquad A_3 \rightarrow b$

$A_1 \rightarrow aA_1A_3 \qquad A_1 \rightarrow aA_3$

Step 5: Convert the Z rules to GNF.

$Z \rightarrow A_i Z$

$Z \rightarrow A_i$

$\therefore$ The rules in GNF are

$A_3 \rightarrow b \qquad A_1 \rightarrow aA_1A_3 \qquad A_1 \rightarrow aA_3$

Q. $S \rightarrow SS$

$S \rightarrow aSb$

$S \rightarrow ab$

Step 1:

$S \rightarrow SS \qquad A \rightarrow a$

$S \rightarrow ASB \qquad B \rightarrow b$

$S \rightarrow AB$

Step 2: $S_1 = A_1$

$A = A_2$

$B = A_3$

$A_1 \rightarrow A_1A_1 \qquad A_2 \rightarrow a$

$A_1 \rightarrow A_2A_1A_3 \qquad A_3 \rightarrow b$

$A_1 \rightarrow A_2A_3$

Step 3:

X $A_1 \rightarrow A_1A_1$

✓ $A_1 \rightarrow A_2A_1A_3$

✓ $A_1 \rightarrow A_2A_3$

✓ $A_2 \rightarrow a$ ✓ $A_3 \rightarrow b$

Here A_1 is the problem creator.

$A_1 \to A_1 \underbrace{A_1}_{\alpha_1}$ (left recursive)

$A_1 \to A_2 A_1 A_3$ (β_1)

$A_1 \to A_2 A_3$ (β_2)

($A \to \alpha A \beta$ recursive rule

$A \to A\alpha$ left recursive rule

$A \to \alpha A$ right recursive rule)

$\therefore$ The new rules are (from lemma 2)

$$\left.\begin{array}{l} A_1 \to A_2 A_1 A_3 \\ A_1 \to A_2 A_3 \\ A_1 \to A_2 A_1 A_3 Z \\ A_1 \to A_2 A_3 Z \end{array}\right\} 2 \times 2 \ \beta \text{ rules}$$

$$\left.\begin{array}{l} Z \to A_1 \\ Z \to A_1 Z \end{array}\right\} 2$$

$A_2 \to a$, $A_3 \to b$

Step 4: Convert all A_i to GNF.

$A_1 \to a A_1 A_3$

$A_1 \to a A_3$

$A_1 \to a A_1 A_3 Z$

$A_1 \to a A_3 Z$

$Z \to A_1$ $\quad Z \to A_1 Z$ $\quad A_2 \to a$ $\quad A_3 \to b$

Step 5: $Z \to a A_1 A_3$

$Z \to a A_3$

$Z \to a A_1 A_3 Z$

$Z \to a A_3 Z$

$Z \to a A_1 A_3 Z$

$Z \to a A_3 Z$

$Z \to a A_1 A_3 Z Z$

$Z \to a A_3 Z Z$

$A_1 \to a A_1 A_3$

$A_1 \to a A_3$

$A_1 \to a A_1 A_3 Z$

$A_1 \to a A_3 Z$ $\qquad A_3 \to b$

∴ The final rules are

$Z \rightarrow aA_1A_3$ $\quad Z \rightarrow aA_3$

$Z \rightarrow aA_1A_3Z$ $\quad Z \rightarrow aA_3Z$.

$Z \rightarrow aA_1A_3ZZ$ $\quad Z \rightarrow aA_3Z$

$A_1 \rightarrow aA_1A_3$ $\quad A_1 \rightarrow aA_3$ $\quad A_1 \rightarrow aA_1A_3Z$

$A_1 \rightarrow aA_3Z$ $\quad A_3 \rightarrow b$.

17/10/17 Pumping Lemma

→ Helps to check if a language is non-regular or not.

Eg for non-regular: a^nb^n, ww^R

* Based on Pigeonhole Principle

n pigeons

m pigeonholes $\quad n > m$.

A pigeonhole must contain atleast 2 pigeons.

* If $|w| \geq$ # states of DFA

by pigeonhole principle, a state is repeated in the walk w.

* Finite language = regular

* $w \in L$ $\quad |w| \geq m$. (There is repetition)

Then $w = x \; y \; z$

$x = \sigma_1 \ldots \sigma_i$ $\quad y = \sigma_{i+1} \ldots \sigma_j$ $\quad z = \sigma_{j+1} \ldots \sigma$

Observations

1) $|xy| \leq m$ (In xy, no state is repeated)

2) $|y| \geq 1$ (∵ there is atleast one transition in loop)

3) String xz is accepted.

4) String $xyyz$ is accepted.

i.e; String xy^iz is accepted $i=0,1,2\ldots$

$\therefore xy^iz \in L$

The Pumping lemma:

- Given an infinite regular language L.
- There exists an integer m (critical length)
- For any string $w \in L$ with length $|w| \geq m$
- We can write $w = xyz$
- with $|xy| \leq m$ and $|y| \geq 1$
- Such that $xy^iz \in L$ $i=0,1,2\ldots$

Application:

An infinite language L is not regular.

1. Assume the opposite: L is regular
2. The pumping lemma should hold for L.
3. Use the pumping lemma to obtain a contradiction.
4. Therefore, L is not regular

Step 1:

1. Let m be the critical length for L.
2. Choose a particular string $w \in L$ which satisfies $|w| \geq m$

3. Write $w = xyz$
4. Show that $w' = xy^iz \notin L$ for some $i \neq$
5. This gives a contradiction, since from pumping lemma $w' = xy^iz \in L$.

Theorem: The language $L = \{a^n b^n : n \geq 0\}$ is not regular.

Step 1: L is regular and infinite
We can apply pumping lemma.

Let m be the critical length
Pick a string w such that: $w \in L$ and length $|w| \geq m$.
Let us pick $w = a^m b^m$.

From pumping lemma
we write $w = a^m b^m = xyz$
with lengths $|xy| \leq m$, $|y| \geq 1$

$$w = xyz = a^m b^m = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..a\,b..b}_{z}}$$

with m over the a's and m over the b's.

Thus: $y = a^k$, $1 \leq k \leq m$
(No x part)

$xyz = a^m b^m$; $y = a^k$, $1 \leq k \leq m$

From the pumping lemma: $xy^2z \in L$

$$xy^2z = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..a}_{y}\underbrace{a..a\,b..b}_{z}} \in L$$

with $m+k$ over the a's and m over the b's.

Thus: $a^{m+k} b^m \in L$

Our L is of the form $a^n b^n$
$\therefore a^{m+k} b^m \notin L$.

$\therefore$ Our assumption that L is a regular language is not true.

Conclusion: L is not a regular language.

Q10 $L = ww$

Eg: $a^2 b a^2 b$

L is regular.
Applying pumping lemma.
Let m be the critical length.
Pick a string w such that $|w| \geq m$.
Let us choose $w = a^m b a^m b = xyz$.

$|xy| \leq m \quad |y| \geq 1$

$$w = xyz = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..aba..ab}_{z}}^{m}$$

Thus $y = a^k$, $1 \leq k \leq m$

From pumping lemma: $xy^2z \in L$

$$xy^2z = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..a}_{y}\underbrace{a..ab}}^{m+k}\overbrace{\underbrace{a..ab}_{z}}^{m}$$

Thus $a^{m+k} b a^m b \in L$

Our L is of the form $a^m b a^m b$
$\therefore a^{m+k} b a^m b \notin L$.

$\therefore L$ is non-regular.

Q1. $L = \{ x \in (0,1)^* \mid x = x^R \}$ (Palindrome)

Q2. $L = \{ w \in \Sigma^* \mid n_a(w) < n_b(w) \}$

Ans 1. Assume L is regular.

Applying pumping lemma.

Let m be the critical length

Pick a string w such that $|w| \geq m$.

Let $w = 0^m 1 0^m$, $|w| > m$.

$|xy| \leq m$, $|y| \geq 1$

$$w = xyz = \underbrace{\overbrace{0..00..00..0}^{m}}_{x\ y}\ \underbrace{1\ \overbrace{0...0}^{m}}_{z}$$

$y = 0^k$, $1 \leq k \leq m$

From lemma $xy^2z \in L$

$$xy^2z = \overbrace{\underbrace{0..0}_{x}\underbrace{0..00}_{y}\underbrace{..00}_{y}..0}^{m+k}\underbrace{0..010..0}_{z}$$

Thus : $0^{m+k} 1 0^m \in L$

Our string was of the form $0^n 1 0^n$ $\therefore 0^{m+k} 1 0^m \notin L$

$\therefore L$ is non-regular.

Ans 2. Assume L is regular.

Applying pumping lemma

Let m be the critical length

Pick a string such that $|w| \geq m$.

Let $w = a^m b^{m+1}$; $|w| \geq m$

$|xy| \leq m \quad |y| \geq 1$

$$w = xyz = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..a}_{z}}^{m}\overbrace{b..b}^{m+1}$$

$y = a^k \quad 1 \leq k \leq m$

from lemma $xy^2z \in L$

$$xy^2z = \overbrace{\underbrace{a..a}_{x}\underbrace{a..a}_{y}\underbrace{a..a}_{y}\underbrace{a..a}_{z}}^{m+k}\overbrace{b..b}^{m+1}$$

Thus $a^{m+k}b^{m+1} \in L \quad 1 \leq k \leq m$.

Our language has $n_a(w) < n_b(w)$

$\therefore a^{m+k}b^{m+1} \notin L$

$\therefore$ L is non regular.

Q. $L = \{0^p \mid p \text{ is a prime number}\}$

$w = xyz$

$|xyz| = p$

$|xy^iz| = |xy^{p+1}z|$

$= |xyz| + |y|^p$

$= p + p(|y|)$

$= \underbrace{p(1+|y|)}$ $\quad |y| \geq 1$

This is not a prime number

Hence L is non regular.

Q. $L = \{a^i b^j \mid i \neq j\}$

Step 1: Assume L is regular.

Step 2: By Pumping lemma,

$z = a^i b^j = uvw$ such that $|v| \neq 0$ or $|v| \geq 1$

Step 3: There are two cases:

i) $i > j$, in this case $v = a^{i-j}$

ii) $j \geq i$, in this case $v = b^{j-i}$

Case i) $z = uv^k w$

$z = a^i b^j = a^j a^{(i-j)k} b^j$

for $k = 0$ we have $z = a^j b^j$ which is a contradiction.

Case ii) $z = uv^k w$

$z = a^i b^j = a^i b^{(j-i)k} b^i$

for $k = 0$, we have $z = a^i b^i$ which is a contradiction.

OR

Let $w = a^m b^{m+1}$ $\quad |w| \geq m$

$|xy| \leq m$ $\quad |y| \geq 1$

$$w = xyz = \overbrace{\underbrace{a \ldots a}_{x}\underbrace{a \ldots a}_{y}}^{m}\underbrace{a \ldots a\overbrace{b \ldots b}^{m+1}}_{z}$$

$y = a^k$ $\quad 1 \leq k \leq m$

$$xy^2z = \overbrace{\underbrace{a \ldots a}_{x}\underbrace{a \ldots a}_{y}\underbrace{a \ldots a}_{y}a \ldots a}^{m+k}\underbrace{\overbrace{b \ldots b}^{m+1}}_{z} \quad 1 \leq k \leq m$$

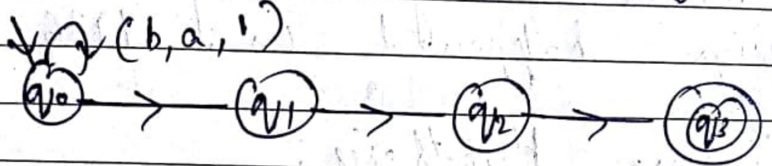
Thus $a^{m+k} b^{m+1} \in L$ $\quad 1 \leq k \leq m$

Our $L = \{a^i b^j \mid i \neq j\}$
For $k = 1$

$$a^{m+1} b^{m+1} \notin L.$$

Hence L is non regular.

Note:



$$\delta(q_0, b, a) = (q_1, 1)$$

23/10/17

Module - V

Turing Machine

A model designed by A.M. Turing. The design happened in 1936.

Church Turing Hypothesis:

"What could naturally be called an effective procedure can be realized by a turing machine."

1) Given a number 'n'. Is it perfect or not?

Can be done using algorithm. It halts (gives yes or no).

2) Given a number 'n'. Is there ~~xxx~~ a perfect number greater than 'n'?

We start with n+1. If it is not a perfect number, we again continue with n+2 and so on. Therefore, it may or may not halt.

3) Check whether a number is prime.

- Algorithm

⟹ Language accepted by turing machine is "Recursively enumerable Language".

* The input tape is an infinite tape. It has cells. Some cells are mapped as blanked (B).

B	B	B	0	1	0	1	B	B	B

* It can be an acceptor or an I/O device.
* We can modify the i/p (content) on the i/p tape.
* From state q on a, we move to another state p, write X to the i/p tape & move the head left or right.

$$\delta(q, a) = (p, X, R)$$

* Non-blank portion is finite.

Turing machine as an acceptor:

It is a 7-tuple.

$$M = (K, \Sigma, \Gamma, \delta, q_0, B, F)$$

$K(Q)$: Finite set of states.
Σ : input symbols
Γ : tape symbol
B : blank symbol on tape

$\Sigma \subseteq \Gamma$ $\quad B \in \Gamma$ $\quad \Sigma \cup B = \Gamma$

$B \notin \Sigma$

q_0 : initial state
F : set of final states

$\delta : K \times \Gamma \rightarrow K \times \Gamma \times \{L, R\}$

(we start reading from the blank cells).

Turing Machine as an I/O Machine

Q. Check whether the number of 1's in the given input is even or odd.

0 1 0 1 1 0

...	B	B	B	B	X	X	X	X	X	X	B	B	B	...

1) $\delta(q_0, 0) = (q_0, X, R)$

q_0 : Even number of 1's
q_1 : Odd number of 1's

2) $\delta(q_0, 1) = (q_1, X, R)$
3) $\delta(q_1, 0) = (q_1, X, R)$
4) $\delta(q_1, 1) = (q_0, X, R)$
$\cancel{\delta(q_0, X, R) = (q_1, X, R)}$

5) $\delta(q_1, B) = (H, O, -)$: The first blank cell gives the result.
(Odd)

6) $\delta(q_0, B) = (H, E, -)$

$K = \{q_0, q_1, H\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, X, B, O, E\}$

Acceptor (One end finite, other infinite)

Q. Given a string of paranthesis. Is it well formed or not? If yes,

4

print Y, otherwise N.

... A	(	(	)	(	)	)	(	)	A ...
A	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	A ...

A : end marker

1. $\delta(q_0, C) = (q_0, C, R)$
2. $\delta(q_0,)) = (q_1, X, L)$
3. $\delta(q_1, C) = (q_0, X, R)$
4. $\delta(q_0, X) = (q_0, X, R)$
5. $\delta(q_1, X) = (q_1, X, L)$
6. $\delta(q_0, A) = (q_2, A, L)$

   q_2 → reached end of string.

