Theory of Computing

- Formal Language
- Automata Theory

FORMAL LANGUAGE

*. Alphabet (Σ) → A finite non-empty set of symbols

Eg:

$\Sigma = \{A, B, C, \ldots, Z\}$

$\Sigma = \{0, 1, 2, \ldots, 9\}$

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

*. Strings/Words → A finite sequence of symbols over Σ

Eg: CAT, ABC, RXZJ, BAT over $\Sigma = \{A, B, C, \ldots, Z\}$

*. Σ^2 → Set of all strings of length 2

Eg: If $\Sigma = \{0,1\}$ then $\Sigma^2 = \{00, 01, 10, 11\}$

*. λ (or ϵ) → A word of length 0

*. $\Sigma^\circ = \{\lambda\}$ for all Σ

*. Σ^* → The set of all words (Infinite set)

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \Sigma^\circ \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$

Countable → Finite, Infinite

Uncountable → Infinite

*. $\Sigma^+ = \Sigma^* - \{\lambda\}$ → Set of all strings of non-zero length

*. Let $u = 01011$ and $v = 0001$

Concatenation $uv = 010110001$

*. $uv \neq vu$

$uv = vu$ if $u = v$ or $u = \lambda$ or $v = \lambda$ or $u = v^n$ or $v = u^n$

$v = 001$

$u = 001001$

*. $|\lambda| = 0$

*. $\lambda u = u\lambda = u$

*. $|u| = 0$ iff $u = \lambda$

Proof by contrapositive.

If $u \neq \lambda$,

$|u| > 0 = n \Rightarrow u \neq \lambda$

For any $a \in \Sigma^1$,

$|a| = 1$

Let $w \in \Sigma^*$

$|wa| = |w| + 1 = |aw|$

Proof continued,

Using the above definition,

$|u| \geq 1 \Rightarrow u = a\beta$ where $a \in \Sigma$ and $\beta \in \Sigma^*$

*. $|u^n| = n|u|$

*. $|uv| = |u| + |v|$

*. If $u = a_1 a_2 \ldots a_{n-1} a_n$

$u^R = a_n a_{n-1} \ldots a_2 a_1$

*. $|u| = |u^R| = n$

*. $(u^R)^R = u$

*. Language → Any subset of Σ^*

$L \subseteq \Sigma^*$

Eg: ϕ (No elements), $\{\lambda\}$ (An element with 0 alphabets), Σ^*

*. $L_1 = \{(01)^n : n \geq 1\} = \{01, 0101, 010101, \ldots\}$

$L_2 = \{0^n 1^n : n \geq 1\} = \{01, 0011, 000111, \ldots\}$

$L_1 \neq L_2$

*. $L_1 = \{0^n 1^m : n, m \geq 0\}$

$L_2 = \{0^n 1^n : n \geq 0\}$

$L_2 \subset L_1$

*. $L_1 \cup L_2 \rightarrow \{w : w \in L_1 \text{ or } w \in L_2\}$

*. $L_1 \cap L_2 \rightarrow \{w : w \in L_1 \text{ and } w \in L_2\}$

*. $L_1^R \rightarrow \{w^R : w \in L_1\}$

*. If $L_1 = \{aw : w \in \Sigma^*\}$ where $\Sigma = \{a, b\}$

$L_1^R = \{wa : w \in \Sigma^*\}$

*. If $L_1 \subseteq \Sigma^*$,

$L_1^c = \{w : w \notin L_1\}$

*. If $L_1 = \{aw : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$

Then $L_1^c = \{bw : w \in \Sigma^*\} \cup \{\lambda\}$

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

$L_1 = \{0w : w \in \Sigma^*\}$

$L_1^R = \{w0 : w \in \Sigma^*\}$

$L_1^c = \{1w : w \in \Sigma^*\} \cup \{\lambda\}$

$L_2 = \{0w1 : w \in \Sigma^*\}$

$L_2^R = \{1w0 : w \in \Sigma^*\}$

$L_2^c = \{1w0 \cup 0w0 \cup 1w1 : w \in \Sigma^*\} \cup \{\lambda, 0, 1\}$ $L_2^c = \{1w \cup w0\} \cup \{\lambda, 0, 1\}$