7. $\delta(q_2, X) = (q_2, X, L)$
8. $\delta(q_2, A) = (H, Y, \_)$
9. $\delta(q_2, C) = (H, N, \_)$
10. $\delta(q_1, A) = (H, N, \_)$

Q. For the string $a^n b^n$ (as an acceptor) (Here the pattern & count is imp.)

a	a	a	b	b	b		
~~X~~	~~X~~	~~X~~	~~Y~~	~~Y~~	~~Y~~	B	B

$F = \{q_4\}$

$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_2, Y, L)$
$\delta(q_2, a) = (q_2, a, L)$
$\delta(q_2, X) = (q_0, X, R)$
$\delta(q_1, Y) = (q_1, Y, R)$
$\delta(q_2, Y) = (q_2, Y, L)$
$\delta(q_0, Y) = (q_3, Y, R)$
$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_4, \_, \_)$
$\delta(q_3, b) = (H, \ , \ )$
$\delta(q_1, B) = (H, \ , \ )$

Extra Cases:

$\delta(q_3, a) = (H, \ , \ )$
$\delta(q_0, b) = (H, \ , \ )$

5

Camlin Page
Date / /

q. Concatenate the strings of 1's separated by #

{ | B | B | 1 | 1 | # | 1 | 1 | 1 | B | B }

$\delta(q_1, 1) = (q_1, 1, R)$
$\delta(q_1, \#) = (q_2, 1, R)$
$(q_2, 1) = (q_2, 1, R)$
$(q_2, B) = (q_3, B, L)$
$(q_3, 1) = (q_3, B, \ )$

Q1. $L = (a,b)^*$ $n_a + n_b$ = even number.
Q2. That computes $f(x) = x+1$
Q3. That copies a string of 0s & paste it just after the string.
Q4. Perform 1's complement operation on binary string.

25/10/17

$F = \{q_2\}$

Ans 1. $\delta(q_0, a) = (q_1, a, R)$
$\delta(q_0, b) = (q_1, b, R)$
$\delta(q_1, a) = (q_0, a, R)$
$\delta(q_1, b) = (q_0, b, R)$
$\delta(q_0, B) = (q_2, \ , \ )$
$\delta(q_1, B) = (H, \ , \ )$

2. x : No. represented as strings of 1s.
$\delta(q_0, 1) = (q_0, 1, R)$
$\delta(q_0, B) = (q_1, 1, R)$
$\delta(q_1, B) = (q_2, 1,)$

$F = \{q_2\}$

6

Camlin Page

Date / /

3. {.. |B|B|0|0|0|B|B..}

{.. |B|B|0|0|0|0|0|0|B|B..}

$\delta(q_0, 0) = (q_0, X, R)$
$\delta(q_0, B) = (q_1, B, L)$
$\delta(q_1, X) = (q_2, 0, R)$
$\delta(q_2, B) = (q_1, 0, L)$
$\delta(q_1, 0) = (q_1, 0, L)$
$\delta(q_2, 0) = (q_2, 0, R)$
$\delta(q_1, B) = (q_f, B, \ )$

4. $\delta(q_0, 0) = (q_0, 1, R)$
$\delta(q_0, 1) = (q_0, 0, R)$
$\delta(q_0, B) = (q_f, B, \ )$

Q5. $a^n b^{2n}$

$\delta(q_0, a) = (q_1, X, R)$ aabbbb
$\delta(q_1, a) = (q_1, a, R)$ X YY
$\delta(q_1, b) = (q_1, b, R)$
$\delta(q_1, B) = (q_2, B, L)$ aaaabbbb
$\delta(q_2, b) = (q_3, Y, L)$ XX YYY
$\delta(q_3, b) = (q_4, Y, L)$
$\delta(q_4, b) = (q_4, b, L)$
$\delta(q_4, a) = (q_4, a, L)$
$\delta(q_4, X) = (q_0, X, R)$
$\delta(q_1, Y) = (q_2, Y, L)$
$\delta(q_0, Y) = (q_f, \ , \ )$
$\delta(q_2, a) = (H, \ , \ )$
$\delta(q_3, X) = (H, \ , \ )$
$\delta(q_2, X) = (H, \ , \ )$

27/10/17

Tutorial - IV

Q1. $S \rightarrow aAa \mid bBb \mid \varepsilon$
$A \rightarrow C \mid a$
$B \rightarrow C \mid b$
$C \rightarrow CDE \mid \varepsilon$
$D \rightarrow A \mid B \mid ab$

Step 1: Nullable non-terminals : S, C, A, B,

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

Step 2:

P_1	P_2
$S \rightarrow aAa \mid aa \mid bBb \mid bb$	$A \rightarrow C$
$A \rightarrow a$	$B \rightarrow C$
$B \rightarrow b$	$C \rightarrow E$
$C \rightarrow CDE \mid CE \mid DE$	$D \rightarrow A$
$D \rightarrow ab$	$D \rightarrow B$

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

The unit production $A \rightarrow E$, $B \rightarrow E$, $C \rightarrow E$, $D \rightarrow E$ can't be removed as E has no production.

Step 3:

lemma 1 : GEN = {a, b, D, B, A, S}

C is useless.

$S \rightarrow aAa / aa / bBb / bb$

$A \rightarrow a /$

$B \rightarrow b$

$D \rightarrow ab / a / b$

lemma 2: Reach = {S, A, B, a, b}

$S \rightarrow aAa / aa / bBb / bb$

$A \rightarrow a$

$B \rightarrow b$

Conversion to CNF:

Step 1: Convert terminals to non-terminals

$X_a \rightarrow a$

$X_b \rightarrow b$

$S \rightarrow X_a A X_a / X_a X_a / X_b B X_b / X_b X_b$

$A \rightarrow a$

$B \rightarrow b$

Step 2: Convert all productions to CNF form

$S \rightarrow X_a A X_a$

$S \rightarrow X_a D_1 \qquad D_1 \rightarrow A X_a$

$S \rightarrow X_b B X_b$

$S \rightarrow X_b D_2 \qquad D_2 \rightarrow B X_b$

∴ Final production rules are

$S \rightarrow X_a D_1 \qquad X_a \rightarrow a$

$D_1 \rightarrow A X_a \qquad X_b \rightarrow b$

$S \rightarrow X_b D_2 \qquad A \rightarrow a$

$D_2 \rightarrow B X_2 \qquad B \rightarrow b$

$S \rightarrow X_a X_a$

$S \rightarrow X_b X_b$

Q2. Design a TM that accepts string with it equal number of a's and b's.

q_0: searches for a
q_2: searches for b

$\delta(q_0, a) = (q_1, X, L)$
$\delta(q_0, b) = (q_0, b, R)$
$\delta(q_0, X) = (q_0, X, R)$
$\delta(q_1, a) = (q_1, a, L)$
$\delta(q_1, B) = (q_2, B, R)$
$\delta(q_2, a) = (q_2, a, R)$
$\delta(q_2, X) = (q_2, X, R)$
$\delta(q_2, b) = (q_3, Y, L)$
$\delta(q_3, X) = (q_3, X, L)$
$\delta(q_3, a) = (q_3, a, L)$
$\delta(q_3, B) = (q_0, B, R)$
$\delta(q_0, Y) = (q_0, Y, R)$
$\delta(q_1, X) = (q_1, X, L)$
$\delta(q_1, Y) = (q_1, Y, L)$
$\delta(q_3, Y) = (q_3, Y, L)$
$\delta(q_2, Y) = (q_2, Y, R)$
$\delta(q_0, B) = (H, \ , \ )$

8 I/O ≈ Computing

30/10/19

Q. Design a turing machine (I/O) as a copy machine.

I/P : #ω#

O/P : #ω#ω#

Write the tuples, explanation & then transitions

$\delta(q_0, a) = (q_a, X, R)$
$\delta(q_a, a) = (q_a, a, R)$
$\delta(q_a, b) = (q_a, b, R)$
$\delta(q_a, \#) = (q_a, \#, R)$
$\delta(q_a, B) = (q_1, a, L)$
$\delta(q_1, \#) = (q_1, \#, L)$
$\delta(q_1, a) = (q_1, a, L)$
$\delta(q_1, b) = (q_1, b, L)$
$\delta(q_1, X) = (q_0, X, R)$
$\delta(q_0, b) = (q_b, Y, R)$
$\delta(q_b, a) = (q_b, a, R)$
$\delta(q_b, b) = (q_b, b, R)$
$\delta(q_b, \#) = (q_b, \#, R)$
$\delta(q_b, B) = (q_1, b, L)$
$\delta(q_1, Y) = (q_0, Y, R)$
$\delta(q_0, \#) = (q_2, \#, R)$
$\delta(q_2, a) = (q_2, a, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, B) = (q_3, \#, L)$
$\delta(q_3, a) = (q_3, a, L)$
$\delta(q_3, b) = (q_3, b, L)$
$\delta(q_3, \#) = (q_4, \#, L)$
$\delta(q_4, X) = (q_4, a, L)$
$\delta(q_4, Y) = (q_4, b, L)$
$\delta(q_4, \#) = (q_f, \#, -)$

Q. UNARY TO BINARY Converter
$(1^i / 0^i)$
(a^i / b^i)
Let us consider the i/p as a string of a's.

B B a a a a a B B
1 x
1 0 x x
1 1 x x x
1 0 0 x x x x
1 0 1 x x x x x

$\delta(q_0, a) = (q_1, x, L)$
$\delta(q_1, B) = (q_0, 1, R)$
$\delta(q_0, x) = (q_0, x, R)$
$\delta(q_1, x) = (q_1, x, L)$
$\delta(q_1, 1) = (q_1, 0, L)$
$\delta(q_0, 0) = (q_0, 0, R)$
$\delta(q_1, 0) = (q_0, 1, R)$
$\delta(q_0, B) = (H, ,)$

* Q1. Design a TM that accepts $a^i b^j c^k$, $i, j, k \geq 1$ where $i = j + k$.
Q2. Design a TM accepts $a^n b^n c^n$
Q3. I/p : #w#   o/p : #w^R#
Q4. Accepts strings that ends with b and begins with a.
Q5. $x+2$
Q6. Copy strings of 1's.

10

Ans:

1. aaaaa bb ccc BBB

aabc BB

aa cc

$\delta(q_0, a) = (q_1, X, R)$

$\delta(q_1, a) = (q_1, a, R)$

$\delta(q_1, b) = (q_2, Y, L)$

$\delta(q_2, a) = (q_2, a, L)$

$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_1, Y) = (q_3, Y, R)$

$\delta(q_3, b) = (q_2, Y, L)$

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, c) = (q_4, Z, L)$

$\delta(q_4, Y) = (q_4, Y, L)$

$\delta(q_4, a) = (q_4, a, L)$

$\delta(q_4, X) = (q_0, X, R)$

$\delta(q_3, Z) = (q_5, Z, R)$

$\delta(q_5, c) = (q_4, Z, L)$

$\delta(q_4, Z) = (q_4, Z, L)$

$\delta(q_5, Z) = (q_5, Z, R)$

$\delta(q_0, Y) = (q_6, Y, R)$

$\delta(q_6, Y) = (q_6, Y, R)$

$\delta(q_6, Z) = (q_7, Z, R)$

$\delta(q_7, Z) = (q_7, Z, R)$

$\delta(q_7, B) = (q_8, \ , \ )$

q_0 a q_1 X R

q_1 b q_2 Y L

q_2 X q_0 X R

q_1 Y q_3 Y R

q_3 b q_2 Y L

q_3 Y

q_3 c

11

30/10/17. Assignment Qs

1) Copy machine
2) $a^i b^j c^k$
3) Paranthesis.
✓ 4) Reversal.

2. aabbcc BB. $n \geq 1$

X Y Z.

$$\delta(q_0, a) = (q_1, X, R)$$
$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_2, Y, R)$$
$$\delta(q_2, b) = (q_2, b, R)$$
$$\delta(q_2, c) = (q_3, Z, L)$$
$$\delta(q_3, b) = (q_3, b, L)$$
$$\delta(q_3, Y) = (q_3, Y, L)$$
$$\delta(q_3, a) = (q_3, a, L)$$
$$\delta(q_3, X) = (q_0, X, R)$$
$$\delta(q_1, Y) = (q_1, Y, R)$$
$$\delta(q_2, Z) = (q_2, Z, R)$$
$$\delta(q_3, Z) = (q_3, Z, L)$$
$$\delta(q_0, Y) = (q_4, Y, R)$$
$$\delta(q_4, Y) = (q_4, Y, R)$$
$$\delta(q_4, Z) = (q_5, Z, R)$$
$$\delta(q_5, Z) = (q_5, Z, R)$$
$$\delta(q_5, B) = (q_6, \ , \ )$$
$$\delta(q_4, b) = (H, \ , \ )$$
$$\delta(q_5, c) = (H, \ , \ )$$
$$\delta(q_1, Z) = (H, \ , \ )$$

→①

30/10/17

Instantaneous Description for a TM

It is expressed as $\alpha_1 q \alpha_2$.
Suppose there is a non-blank portion containing $X_1 X_2 \ldots X_m$ then

$$\alpha_1\, q\, \alpha_2 = \underbrace{X_1 X_2 \ldots X_{i-1}}\; q\, \underbrace{X_i \;\ldots\; X_m}_{\alpha_2}$$

is the current ID!

1) If $\delta(q, X_i) = (p, A, R)$
then next ID is

$$\vdash \underbrace{X_1 X_2 \ldots X_{i-1} A}_{\alpha_1}\, p\, \underbrace{X_{i+1} \ldots X_m}_{\alpha_2}$$

2) If $\delta(q, X_i) = (p, A, L)$
then next ID is

$$\vdash \underbrace{X_1 X_2 \ldots X_{i-2}}_{\alpha_1}\, p\, \underbrace{X_{i-1} A X_{i+1} \ldots X_m}_{\alpha_2}$$

Acceptance is given by

$$q_0 w \vdash^* \alpha_1 q_f \alpha_2$$

Language acceptance

$$L(M) = \{ w \mid w \in \Sigma^*,\ q_0 w \vdash^* \alpha_1 q_f \alpha_2,\ q_f \in F,\ \alpha_1, \alpha_2 \in \Gamma^* \}$$

(Explain each symbol in the formal description).

31/10/17

19 Q1. Design a TM that recognizes set all strings of 0's & 1's containi[ng] atleast one 1.

Q2. TM that computes $f(n, m) = n + m$

Q3. Compute a TM that accepts $\#w\#$ and outputs $\#\#w\#$

Q4. TM as an eraser
I/p: $\#w\#$ o/p: $\#\#$

Q5. $f(w) = ww^R$

Q6. $f(m) = \begin{cases} m-2 & \text{if } m > 2 \\ 1 & \text{if } m \leq 2 \end{cases}$

Q7. $f(x, y) = y$
I/p: $\#11\#111\#$
o/p: $\#111\#$

Q8. Design a TM for the regular expression $r = aa^*$.

Q9. Language $L = (ab)^n$, $n \geq 0$

Q10. $L = \{w, w \in (a, b, c)^* \mid w$ has equal number of a's, b's & c's

Q11. $L = ab^n$, $n > 0$

Q12. Acceptor for a palindrome.

Q13. $f(x, y) = x - y$ where $x > y$

Q14. Design a TM for infinite loop.

Q15. Design a TM that performs 2's complement.

Ans 6. { | 1 | 1 | 1 | 1 | B | B }

$$\delta(q_0, B) = (q_f, 1, -)$$
$$\delta(q_0, 1) = (q_1, 1, R)$$
$$\delta(q_1, B) = (q_f, B, \ )$$
$$\delta(q_1, 1) = (q_2, 1, R)$$
$$\delta(q_2, B) = (q_3, B, L)$$
$$\delta(q_3, 1) = (q_f, B, \ )$$
$$\delta(q_2, 1) = (q_4, 1, R)$$
$$\delta(q_4, 1) = (q_4, 1, R)$$
$$\delta(q_4, B) = (q_5, B, L)$$
$$\delta(q_5, 1) = (q_6, B, L)$$
$$\delta(q_6, 1) = (q_f, B, \ )$$

7.
$$\delta(q_0, \#) = (q_1, B, R)$$
$$\delta(q_1, 1) = (q_1, B, R)$$
$$\delta(q_1, \#) = (q_2, \#, \ )$$

1.
$$\delta(q_0, 0) = (q_0, 0, R)$$
$$\delta(q_0, 1) = (q_1, 1, R)$$
$$\delta(q_1, 1) = (q_1, 1, R)$$
$$\delta(q_1, 0) = (q_1, 0, R)$$
$$\delta(q_1, B) = (q_f, B, \ )$$

Scanned by CamScanner

$\delta(q_0, B) = (H, \ , \ )$

2. $\delta(q_0, 1) = (q_0, 1, R)$ M/c reads first i/p

$\delta(q_0, \#) = (q_1, 1, R)$

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, \#) = (q_2, B, L)$

$\delta(q_2, 1) = (q_3, \#, R)$

$\delta(q_3, B) = (H, \ , \ )$

4. $\delta(q_0, \#) = (q_1, \#, R)$

$\delta(q_1, a) = (q_2, \#, R)$

$\delta(q_1, b) = (q_2, \#, R)$

$\delta(q_2, a) = (q_2, B, R)$

$\delta(q_2, b) = (q_2, B, R)$

$\delta(q_2, \#) = (q_f, B, \ )$

8. $\delta(q_0, a) = (q_1, a, R)$

$\delta(q_1, B) = (q_f, b, \ )$

$\delta(q_1, a) = (q_1, a, R)$

11. $\delta(q_0, a) = (q_1, a, R)$

$\delta(q_1, b) = (q_2, b, R)$

$\delta(q_2, b) = (q_2, b, R)$

$\delta(q_2, B) = (q_f, B, \ )$

14. $\delta(q_0, B) = (q_0, B, R)$

15. $\delta(q_0, 0) = (q_0, 0, R)$ $\#w\#$

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, \#) = (q_1, \#, L)$

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

$\delta(q_1, 1) = (q_2, 1, L)$

$\delta(q_2, 0) = (q_2, 1, L)$

$\delta(q_2, 1) = (q_2, 0, L)$

$\delta(q_2, \#) = (q_f, \#, \ )$

9. $\delta(q_0, \#) = (q_1, \#, R)$

$\delta(q_1, \#) = (q_f, \#, \ )$

$\delta(q_1, a) = (q_2, a, R)$

$\delta(q_2, b) = (q_1, b, R)$

$\delta(q_1, b) = (H, \ , \ )$

$\delta(q_2, a) = (H, \ , \ )$

13. $\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, \#) = (q_1, \#, R)$

$\delta(q_1, 1) = (q_2, X, L)$

$\delta(q_2, \#) = (q_3, \#, L)$

$\delta(q_3, 1) = (q_3, 1, L)$

$\delta(q_3, \#) = (q_4, \#, R)$

$$\delta(q_4, 1) = (q_0, X, R)$$
$$\delta(q_1, X) = (q_1, X, R)$$
$$\delta(q_3, X) = (q_4, X, R)$$
$$\delta(q_1, \#) = (q_5, B, L)$$
$$\delta(q_5, X) = (q_5, B, L)$$
$$\delta(q_5, \#) = (q_6, B, L)$$
$$\delta(q_6, 1) = (q_6, 1, L)$$
$$\delta(q_6, X) = (q_6, B, L)$$
$$\delta(q_6, \#) = (q_7, \#, R)$$
$$\delta(q_7, 1) = (q_7, 1, R)$$
$$\delta(q_7, B) = (q_f, \#, \ )$$

3. $$\delta(q_0, \#) = (q_1, \#, L)$$
$$\delta(q_1, B) = (q_f, \#, \ )$$

4. abab

$$(q_0 \# abab \#)$$
$$\vdash (\# q_1 abab \#)$$
$$\vdash (\# a q_2 bab \#)$$
$$\vdash (\# ab q_1 ab \#)$$
$$\vdash (\# aba q_2 b \#)$$
$$\vdash (\# abab q_1 \#)$$
$$\vdash (\# abab \# q_f)$$

01/11/17

Context Sensitive Grammar And Linear Bounded Automata

$a^n b^n c^n$ What? Eg? OT?

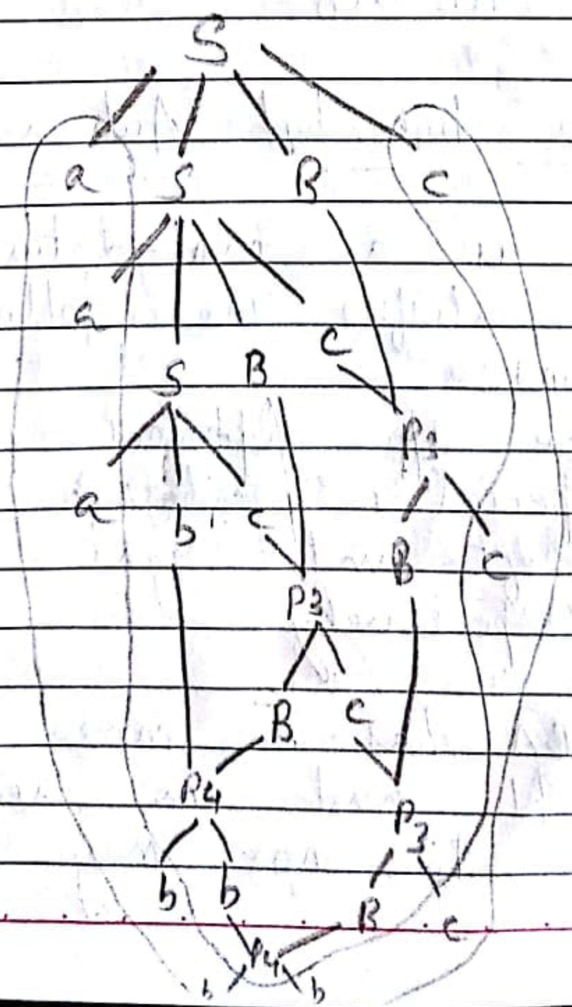
$S \rightarrow aSBc$ P_1

$S \rightarrow abc$ P_2

$cB \rightarrow Bc$ P_3

$bB \rightarrow bb$ P_4

Derivation tree for $a^3b^3c^3$.



Camlin Page

Date / /

Kuroda Normal Form (way to rep. CSG)

A length increasing grammar is said to be in KNF if any of rules has the following 4 forms

$A \rightarrow a$

$A \rightarrow B$

$A \rightarrow BC$

$AB \rightarrow CD$

Linear bounded Automata

It is a non-deterministic TM that satisfies the following two conditions:

1) The i/p alphabet includes 2 special symbols $1, $2, the left and right end marker respectively.

2) LBA has no move left from $1 and no right from $2. nor may it print

aaabbb

Camlin Page
Date / /

another symbol over \$1 and \$2.

The LBA is a TM which instead of having potentially infinite tape on which to compute is restricted to the portion of the tape containing the i/p plus the two tape squares holding the end markers.

Formal Description:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \$1, \$2, F)$$

$$\$1, \$2 \in \Gamma$$

$$L(M) = \{ w \mid w \in \Sigma^*, (q_0, \$1 w \$2) \vdash (h, \$1 y \$2) \}$$

(Halt state) $h \in F$

y : says if its accepted or not.

06/11/17

Equivalence of NFA and DFA

Theorem 1:

If ~~let~~ L ~~be~~ is accepted by a NFSA then L is accepted by a DFSA.

Proof: let L be accepted by a NFSA $M = (K, \Sigma, \delta, q_0, F)$. Then we construct a DFSA $M' = (K', \Sigma, \delta', q_0', F')$ as follows: $K' = P(K)$, power set of K. Corresponding to each subset of K, we have a state in K'. q_0' corresponds to the subset containing q_0 alone. F' consists of states corresponding to subsets having at least one state from F. We define δ' as follows:

$$\delta'([q_1, \ldots, q_k], a) = [r_1, r_2, \ldots r_s]$$

if and only if

$$\delta(\{q_1, \ldots, q_k\}, a) = \{r_1, r_2, \ldots r_s\}$$

$L(M) = L(M')$

We show that $T(M) = T(M')$.

We prove this by induction on the length of the string.

We show that:

$$\delta'(q_0', x) = [p_1, \ldots, p_r]$$

if and only if

$$\delta(q_0, x) = \{p_1, \ldots, p_r\}$$

Basis:

$|x| = 0$ i.e, $x = \varepsilon$

$$\delta'(q_0', \varepsilon) = q_0' = [q_0]$$

$$\delta(q_0, \varepsilon) = \{q_0\}$$

Induction:

Assume that the result is true for strings x of length upto m. We have to prove for string of length $m+1$. By induction hypothesis

$$\delta'(q_0', x) = [p_1, \ldots, p_r]$$

if and only if

$$\delta(q_0, x) = \{p_1, \ldots, p_r\}$$

$$\delta'(q_0', xa) = \delta'([p_1, \ldots, p_r], a)$$

$$\delta(q_0, xa) = \bigcup_{p \in P} \delta(p, a),$$

where $P = \{p_1, p_2, \ldots p_k\}$

Suppose $\bigcup_{p \in P} \delta(p, a) = \{s_1, \ldots s_m\}$

$$\delta(\{p_1, \ldots, p_k\}, a) = \{s_1, \ldots, s_m\}$$

By our construction

$$\delta'([p_1, \ldots p_k], a) = [s_1, \ldots, s_m]$$

and hence,

$$\delta'(q_0', xa) = \delta'([p_1, \ldots, p_k], a) = [s_1, \ldots, s_m]$$

In M', any state representing a subset having a state from F is in F'. So, if a string w is accepted in M, there is a sequence of states which takes M to a final state f and M' simulating M will be in a state representing

Acceptor - F 4 H

a subset containing f. Thus, $L(M) = L(M')$.

Q.5 Strings beginning with a and ending with b. Design a TM.

$$\delta(q_0, a) = (q_1, a, R)$$
$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, b, R)$$
$$\delta(q_1, B) = (q_2, B, L)$$
$$\delta(q_2, b) = (q_f, b, -)$$
$$\delta(q_0, b) = (H, \ , \ )$$
$$\delta(q_2, a) = (H, \ , \ )$$

Q. $x + 2$

$$\delta(q_0, 1) = (q_0, 1, R)$$
$$\delta(q_0, B) = (q_1, 1, R)$$
$$\delta(q_1, B) = (q_f, 1, \ )$$

Q. ww^R

$$\delta(q_0, a) = (q_0, a, R)$$
$$\delta(q_0, b) = (q_0, b, R)$$
$$\delta(q_0, B) = (q_1, B, L)$$

$$\delta(q_1, a) = (q_a, X, R)$$
$$\delta(q_a, B) = (q_1, X, L)$$
$$\delta(q_1, X) = (q_1, X, L)$$
$$\delta(q_1, b) = (q_b, Y, R)$$
$$\delta(q_b, X) = (q_b, X, R)$$
$$\delta(q_b, B) = (q_1, Y, L)$$
$$\delta(q_1, Y) = (q_1, Y, L)$$
$$\delta(q_a, X) = (q_a, X, R)$$
$$\delta(q_a, Y) = (q_a, Y, R)$$

$$\delta(q_b, Y) = (q_b, Y, R)$$

$$\delta(q_1, B) = (q_2, B, R)$$
$$\delta(q_2, X) = (q_2, a, R)$$
$$\delta(q_2, Y) = (q_2, b, R)$$
$$\delta(q_2, B) = (q_f, \ , \ )$$

7/11/17

Theorem: Let L be accepted by a NFSA with ε-moves. Then L can be accepted by a NFSA without ε-moves.

Proof: Let L be accepted by a NFSA with ε-moves $M = (K, \Sigma, \delta, q_0, F)$. Then we construct a NFSA $M' = (K, \Sigma, \delta', q_0, F')$ without ε-moves for accepting L as follows

$F' = F \cup \{q_0\}$ if ε-closure of q_0 contains a state from F.
$= F$ otherwise

$$\delta'(q_i, a) = \hat{\delta}(q_i, a)$$

We should show $T(M) = T(M')$

We wish to show by induction on the length of the string x accepted that

$$\delta'(q_0, x) = \hat{\delta}(q_0, x)$$

We start the basis with $|x| = 1$ because for $|x| = 0$, i.e, $x = \varepsilon$ this may not hold.

We may have $\delta'(q_0, \varepsilon) = \{q_0\}$ and $\hat{\delta}(q_0, \varepsilon) = \varepsilon$-closure of q_0 which may include other states.

Basis:

$|x| = 1$. Then x is a symbol of Σ say a, and $\delta'(q_0, a) = \hat{\delta}(q_0, a)$ by our definition of δ'.

Induction:

$|x| > 1$. Then $x = ya$ for some $y \in \Sigma^*$ and $a \in \Sigma$

Then $\delta'(q_0, ya) = \delta'(\delta'(q_0, y), a)$

By the inductive hypothesis

$$\delta'(q_0, y) = \hat{\delta}(q_0, y)$$

Let $\hat{\delta}(q_0, y) = P$

$$\delta'(P, a) = \bigcup_{p \in P} \delta'(p, a) = \bigcup_{p \in P} \hat{\delta}(p, a)$$

$$\bigcup_{p \in P} \hat{\delta}(p, a) = \hat{\delta}(q_0, ya)$$

$$\therefore \delta'(q_0, ya) = \hat{\delta}(q_0, ya)$$

It should be noted that $\delta'(q_0, x)$ contains a state in F' iff $\hat{\delta}(q_0, x)$ contains a state in F.

For ε this is clear from the definition. For $x = ya$, if $\hat{\delta}(q_0, x)$ contains a state from F, then surely $\delta'(q_0, x)$ contains the same state in F'. Conversely, if $\delta'(q_0, x)$ contains a state from F' other than q_0, then $\delta'(q_0, x)$ contains this state of F. The only case ~~of~~ the problem can arise is when $\delta'(q_0, x)$ contains q_0 & q_0 is not in F. This ~~closure of~~ $\cancel{\delta(\hat{\delta}(q_0, y), a)}$ can happen if ε-closure of q_0 contains some other states. In this case $\hat{\delta}(q_0, x) =$ ε-closure of $\delta(\hat{\delta}(q_0, y), a)$. Some state of q other than q_0 must have been reached from q_0 and this must be in $\hat{\delta}(q_0, x)$.