$L_3 = \{w_1 01 w_2 : w \in \Sigma^*\}$

$L_3^R = \{w_1 10 w_2 : w \in \Sigma^*\}$

$L_3^c = \{1^m 0^n : m, n \geq 0\}$

$L_4 = \{0^n 1^n : n \geq 0\}$

$L_4^R = \{1^n 0^n : n \geq 0\}$

$L_5 = \{w : |w|_0 = |w|_1\}$

$L_5^R = L_5$

$L_6 = \{waw^R : a \in \Sigma \cup \{\lambda\}, w \in \Sigma^*\}$

$L_6^R = L_6$

$L_7 = \{ww^R : w \in \Sigma^*\}$

$L_7^R = L_7$

$L_8 = \{w : |w| \equiv 0 \bmod 3\}$

$L_8^R = L_8$

*. Concatenation:

$L_1 \cdot L_2 = \{w_1 \cdot w_2 : w_1 \in L_1,\ w_2 \in L_2\}$

$(\{w : w \in \Sigma^*\} \cup \{\lambda\}) \cdot (\{w0 : w \in \Sigma^*\}) = \{w0 : w \in \Sigma^*\} \cup \{w0 : w \in \Sigma^*\}$

Due to → Same ←

*. $L^2 = L \cdot L$

*. L^* = $L^0 \cup L^1 \cup L^2 \cup \ldots$ → Contains all possible numbers of concatenations of L

↓ Kleene Closure

*. $L_1 = \{w : |w| \text{ is even}\}$

$L_1^* = L_1$

$L_2 = \{w : |w| \text{ is odd}\}$

$L_2^2 = L_1 - \{\lambda\}$

$L_2^3 \neq L_2$ ∵ L_2^3 is at least of length 3 since $\lambda \notin L_2$

$L_2^* = \Sigma^*$ $L_2^* = L_2^0 \cup L_2^1 \cup L_2^2 \cup \ldots$

L_2^0 → $\{\lambda\}$; L_2^1 → All odd length strings; L_2^2 → All even length strings except $\{\lambda\}$

Properties of Languages:

$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$

$L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$

$L^* = L^+ \cup \{\lambda\}$

$L^k = L^{k-1} \cdot L = L \cdot L^{k-1}$

i) Associativity:

$(L_1 L_2) L_3 = L_1 (L_2 L_3) = L_1 L_2 L_3$ [As long as order is the same brackets do not matter]

ii) Not commutative:

$L_1 L_2 \neq L_2 L_1$

iii) $L \cdot \{\lambda\} = \{\lambda\} \cdot L = L$

iv) $L \cdot \phi = \phi \cdot L = \phi$

v) Distributive:

$(L_1 \cup L_2) \cdot L_3 = (L_1 L_3) \cup (L_2 L_3)$

vi) $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$

vii) If $L_1 \subseteq L_3$ and $L_2 \subseteq L_4$
then $L_1 \cdot L_2 \subseteq L_3 \cdot L_4$

viii) $\phi^* = \{\lambda\} = \{\lambda\}^*$

ix) If $\lambda \in L$,
then $L^* = L^+$

Proof:
$L \subseteq L^+$
$\lambda \in L \Rightarrow \lambda \in L^+$
$L^+ \cup \{\lambda\} = L^+$ [$\because \lambda \in L^+$]
$L^* \cup \{\lambda\} = L^*$ [By definition]
$\therefore L^+ = L^*$

x) $(L^*)^* = L^*$

xi) $L^* \cdot L = L \cdot L^* = L^+$
If $\lambda \notin L$,
$L^* \cdot L = (L^+ \cdot L) \cup (L^1 \cdot L) \cup \ldots$ ($\{\lambda\}$ will be lost)
If $\lambda \in L$,
Use property (ix)

xii) $L^* L^* = L^*$

xiii) $(L_1 \cup L_2)^* = (L_1^* L_2^*)^*$

Proof:
$L_1 \subseteq L_1^* \subseteq L_1^* L_2^*$
$L_2 \subseteq L_2^* \subseteq L_1^* L_2^*$
$\therefore L_1 \cup L_2 \subseteq L_1^* L_2^*$
$\therefore (L_1 \cup L_2)^* \subseteq (L_1^* L_2^*)^*$
$\therefore (L_1 \cup L_2)^* = (L_1^* L_2^*)^*$