08/11/17

Properties of Regular languages

⇒ Closure Properties:

L_1, L_2 : Regular languages

Union : $L_1 \cup L_2$

Concatenation : $L_1 L_2$

Star : L_1^*

Reversal : L_1^R

Complement : $\overline{L_1}$

Intersection : $L_1 \cap L_2$

R.L are closed under the above ops

Closure property is a statement that a certain operation on languages, when applied to languages in a class, produces a result that is also in that class.

Reversal:

$$L = \{a^n b\}$$

$$L^R = \{b a^n\}$$

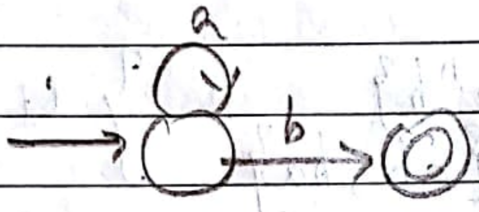
Complement : Make accepting states as non-final and vice versa.

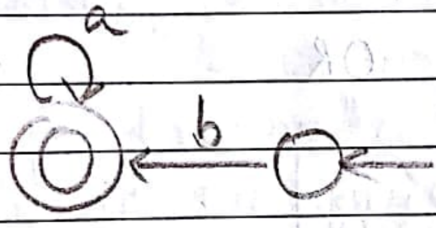
$L_1 = \{a^n b\}$

$\overline{L_1} = \{a,b\}^* - \{a^n b\}$

Reversal:

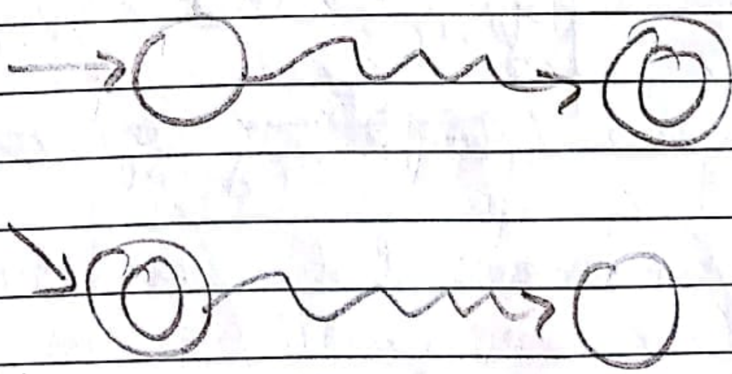
$L_1 = \{a^n b\}$





$L_1^R = \{ba^n\}$

Complement:



Intersection:

De-Morgan's law : $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

L_1 , L_2 : regular

$\Rightarrow \overline{L_1} , \overline{L_2}$: regular

$\Rightarrow \overline{L_1} \cup \overline{L_2}$: regular

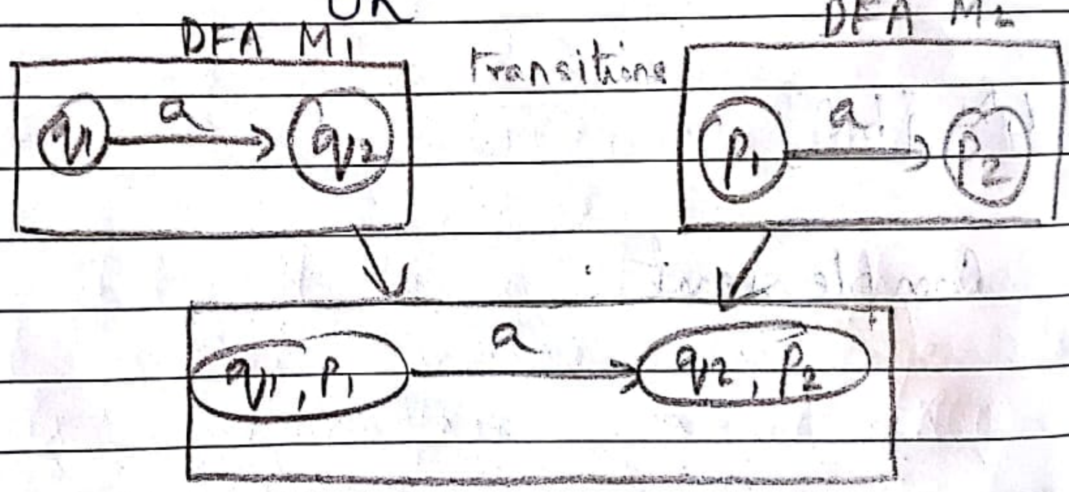
$\Rightarrow \overline{\overline{L_1} \cup \overline{L_2}}$: regular

$\Rightarrow L_1 \cap L_2$: regular

Eg:

$\left.\begin{array}{l} L_1 = \{a^n b\} \\ L_2 = \{ab, ba\} \end{array}\right\} L_1 \cap L_2 = \{ab\}$

OR



DFA M

Combine initial state (q_1, p_1)

Combine final state

q_2 p_1 p_2

(q_1, p_1) (q_1, p_2)

Difference $L_1 - L_2$
Homomorphism
Inverse homomorphism

Homomorphism:

A string homomorphism is a fn. on strings that works by substituting a particular string for each symbol. Suppose Σ and Σ' are alphabets, then function $h: \Sigma \rightarrow \Sigma'$.

To understand this

$\Sigma = \{0,1\}$ $\Sigma' = \{0,1,2\}$

$h(0) = 01$

$h(1) = 112$

Find $h(010)$

$h(010) = 0111201$

If we have a RE R for a language L, then a RE for $h(L)$ can be obtained by simply applying the homomorphism to each Σ symbol of R.

If L is a language on Σ, then its homomorphic image is defined as $h(L) = h(\omega) : \omega \in L$

Q. Find out homomorphic image of $L = \{00, 010\}$

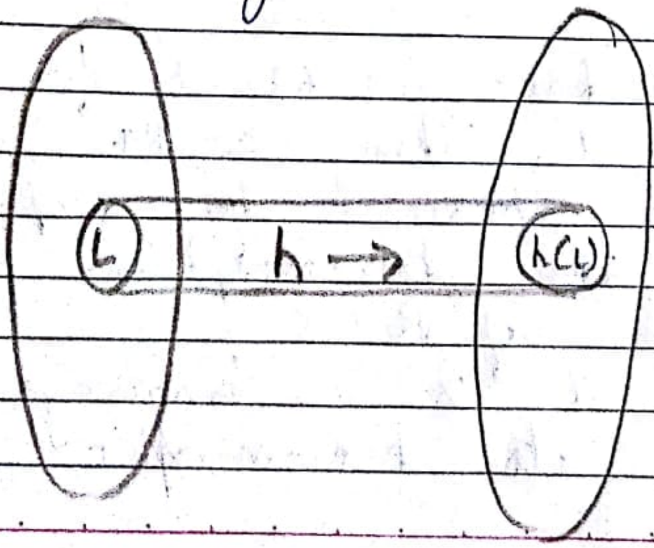
$$h(L) = \{0101, 0111201\}$$

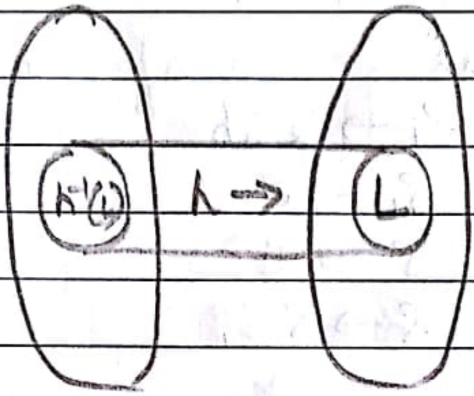
Inverse Homomorphism

$$h^{-1}(0101) = 00$$
$$h^{-1}(0111201) = 010$$

10/11/17

Let h be a homomorphism. If L is a regular language, then its homomorphic image $h(L)$ is also regular.





Closure Property of CFL

Note: $L_1 - L_2 = L_1 \cap \overline{L_2}$

L_1, L_2 : regular

$\overline{L_2}$ regular

$L_1 \cap \overline{L_2}$ regular

$\Longrightarrow L_1 - L_2$ is regular.

$L(G_1) = a^n b^n$ $L(G_2) = c^m d^m$

I) Union

We combine using a new start symbol

II) Concatenation

$S_1 \rightarrow aS_1b$

$S_1 \rightarrow \varepsilon$

$S_2 \rightarrow cS_2d$

$S_2 \rightarrow \varepsilon$

$S \rightarrow S_1S_2$

III) Kleene star

$S_1 \rightarrow aS_1b$

$S_1 \rightarrow \epsilon$

$S \rightarrow \epsilon$

$S \rightarrow SS_1$

* Intersection, set difference, complement are not supported here.

Pumping lemma for CFL

* Used to show that certain languages are not CFL.

$S \rightarrow aAa$

$A \rightarrow bBb$

$B \rightarrow cCc$

$C \rightarrow S$

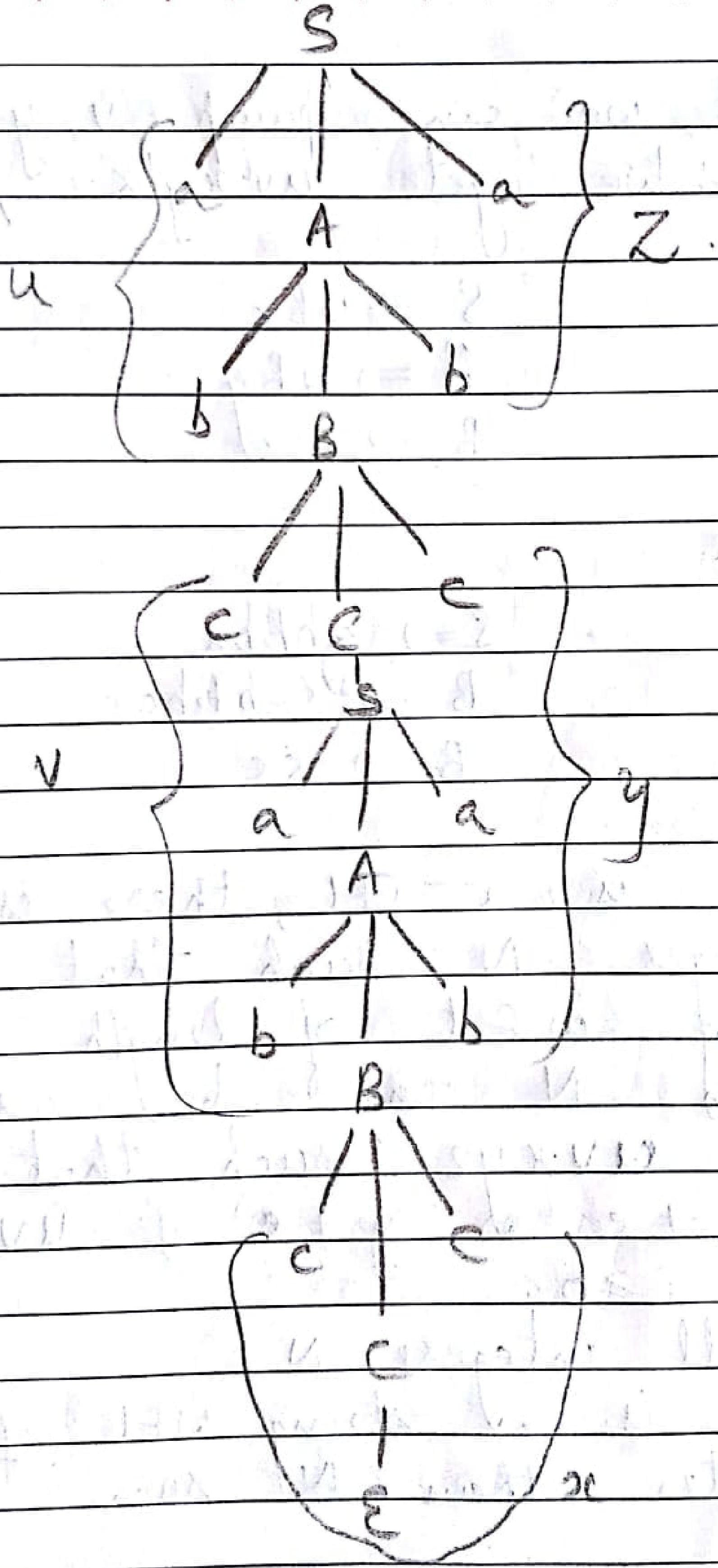
$C \rightarrow \epsilon$

$S \Rightarrow aAa$

$S \Rightarrow abBba$

$S \Rightarrow abcCcba$

$S \Rightarrow abccba$



$u = ab$
$v = cab$
$x = c\varepsilon c$
$y = bac$
$z = ba$

abcabccbacba
$L = (abc)^n (bca)^n$

Here, we can pump v, y.
$\therefore$ we get $uv^i xy^i z$, $i \geq 0$

$S \Rightarrow uBz$
$B \Rightarrow vBy$
$B \Rightarrow x$

$S \Rightarrow abBba$
$B \Rightarrow cabBbac$
$B \Rightarrow cc$

* If L is a CFL, there is an integer N such that any string $w \in L$ of length larger than N can be written as as $uvxyz$ such that $(v \neq e$ or $y \neq e)$ & $uv^i xy^i z \in L$ for $i \geq 0$.
* For all integers N there is a string $w \in L$ of length greater than N such that

13/11/17

Q. Give the CFG to generate
$L = \{ a^n b^m c^k \mid n, m, k \geq 0,\ k = |n-m| \}$

1st possibility : $n > m$
2nd : $n = m$
3rd : $n < m$

Case I : $k = n - m$

$n = m + k$

$a^{m+k} b^m c^k = a^m a^k b^m c^k$
$= a^k \boxed{a^m b^m} c^k$
$= a^k A c^k$

$S_1 \rightarrow aAc$
$S_1 \rightarrow aS_1c$
$A \rightarrow aAb$
$A \rightarrow \varepsilon$

Case II : $n = m$

$k = n - m = 0$

$a^n b^n$

$S_2 \rightarrow \varepsilon$
$S_2 \rightarrow aS_2b$

Case III :

$k = m - n$

$a^n b^{k+n} c^k = a^n b^n b^k c^k$

$= CD$

$S_3 \rightarrow CD$

$C \rightarrow aCb$

$C \rightarrow \varepsilon$

$D \rightarrow bDc$

$D \rightarrow bc$

$S \rightarrow S_1$

$S \rightarrow S_2$

$S \rightarrow S_3$

MODULE - VI

Encoding of Turing Machine

Represent TM as a binary string.

	0	1	B
q_1	$(q_2\ 0\ R)$	–	–
q_2	$(q_3\ 0\ R)$	–	–
q_3	$(q_4\ 1\ R)$	$(q_4\ 1\ L)$	$(q_4\ 1\ R)$
q_4	$(q_3\ 1\ L)$	$(q_5,\ 1\ R)$	$(q_3\ 1\ L)$
q_5	–	–	–

Consider the following TM with alphabets 0, 1, B & set of states $\{q_1 \ldots q_5\}$. q_1 is the initial & q_5 is the final state. The mapping is given above.

$$\delta(q_i, X_j) = (q_k, X_l, \overset{L}{R})$$

Encoding for input :

$x_1 = 0$

$x_2 = 1$

$x_3 = B$

$L = 1$

$R = 2$

$1 \leq i, k \leq 5$

Encoding for transition can be written as

$0^i 1 0^j 1 0^k 1 0^l 1 0^m$

$1 \leq j, l \leq 3$

$1 \leq m \leq 2$

$\delta(q_1, 0) = (q_2, 0, R)$

$0^1 1 0^1 1 0^2 1 0^1 1 0^2$

$\therefore m1 = 01010010100$

$\delta(q_2, 0) = (q_3, 0, R)$

$0^2 1 0^1 1 0^3 1 0^1 1 0^2$

$m2 = 0010100010100$

Camlin Page
Date / /

A TM is encoded as a binary string of the form

111 11 11 11 111

$\downarrow$ starting $\downarrow$ separates transitions. $\downarrow$ ending

* The above TM along with the i/p is given to another TM called Universal TM.

15/11 F Two versions of TM:
- Generalized
- Restricted

Generalized Versions of TM

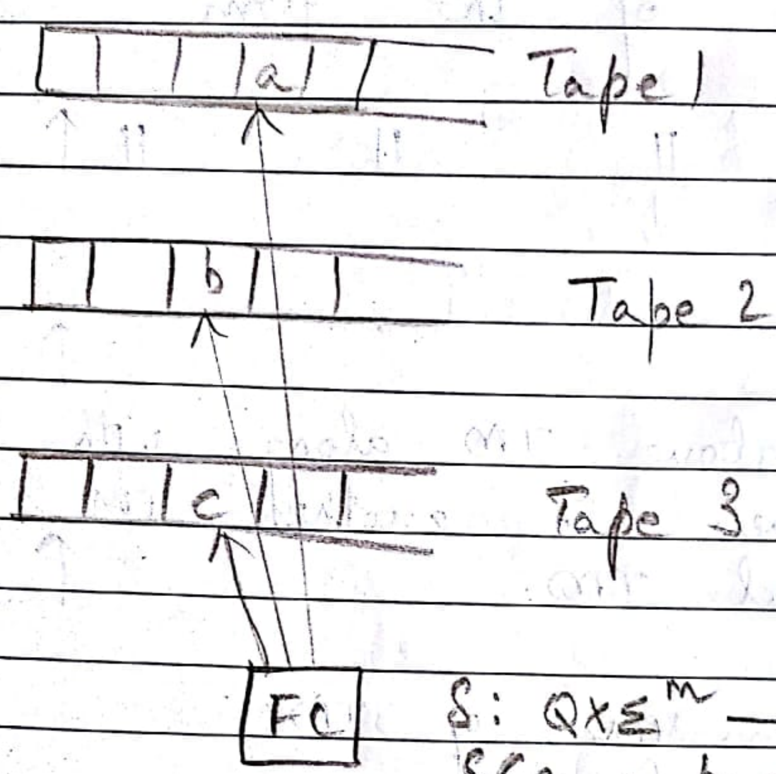
Standard model: One end finite.

1) A TM with two-way infinite tape.

Standard model
$(q_0, \#ab\#cdef\# \ldots \#)$
In this case,
$(q_0, \# \ldots \#ab\#cdef\# \ldots \#)$

Scanned by CamScanner

2) Multi-tape TM



$\delta : Q \times \Sigma^{m} \longrightarrow Q$

$\delta(q, a, b, c) = (q', [A, R], [B, L], [C, R])$

Eg: Copy Machine

Move 1	Move 2	Move 3
#w#	#w#	#w#w#
#w#	#w#	#w#

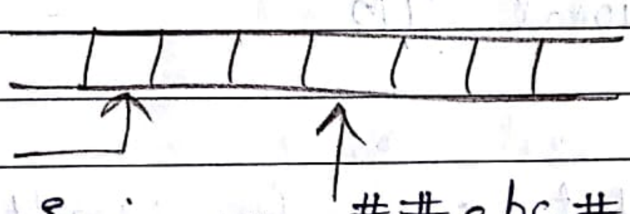
Initially,

Tape 1 : #w#
Tape 2 : empty

Move 1: Move content of Tape 1 to 2.

3) Multi-head TM

- Multiple number of heads on the same tape.



Eg: ... ##abc#def##...

H_1 (under "a"), H_2 (under "f")

δ(state, symbol under H_1, symbol under H_2)
= (new state, (s_1, m_1), (s_2, m_2))

where s_i is the symbol to be written on the cell under i^{th} head.

m_i : denotes movement of H_i
(L, R, N)
N → No movement

* If both heads access the same cell, then we set a priority for the heads & then go with the H having highest priority.

4) Non-deterministic TM

$$\delta(q, a) = \{(p_1, A, R), (p_2, B, L), (p_3, C, R)\}$$

5) k-dimensional TM

* 2D TM

- Input can be written anywhere in a 2D space, and the head can move left, right, up, down.

$\not B$	a_1	$\not B$
a_2	a_3	a_4
$\not B$	a_5	$\not B$
a_6	a_7	$\not B$

Blank = B / $\not B$

* $\delta(q, a) = (p, A, L/R/U/D)$

Camlin Page

Date / /

$\# \; b \; a_1 \; b \; \# \; a_2 a_3 a_4 \; \# \; b \; a_5 \; b \; \# \; a_6 a_7 \; b \; \#$

(All versions can be changed to single tape std. model)

From a_5, it is easy to move L or R.

To move up,
we move left till # & keep its count (incrementing) in a tape.
We keep moving left till the other # & then we move right count no. (by decrementing count) of times.

|2|

* To add a new row, column, keep adding b to the i/p as required.

Q. Construct a TM with a 2D tape.

i/p :

$$\begin{array}{cccccccc} b & b & b & b & & b & b & b & b \\ b & x & x & x & \cdots & x & x & x & b \\ b & b & b & b & & b & b & b & b \end{array}$$

o/p : $\not{b}\ \not{b}\ X\ X \quad \dots \quad X\ X\ \not{b}\ \not{b}$

$\not{b}\ X\ X\ X \quad \dots \quad X\ X\ X\ \not{b}$

$\not{b}\ \not{b}\ X\ X \quad \dots \quad X\ X\ \not{b}\ \not{b}$

$K = \{q_0, \dots q_{11}\}$

$\Gamma = \{X, Y, \not{b}\}$

δ :

YXX XXX
Y

$\delta(q_0, X) = (q_1, Y, R)$

YYX XXX

$\delta(q_1, X) = (q_2, Y, U)$

X
YYX XXX

$\delta(q_2, \not{b}) = (q_3, X, D)$

$\delta(q_3, Y) = (q_4, Y, D)$

X
YYX XXX
X

$\delta(q_4, \not{b}) = (q_5, X, U)$

$\delta(q_5, Y) = (q_1, Y, R)$

$\delta(q_1, \not{b}) = (q_6, \not{b}, L)$

$\delta(q_6, Y) = (q_7, Y, U)$

$\delta(q_7, X) = (q_8, \not{b}, D)$

$\delta(q_8, Y) = (q_9, Y, D)$

$\delta(q_9, X) = (q_{10}, \not{b}, U)$

$\delta(q_{10}, Y) = (q_{10}, X, L)$

$\delta(q_{10}, \not{b}) = (q_{11}, \not{b}, halt)$

TM as a Generator / Enumerator

- No input tape
- Machine that generates strings

$$G(M) = \{ w \mid w \text{ is generated by } M \}$$

o/p :

	#	w_1	#	w_2	#	w_3	#	...	$\not b$	$\not b$	$\not b$

7/11/17

Universal Turing Machine (write encoding)

From the prev. eg

$\delta(q_2, 0) =$

$(q_3, 0, R)$

$$0^2 1 0^1 1 0^3 1 0^1 1 0^2$$

$$00101000101 00$$

* Universal TM

i/p

encoded TM	input string
d_T	t

↓

$0 \quad (0^k)$

j

X 0011

0 1 0 1 0

Search

(i, j) stored for reference

Initially tape 2 contains initial state $q_1 \approx 0$

Tape 3 will be given the input t.

Firstly, Tape 1 is scanned to check if the encoded TM is in the correct form. If not, it halts

Note: If at any time, T reaches the final state q_2, 00 will be the content of tape 2 & U halts accepting the input. If T halts on t without accepting, U also halts without accepting.

If T when started on t gets into a loop, U also gets into a loop.

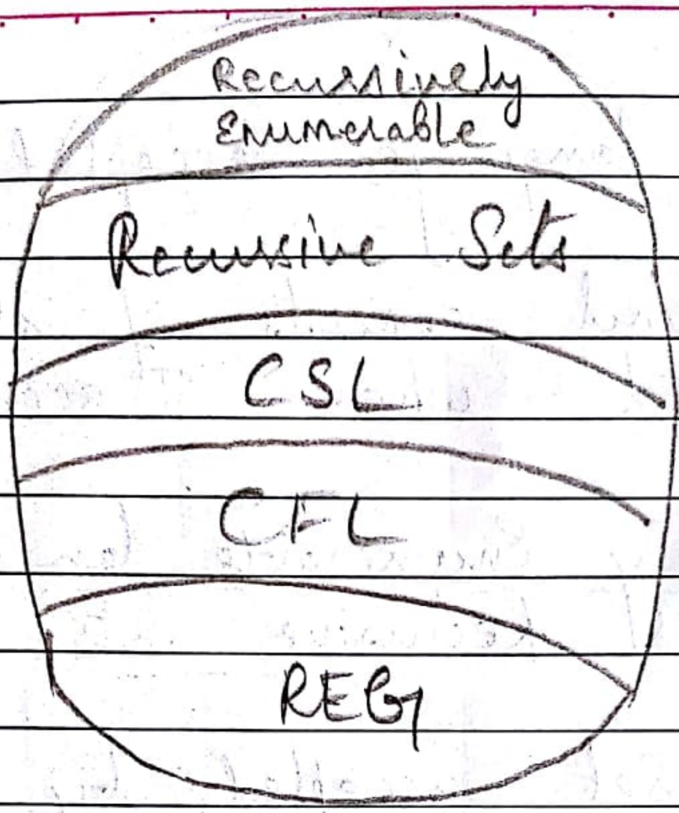
The language accepted by U is L_u and consists of strings of the form $d_T t$ where T accepts t.

Recursively Enumerable languages & Recursive Sets

REL : Set accepted by TM.
language accepted by TM.

Recursive Set:
The language accepted by a TM which halts on all input is called recursive set.
(TM can be realized as an algorithm or procedure.)

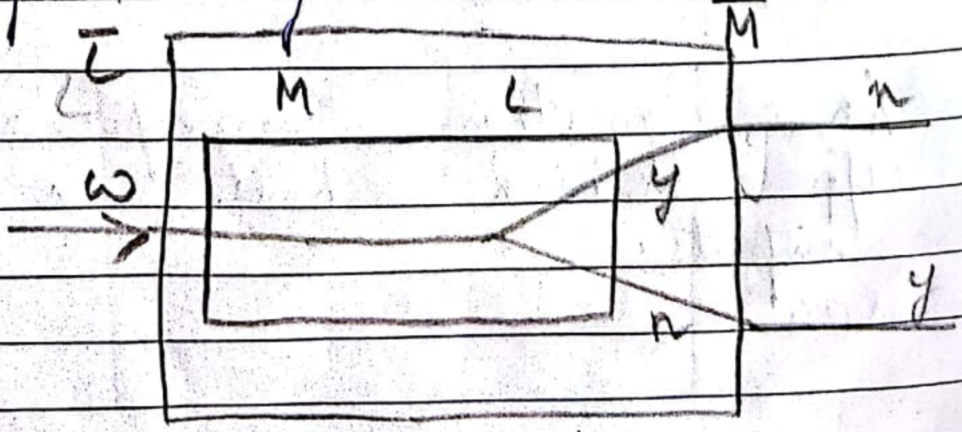
An algorithm corresponds to an i/p which halts on all inputs.



Properties

1) The complement of a recursive set is recursive.

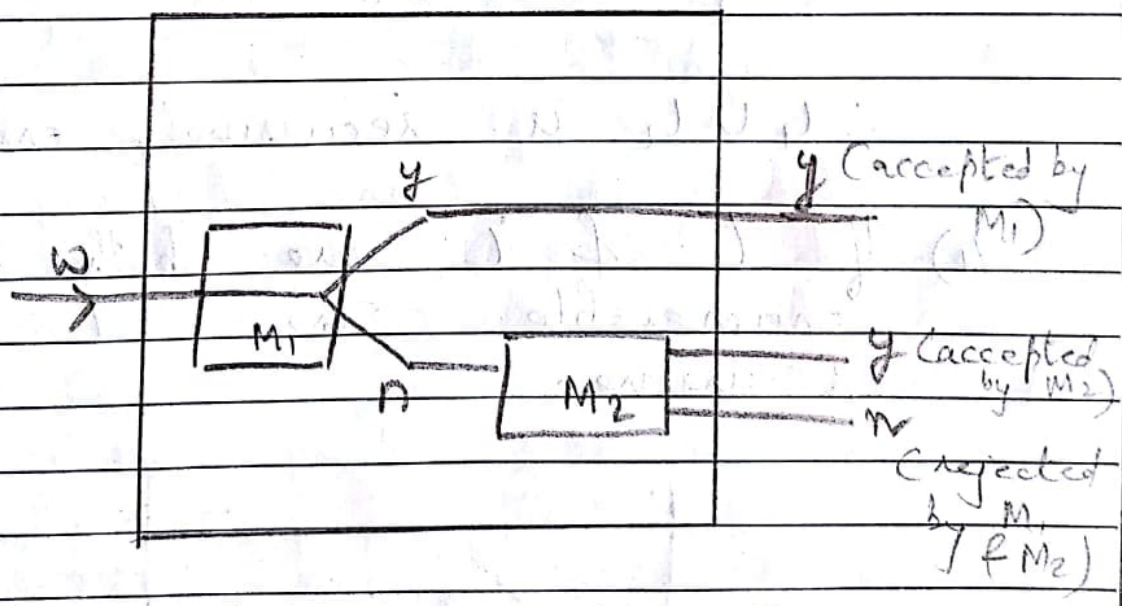
Let L be a recursive set accepted by the machine M.



∴ $\overline{L}$ is recursive. (Since $\overline{M}$ gives either y or n).

2> The union of 2 recursive set is recursive.