$L_1^* \subseteq L_1 \cup L_2$
$L_1^* \subseteq (L_1 \cup L_2)^*$
Similarly,
$L_2^* \subseteq (L_1 \cup L_2)^*$
Using property 7,
$L_1^* \cdot L_2^* \subseteq (L_1 \cup L_2)^* \cdot (L_1 \cup L_2)^*$
Using property 12,
$L_1^* \cdot L_2^* \subseteq (L_1 \cup L_2)^*$
$\therefore (L_1^* \cdot L_2^*)^* \subseteq ((L_1 \cup L_2)^*)^* = (L_1 \cup L_2)^*$
[Property 10]

Regular Expression (R):

Let R be a set such that,

A language (L) is just the concatenation and union of symbols [But it is not true]

$\lambda \in R$

$\phi \subseteq R$

$a \in R \quad \forall\, a \in \Sigma$

If $r_1, r_2 \in R$,

$r_1 + r_2 \in R$ [Union] (+ : Logical OR)

$r_1 \cdot r_2 \in R$ [Concatenation]

$r_1^* \in R$ [Closure]

$(r_1) \in R$

Eg: $(a+b)^* \cdot ab \cdot (a+b)^*$ where $a, b \in \Sigma$

*. For $\Sigma = \{a, b\}$

$(a+b)^*$ represents Σ^*

*. set of strings starting with 'a',

$a \cdot (a+b)^*$

*. Set of strings having 'ab' as substring,

$(a+b)^* \cdot a \cdot b \cdot (a+b)^*$

*. Set of strings of even length:

$[(a+b) \cdot (a+b)]^*$

*. Set of strings of odd length:

$[(a+b) \cdot (a+b)]^* \cdot (a+b)$

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

*. Set of strings that start and end with the same symbol:

$a \cdot (a+b+c)^* \cdot a + b \cdot (a+b+c)^* \cdot b + c \cdot (a+b+c)^* \cdot c + (a+b+c)$

Let $\Sigma = \{a, b, c, \ldots, z\}$

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

*. At least two zeros:

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

*. Exactly two zeros:

$$1^* \cdot 0 \cdot 1^* \cdot 0 \cdot 1^*$$

*. No. of zeros is even:

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

*. At most two zeros:

$$\underbrace{1^*}_{\text{No zeros}} + \underbrace{1^* \cdot 0 \cdot 1^*}_{\text{Exactly one zero}} + \underbrace{1^* \cdot 0 \cdot 1^* \cdot 0 \cdot 1^*}_{\text{Exactly two zeros}}$$

(Or)

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

*. $L = \{a^n b^m ; n, m \geq 0\}$

$R = a^* b^*$

*. $L = \{a^n b^m ; n, m \geq 1\}$

$R = aa^* bb^*$

*. $L = \{a^n b^n ; n \geq 0\}$ → No regular expression exists ⇒ Language ≠ Regular Expression

Properties of Regular Expression Symbol r: $(r \in R)$

*. $r^* \cdot r^* = r^*$

*. $r \cdot \lambda = \lambda \cdot r = r$

*. $r \cdot \phi = \phi \cdot r = \phi$

*. $(r_1 + r_2)^* = (r_1^* r_2^*)^*$

*. $r + \lambda = \lambda + r$

*. $r + \phi = \phi + r = r$

*. A language L that can be represented by a Regular Expression is called a Regular Language

*. If L is a regular language then L^* is also regular.