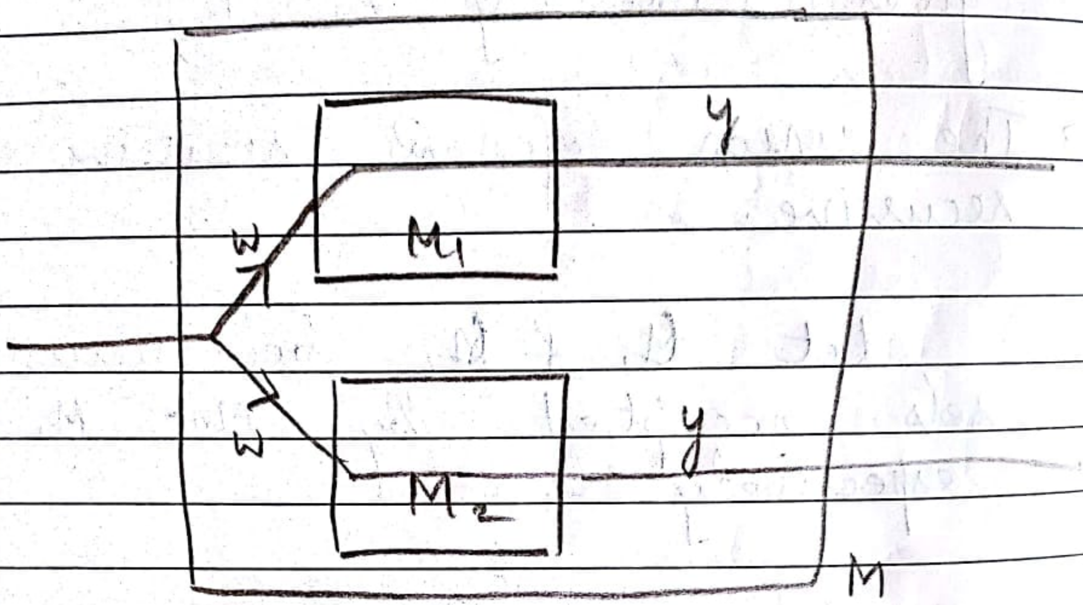
Let L_1 & L_2 be recursive sets accepted by TMs M_1 & M_2 respectively.



Here also, we get either y or n.

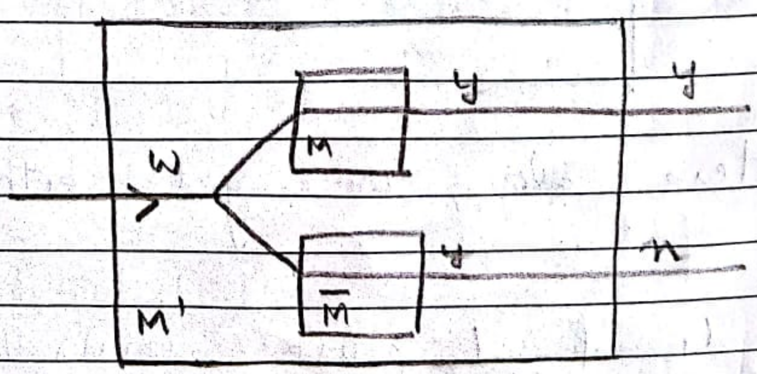
3) If L_1 & L_2 are recursively enumerable sets, then $L_1 \cup L_2$

is



$\therefore L_1 \cup L_2$ is recursively enumerable

4) If L & $\bar{L}$ are both recursively enumerable, then L is recursive.



Let L be accepted by M & $\bar{L}$ by $\bar{M}$. Given w will be either accepted by M or $\bar{M}$. L can be accepted by a TM M' which halts on all inputs & follow the process in the diagram.

Imp

Halting Problem

What is undecidability?

The concept of decidability is a break through in theoretical CS & mathematics in the 1st half of 20th century. After this it became clear that many of the known problems are undecidable.

The concept of undecidability was introduced by showing that the problem for TM is undecidable.

The halting problem for TM can be stated as follows
"Given a TM in an arbitrary configuration, will it eventually halt?"

This problem is said to be undecidable or recursively unsolvable in the sense that there cannot exist an algorithm which will take as input a description of TM T & an i/p t and say whether T on t will halt or not.

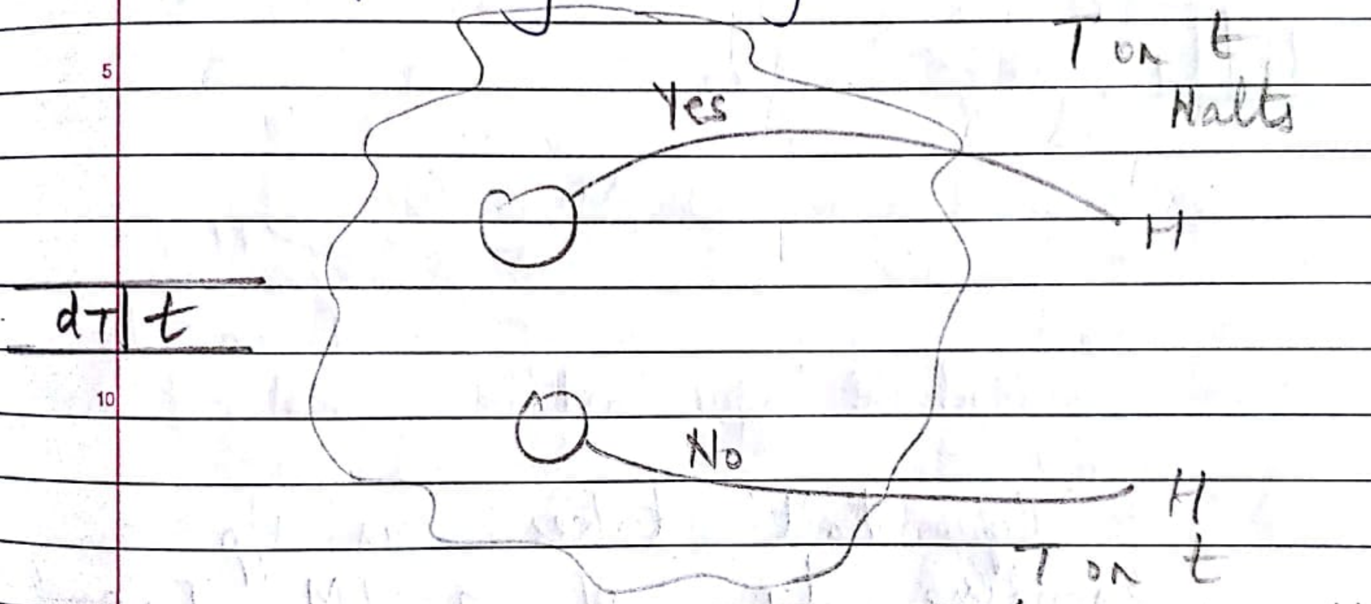
20/11/17 Eg for an undecidable problem
Hilbert's Tenth Problem

$$x^n + y^n = z^n \quad n \geq 3$$

Find x, y, z that satisfies the above equation for $n \geq 3$.

Halting Problem:

Suppose the halting problem is decidable. Then there should an algorithm to solve it. and a corresponding turing machine 'halt'

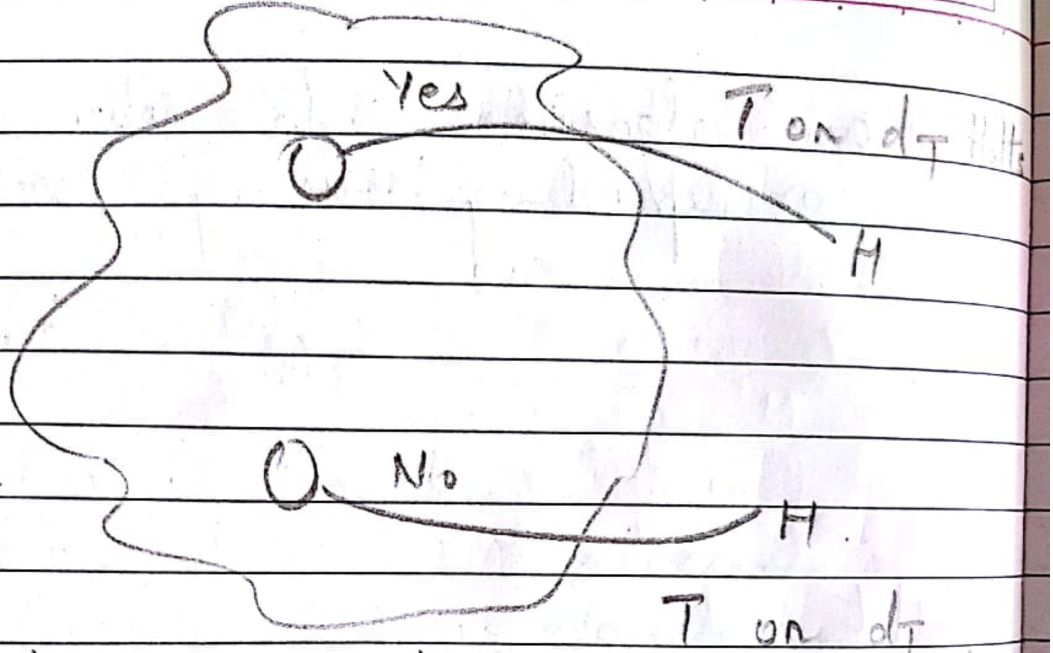


Machine 'halt'

This machine 'halt' takes as input an encoding d_T of a TM T and tells whether T on t will halt or not. The state diagram is shown above.

The machine 'halt' can be modified a little & we can think of TM 'copy halt' as follows:

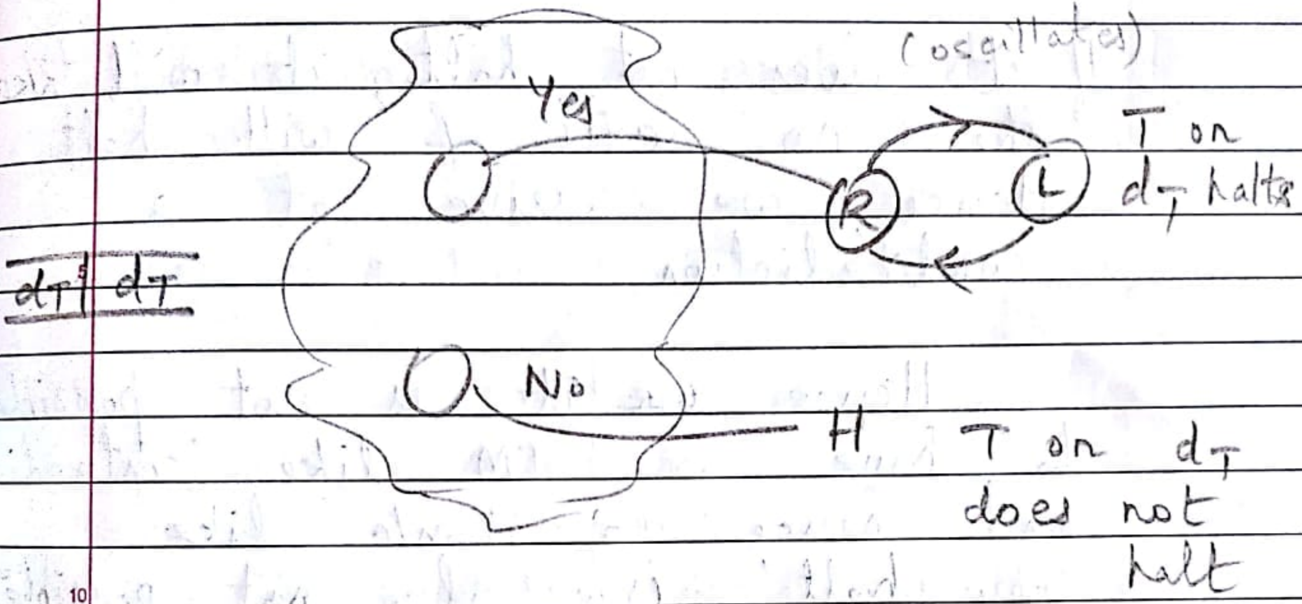
$d_T \mid d_T$



Machine 'copy halt' does not Halt

'Copy halt' takes as i/p an encoding d_T of a TM T, ~~and~~ makes a copy of d_T and simulates halt with $t = d_T$. It is clear if it is possible to have 'halt', then it is also possible to have 'copy halt'.

Now we are going to modify 'copy halt' a little & have a machine called 'contradict' as follows:



Instead of halt, it moves to R & oscillates between R & L. The machine 'contradict' is almost like 'copy halt' except that when the yes exit is taken, the m/c oscillates b/w two states & moves left & right b/w two consecutive cells.

Suppose we give as i/p to 'contradict' the encoding of contradict, it will try to find whether 'contradict' will halt on its own encoding. If it halts, it has to take the yes exit & gets into a loop.

If it does not halt, it will take the no exit & will halt. Hence we arrive at a contradiction.

Hence, ~~we~~ it is not possible to have a TM like 'contradiction' and hence a m/c like 'copy halt' is also not possible & hence 'halt' is also not possible.

Hence, the halting problem of TM is undecidable.

Decision Problem:

It can be encoded as strings & we can think of TM that will accept those strings ~~&~~ which encode the 'yes' instances & reject those strings which will encode 'no' instances of the problem.

Camlin Page
Date / /

(Satisfiability) SAT problem:

To find out whether a boolean expression is satisfiable or not.

Eg:

$$(x_1 + x_2 + x_3)(\neg x_1 + x_4 + x_5) + x_2 x_5$$

is it satisfiable or not.

Is there exist an assignment for the values $x_1, x_2, \ldots$ which will make the expression evaluate to 1 or true. If it's true, we say that the instance has a solution.

AMB Problem:

To find out whether the CFG is ambiguous or not.

A particular CFG is the instance of a problem. If this particular CFG is ambiguous, it is a 'yes' instance, otherwise it is a 'no' instance of the problem.

* All these problems are undecidable.

Camlin Page
Date / /

Optimization Problem :

Find a Hamiltonian circuit in a graph. (optimization pbm)

Find whether a graph is having a Hamiltonian ckt or not (Decision problem)

Mod 5

Pumping Lemma for CFL

Show that $a^n b^n c^n : n \geq 0$ is not CF.

Assume $a^n b^n c^n$ is CF.

Then there exists an integer N such than $|w| > N$

We choose $a^N b^N c^N$

Two cases:

1. vy contains occurences of all three symbols a, b and c.
2. vy contains occurences of only two of the three symbols.

Case 1:

Let $N = 5$

Suppose $u = aa$, $v = aaabb$, $x = bb$, $y = bccc$ and $z = cc$

Here, v & y together has all three symbols.

uv^2xy^2z is $(aa)(aaabb)^2 bb (bccc)^2 cc$

This has a 'b' before an 'a' & is $\notin C$.

Case 2:

Now suppose $u = aa$ $v = aaa$ $w = bbbbb$ $y = ccc$ $z = cc$

Here v & y together has only a and c, no b.

Then uv^2wy^2z is $(aa)(aaa)^2(bbbbb)(ccc)^2cc$

This string has 8 a, 5 b, 8 c.

So the number of a, b, c is not the same in uv^2wy^2z.

So $uv^2wy^2z \notin C$.

Both cases gives us a contradiction
Hence our assumption is wrong

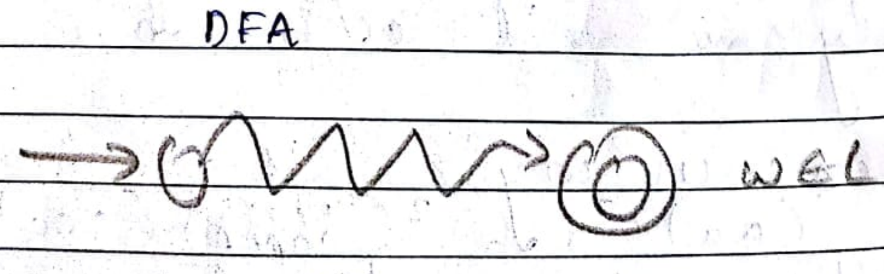
$\therefore$ $a^n b^n c^n$ is not context free

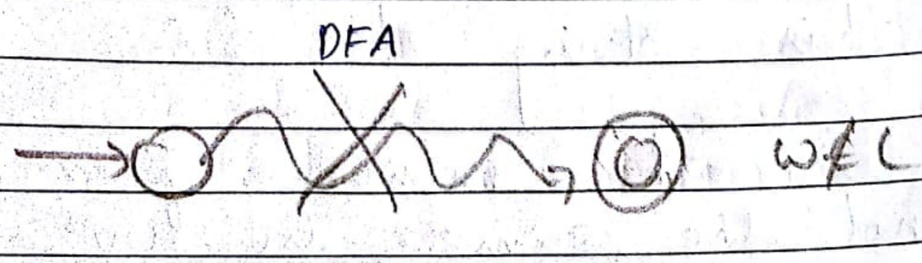
Mod 3: Decidable Problems

1) Membership Property

Qs. Given a RL L & string w how can we check if $w \in L$?

Ans. Take the DFA that accepts L & check if w is accepted.





Camlin Page
Date / /

2) Emptiness Property.

Q. Given RL L, how can we check if L is empty ($L = \phi$)?

Ans: Take the DFA that accepts L

Check if there is any path from the initial state to ~~the~~ an accepting state.

3) Finiteness

Q. Given RL L, how can we check if L is finite?

Ans: Take the DFA that accepts L

Check if there is a walk with cycle from the initial to the final state.

RL:
Pumping
closure
decision

4) Qs: Given RL L_1 and L_2 how can we check if $L_1 = L_2$?

Ans: Find if $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \phi$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \phi$$

21/11/17

Equivalence of Acceptance by Empty Store and Final State in PDA

Theorem: $L = T(M_2)$ for a PDA M_2 iff $L = N(M_1)$ for a PDA M_1.
(T: acceptance by final state & N: acceptance by empty store).

Proof:

Part 1

$L = T(M_2)$ we are supposed to construct a m/c M_1 such that $L = N(M_1)$.

$$M_2 = (K, \Sigma, \Gamma, \delta_2, q_0, z_0, F)$$

Now let us construct M_1

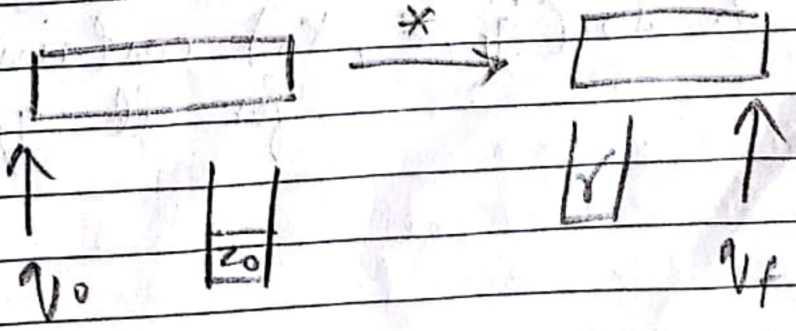
$$M_1 = (K \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_1, q_0', X_0, \phi)$$

q_e is known as the erasing state.
q_0' is the new initial state of M_1.
X_0 is the new initial stack top symbol.
Now let us design δ_1.

1. $\delta_1(q_0', \varepsilon, X_0)$ contains $(q_0, Z_0 X_0)$
2. $\delta_1(q, a, Z)$ includes $\delta_2(q, a, Z)$
3. If $q \in F$, $\delta_1(q, \varepsilon, Z)$ contains (q_e, ε)

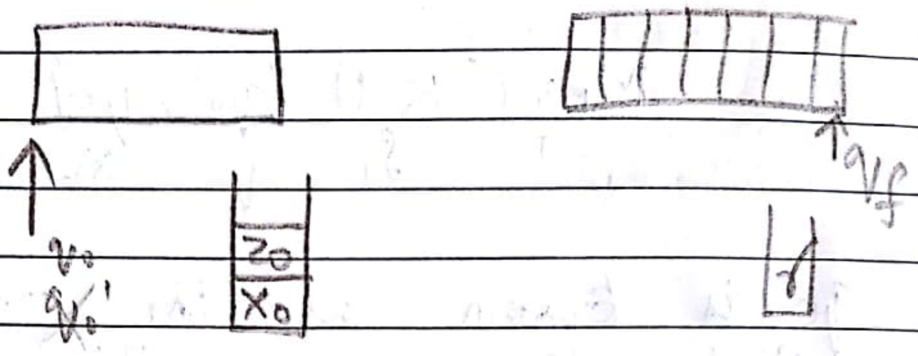
   where $Z \in \Gamma \cup \{X_0\}$
4. $\delta_1(q_e, \varepsilon, Z)$ contains (q_e, ε) for all Z in $\Gamma \cup \{X_0\}$.

M_2



FSA ⇒ DFSA
PDA ⇒ NPDA

M_1



Part 2

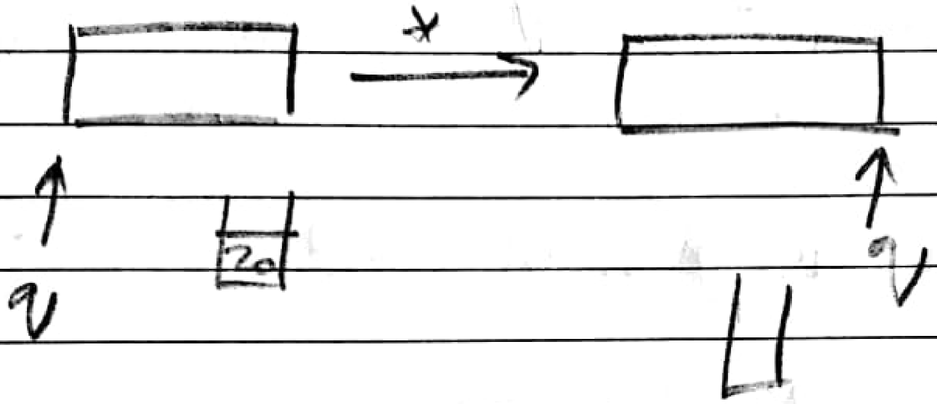
Given M_1, we must construct M_2.

$$M_1 = (K, \Sigma, \Gamma, \delta_1, q_0, z_0, \phi)$$

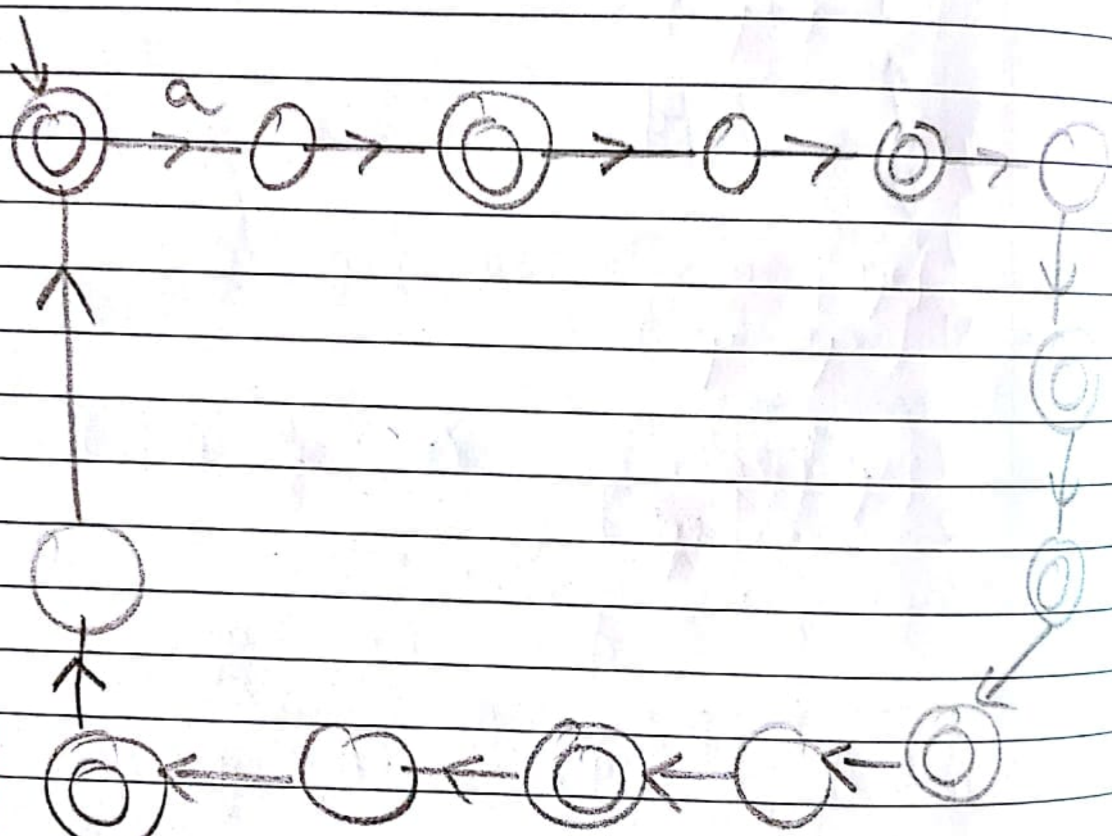
$$M_2 = (K \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_2, q_0', X_0, \{q_f\})$$

1. $\delta_2(q_0', \varepsilon, X_0)$ contains $(q_0, z_0 X_0)$
2. $\delta_2(q, a, z)$ includes $\delta(q, a, z)$
3. $\delta_2(q, \varepsilon, X_0)$ contains (q_f, X_0) for all q in K.

M1



Q. Draw a DFA that accepts strings of a's of length divisible by 2 or 7.



Conversion of NFA with ε to NFA

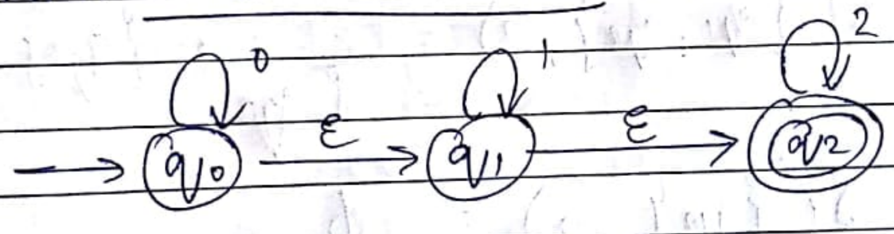
let $M = (K, \Sigma, \delta, q_0, F)$ be an NFSA with ε-moves. Then an NFSA without ε-moves M' can be constructed where $T(M) = T(M')$

$M' = (K, \Sigma, \delta', q_0, F')$

$F' = F$ if ε-closure(q_0) does not contain a state from F.

$= F \cup \{q_0\}$ if ε-closure of q_0 contains a state from F.

NFA with ε to DFA



ε-closure$(q_0) = \{q_0, q_1, q_2\}$

$\delta(\{q_0, q_1, q_2\}, 0) = \{q_0\} \cup \phi \cup \phi$
$= \varepsilon\text{-clos}\{q_0\}$
$= \{q_0, q_1, q_2\}$
$= [q_0\ q_1\ q_2]$

$\delta(\{q_0, q_1, q_2\}, 0) = [q_0\ q_1\ q_2]$

$\delta(\{q_0, q_1, q_2\}, 1) = \phi \cup \{q_1\} \cup \phi$
$= \varepsilon\text{-clos}\{q_1\}$
$= \{q_1, q_2\} = [q_1\ q_2]$

$$\delta(\{q_0, q_1, q_2\}, 2) = \phi \cup \phi \cup \{q_2\}$$
$$= \epsilon\text{-clos}\{q_2\}$$
$$= \{q_2\}$$
$$= [q_2]$$

$$\delta(\{q_1, q_2\}, 0) = \phi$$

$$\delta(\{q_1, q_2\}, 1) = \varepsilon\text{-clos}\{q_1\}$$
$$= [q_1\ q_2]$$

$$\delta(\{q_1, q_2\}, 2) = \varepsilon\text{-clos}\{q_2\}$$
$$= [q_2]$$

$$\delta(\{q_2\}, 0) = \phi$$

$$\delta(\{q_2\}, 1) = \phi$$

$$\delta(\{q_2\}, 2) = \{q_2\} = [q_2]$$

	0	1	2
$[q_0\ q_1\ q_2]$	$[q_0\ q_1\ q_2]$	$[q_1\ q_2]$	$[q_2]$
$[q_1\ q_2]$	ϕ	$[q_1\ q_2]$	$[q_2]$
$[q_2]$	ϕ	ϕ	$[q_2]$
ϕ	ϕ	ϕ	ϕ

22/11/17

Myhill - Nerode Theorem

Use: In minimization of DFA.
(To prove a language is non-regular).

Theorem:
The following three statements are equivalent.
1) The set $L \subseteq \Sigma^*$ is accepted by some FSA.
2) L is the union of some of the equivalent classes of a right invariant equivalence relation of finite index.

3) Let equivalence relation R_L be defined by $x R_L y$ iff for all z in Σ^*, xz is in L exactly when yz is in L. Then R_L is of finite index.

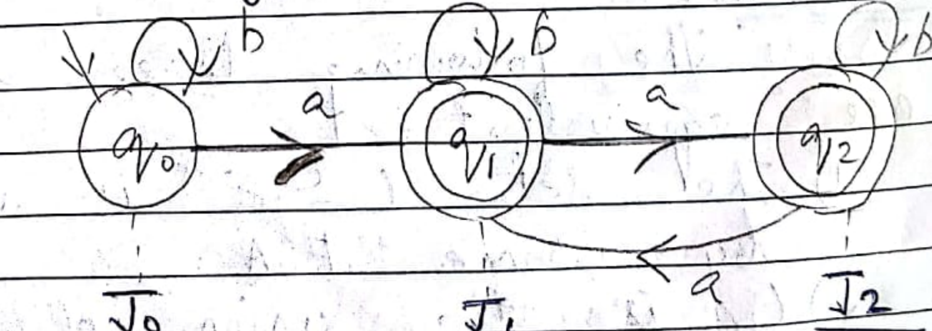
Note: No. of equivalent classes is called index.

Let L be accepted by a DFSA, $M = (K, \Sigma, \delta, q_0, F)$

Let us define an equivalence relation R_M on Σ^*.

$x R_M y$ iff $\delta(q_0, x) = \delta(q_0, y)$

Eg:



J_0	J_1	J_2
$q_0 \to q_0$	$q_0 \to q_1$	$q_0 \to q_2$
b	a	aa
bb	ab	abab
bbb	ba	aabaab
⋮	abb	
	abaa	

J_0 J_1 J_2

babaa babab

Finite index = 3

J_0 : No a's at all

J_1 : Odd no. of a's

J_2 : Even no. of a's.