*. L' is regular ⇒ There exists a regular expression r such that $L' = L(r)$

*. If r is a regular expression r^* is also a regular expression

*. $L(r^*) = L^*$

*. If L_1 and L_2 are regular expressions, then $L_1 \cup L_2$ is also regular

$L_1 = L(r_1)$ $L_2 = L(r_2)$

$r_1 + r_2$ represents $L_1 \cup L_2$

$L_1 \cup L_2 = L(r_1 + r_2)$

*. $r_1 \cdot r_2$ represents $L_1 \cdot L_2$

$L_1 = L(r_1)$

$L_2 = L(r_2)$

$L_1 \cdot L_2 = L(r_1 \cdot r_2)$

*. If L_1 and L_2 are regular languages,

a. $L_1 \cup L_2$ is regular

b. $L_1 \cdot L_2$ is regular

c. L_1^* is regular

L_1^c

$L_1 \cap L_2$ } Regular

L_1^R

AUTOMATA THEORY

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

$L_1 = \{a^n b : n \geq 0\}$

$r = a^*b$

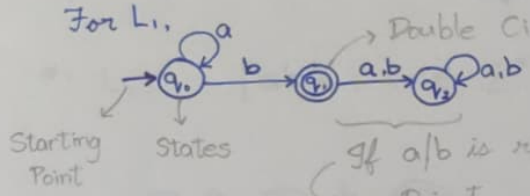
$L(r) = L_1 \Rightarrow L_1$ is regular

Regular expression is a representation of the strings in a language

We wish to construct a set of rules that will accept the strings of a specific language

Three ways to represent rules:

i) Transition Figure:



*. If the final state is a double circle, accept the string:
Else, reject

ii) Table:

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

iii) Transition Function:

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

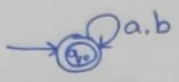
$\delta(q_1, a) = q_2$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_2$

Eg:

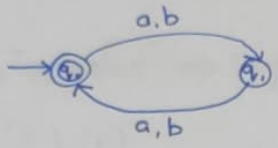


$L = \Sigma^*$

---

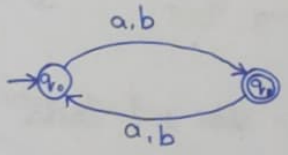
*. $\lambda \in L$ if and only if the initial state is a final state.

Eg:



Language of strings of even length

Eg: Language of strings of odd length:



Deterministic Finite State Automaton:

A deterministic finite state automaton (DFA) is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$

where,

$Q \rightarrow$ A finite set of states

$\Sigma \rightarrow$ Finite non-empty alphabet set

$q_0 \rightarrow$ Initial state $[q_0 \in Q]$. {There is only one initial state}

$F \rightarrow$ Set of final states $[F \subseteq Q]$

$\delta \rightarrow$ $\delta : (Q \times \Sigma) \rightarrow Q$

δ is a total function $\{\delta(q, x)$ is defined $\forall q \in Q$ and $\forall x \in \Sigma\}$

Extendible Transition Function (δ^*):

$\delta^*(q_0, aba) = q_2$ Starting at q_0, what is the final state after reading the string alphabet by alphabet

$\delta^* : (Q \times \Sigma^*) \rightarrow Q$

*. $\delta^*(q_i, \lambda) = q_i$

*. $\delta^*(q_i, wa) = \delta(q_j, a) = \delta(\delta^*(q_i, w), a)$ $\left[\begin{smallmatrix} w \in \Sigma^* \\ a \in \Sigma \end{smallmatrix}\right]$

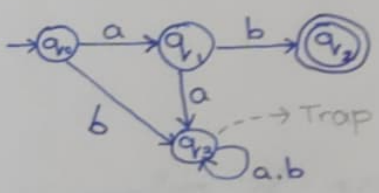
*. A string $w \in L$ is accepted iff $\delta^*(q_0, w) \in F$

*. Set of accepted strings is called the language of the machine $L(M)$

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

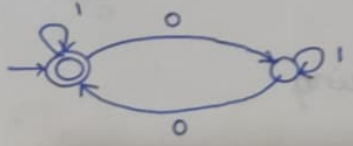
Construction of DFA :

*. $L_1 = \{abw : w \in \Sigma^*\}$

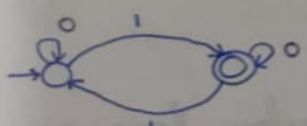


--→ Trap state / Dead state

*. L_2 = Strings with even 0's



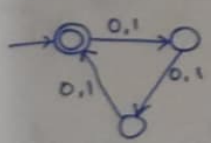
*. L_3 = String with odd 1's



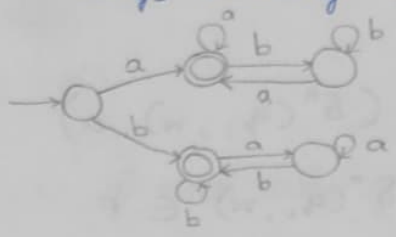
*. L_4 = Strings with length as a multiple of 3