$L = J_1 \cup J_2$

J_0 is not a part of L.

Take '$\cup$' of classes corresponding to final states.

Right Invariant Equivalence Relation

If $x R_M y$, then for any z

$xz R_M yz$

Then R_M is right invariant.

* For a RIE relation, we have finite index

<u>Note</u>: $x R_M y$

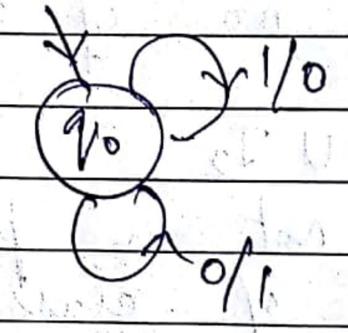
$$\delta(q_0, x) = \delta(q_0, y)$$

$$\delta(q_0, xz) = \delta(\delta(q_0, x), z)$$

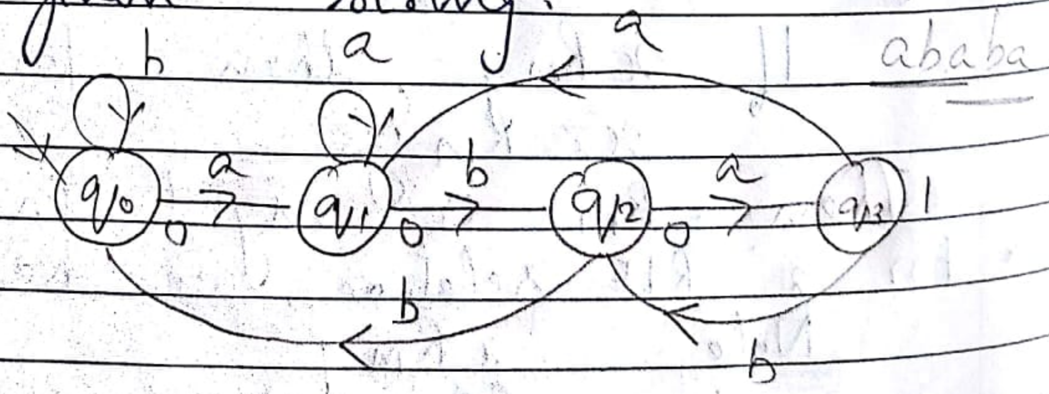
$$= \delta(\delta(q_0, y), z)$$

$$= \delta(q_0, yz)$$

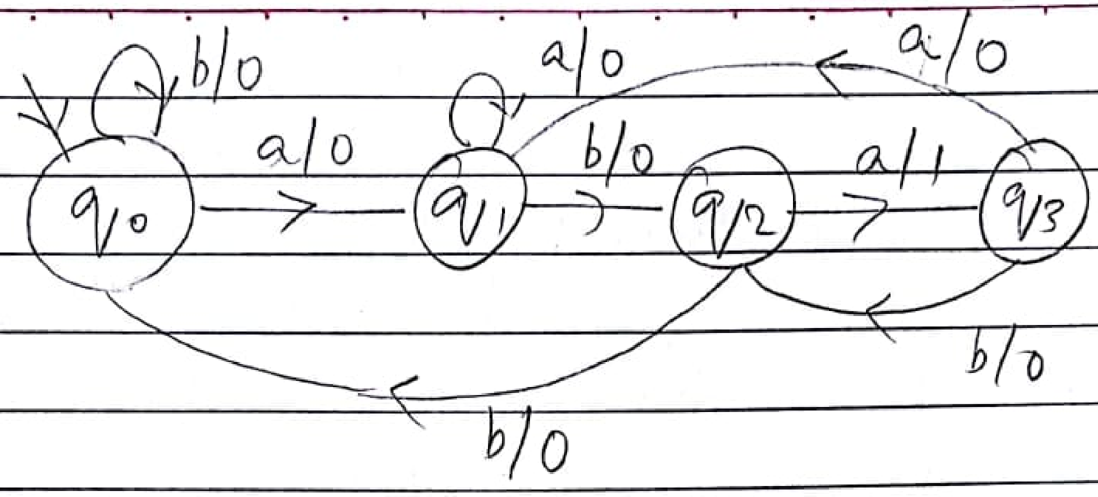
(Construct a)
Q. Design a Mealy machine to get complement of a binary number.



Q. Design a Moore machine which counts the occurences of aba in a given string.



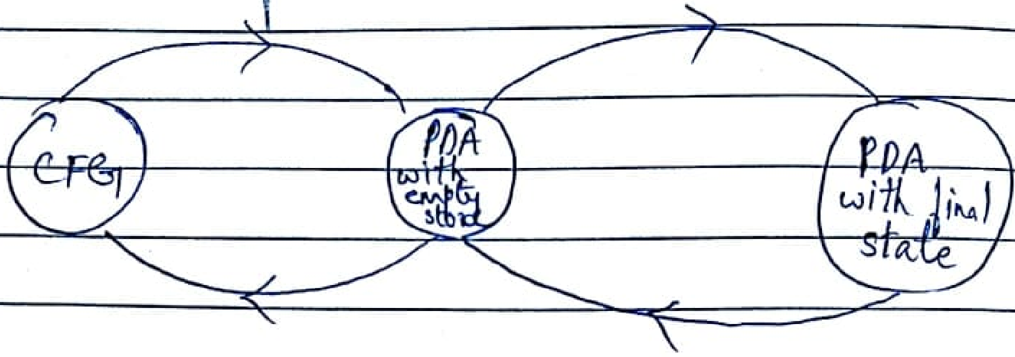
Q. Design a Mealy machine that counts the occurance of aba



Equivalence of Regular Expression & NFA with ε

Theorem: If R is a regular expression representing a regular set, we can construct an NFA with ε moves to accept r.

* Given G, how to construct its equivalent PDA



$N(M_1) = T(M_2)$

Let L be generated by a CFG G. Then L can be accepted by PDA M by empty store.

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

$$M = (\{q\}, T, N \cup T, \delta, q, S, \phi)$$

δ: If $A \to \beta$ is in P, $\beta \in (N \cup T)^*$,

Then $\delta(q, \varepsilon, A)$ contains (q, β)

For each a in T,

$\delta(q, a, a)$ contains (q, ε)

Eg: G: $S \to aSb$

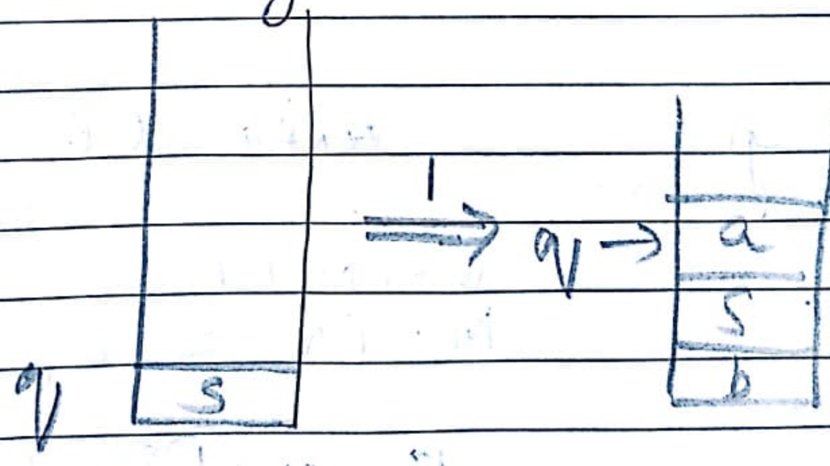
$S \to ab$

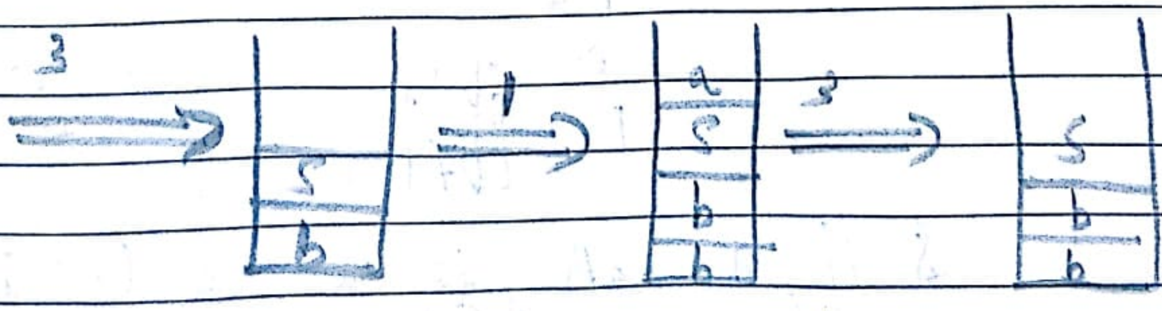
$$M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \phi)$$

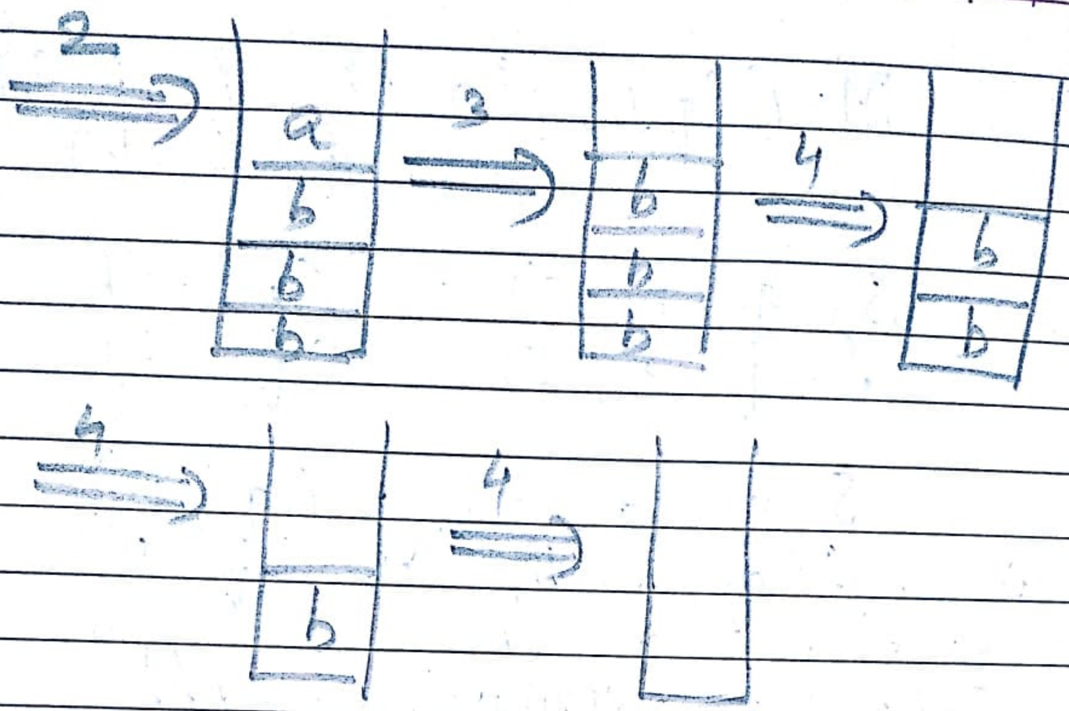
$$\delta(q, \varepsilon, S) = (q, aSb) \quad (1)$$
$$\delta(q, \varepsilon, S) = (q, ab) \quad (2)$$

$$\delta(q, a, a) = (q, \varepsilon) \quad (3)$$
$$\delta(q, b, b) = (q, \varepsilon) \quad (4)$$

Take a string aaabbb







Given a ~~conte~~ R.G, construct FSA

$G = (N\ T\ P\ S)$

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

$Q = N \cup \{q_f\}$

$T = \Sigma$

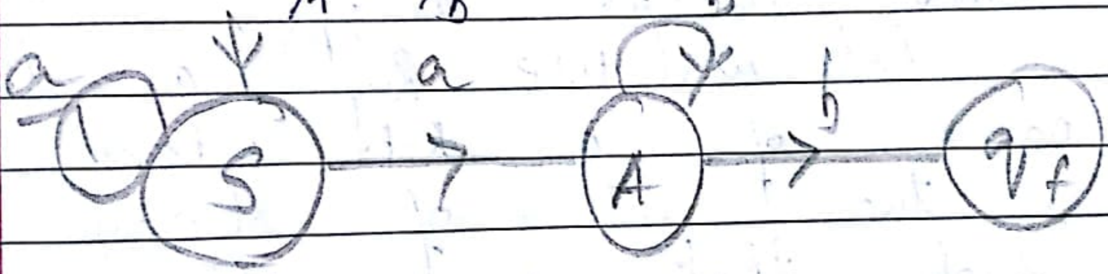
$S = q_0$

$F = \{q_f\}$

δ: If $A \rightarrow aB$ is a rule in P, we have $\delta(A, a)$ contains B.

If $A \to a$ is a rule in P, then $\delta(A, a)$ contains q_f.

Eg: $S \to aS$
$S \to aA$
$A \to bA$
$A \to b$

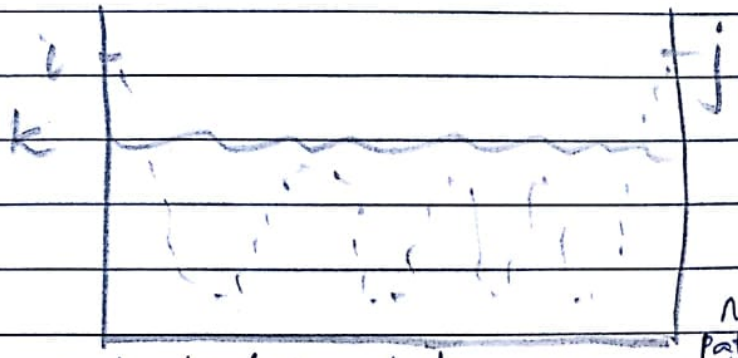


$\delta(S, a) = S$
$\delta(S, a) = A$
$\delta(A, b) = A$
$\delta(A, b) = q_f$

The language accepted is $a^m b^n$

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

Let us suppose that A's states are $1, 2, 3 \ldots n$ for some integer 'n'.
Let us use $R_{ij}^{(k)}$ as the name of a R.E whose language is the set of strings w such that w is the label of a path from state i to j in A & that path has no intermediate node whose number is greater than k.
Note that the beginning & end pts of the path are not intermediate. So there is no constraint that ~~i for~~ i and/or j be less than or equal to k.



Notice that in this diagram, we have shown both i & j to be > k. Also note that the path passes through node k twice but never goes to a state higher than k except at end pts.

To construct the expressions $R_{ij}^{(k)}$ we use the following inductive definition starting at $k=0$ & finally reaching $k=n$.

Basis:

The basis is $k=0$

Since all states are numbered 1 or above, the restrictions on path is that the path must have no intermediate states at all. There are only two kinds of path that meet such a condition.

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