$|w| \equiv 0 \mod 3$

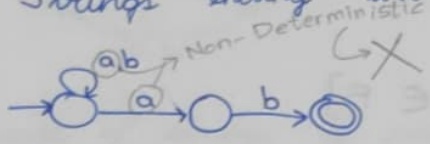
$[|w| \bmod 3 = 0]$

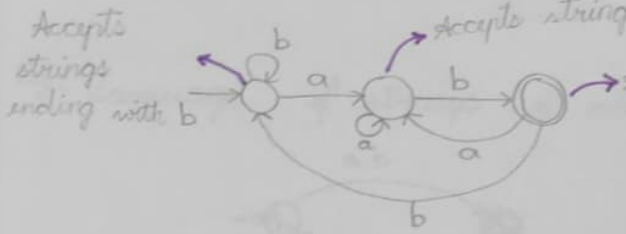


*. L_5 = Strings starting and ending with same symbol



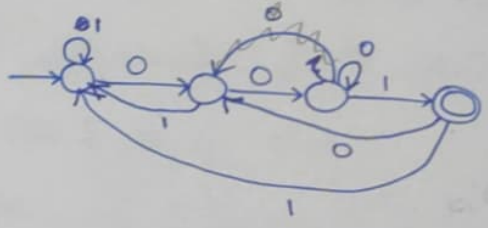
*. L_6 = Strings ending with ab



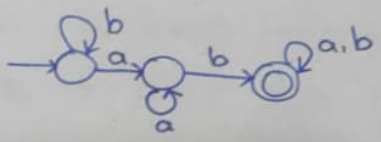


$\Sigma = \{0, 1\}$

*. Strings ending with 001



*. Strings with ab as substring



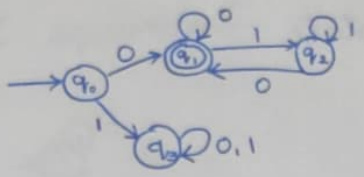
Regular Language:

A language L is said to be a regular language iff there exists a DFA M such that $L = L(M)$

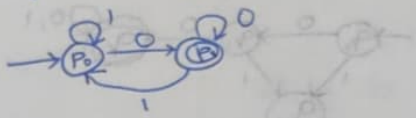
*. ∴ Every DFA has a regular expression and every regular expression has a DFA

*. Complement of a regular language is also regular.
(Just invert the states: Final → Non final
Non final → Final)

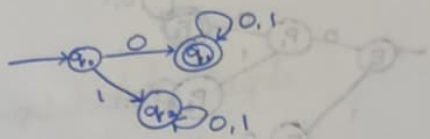
*. Strings starting and ending with 0 over $\Sigma = \{0, 1\}$



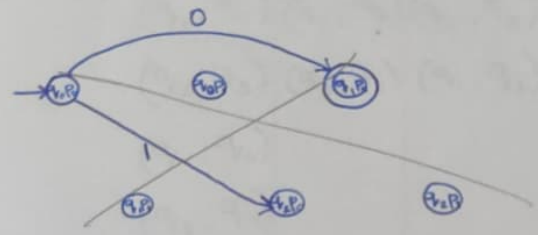
*. Strings ending with 0

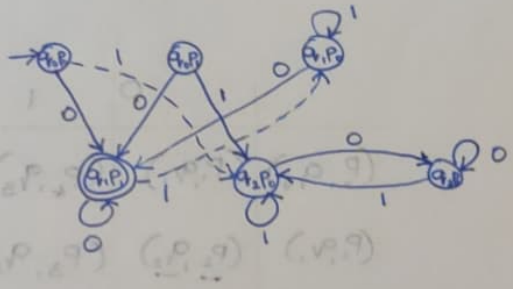


*. Strings starting with 0

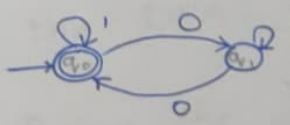


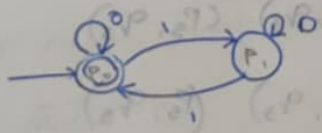
AND

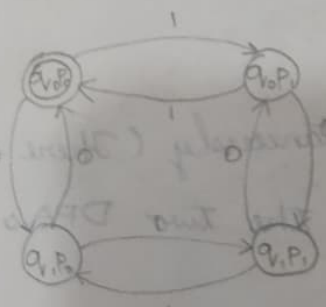




*. Even number of zeros and even number of ones:







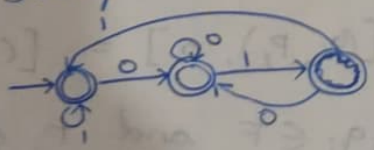
$F = \{q_0P_0\}$ => Even 0's and even 1's

$F = \{q_1P_0\}$ => odd 0's and even 1's

Similarly for q_0P_1 and q_1P_1

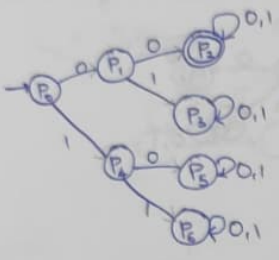
Strings ending with 01

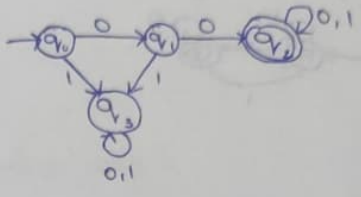
String not ending with 01



*. A DFA will accept some string(s) iff there is a path from the initial state to a final state

Equivalence of DFAs:





	0	1
(P_0, q_{v_0})	(P_1, q_{v_1})	(P_4, q_{v_3})
(P_1, q_{v_1})	(P_2, q_{v_2})	(P_3, q_{v_3})
(P_4, q_{v_3})	(P_5, q_{v_3})	(P_6, q_{v_3})
(P_2, q_{v_2})	(P_2, q_{v_2})	(P_2, q_{v_2})
(P_3, q_{v_3})	(P_3, q_{v_3})	(P_3, q_{v_3})
(P_5, q_{v_3})	(P_5, q_{v_3})	(P_5, q_{v_3})
(P_6, q_{v_3})	(P_6, q_{v_3})	(P_6, q_{v_3})

The final states always happen simultaneously (There is no Tuple with only one final state). Therefore, the two DFAs are equivalent

*. If $\exists$ a Tuple and $a \in \Sigma$ such that

$$[(q_{v_k}, P_l), a] = [(q_{v_i}, P_j)]$$

Such that, $q_{v_i} \in F$ and $P_j \notin F$

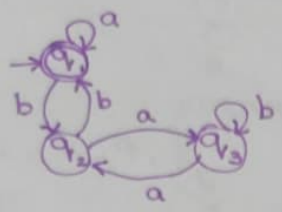
or

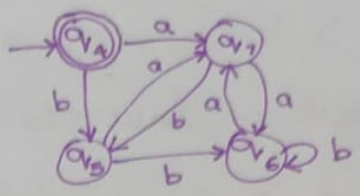
$q_{v_i} \notin F$ and $P_j \in F$,

then the two states (q_{v_k}, P_l) are said to be distinguishable

*. If ∃ a distinguishalele pair of states then the two DFAs are not equivalent.

Eg:





	a	b
(q_1, q_4)	(q_1, q_7)	(q_2, q_5)
(q_1, q_7)	(q_1, q_6)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)
(q_1, q_6)		
(q_3, q_7)		

→ Distinguishalele Tuple

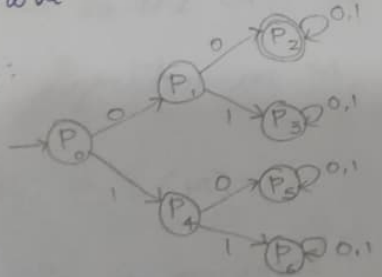
⇓

Not equivalent

HOPCROFT ALGORITHM for minimizing the number of states in the DFA:

i) Remove the inaccessible states k → ∃ no path from initial state to k

ii) Eg:



Collect all the non-final states and final states

Non-final: $\{P_0, P_1, P_3, P_4, P_5, P_6\}$ Final: $\{P_2\}$

or

$\{P_0, P_2, P_3, P_5, P_6\}$

↓

Belongs to the final states set

∴ P_1 has a different behaviour

$\{P_0, P_3, P_4, P_5, P_6\}$ $\{P_1\}$ $\{P_2\}$

↓

$\{P_0, P_3, P_4, P_5, P_6\}$

$\{P_3, P_4, P_5, P_6\}$ $\{P_0\}$ $\{P_1\}$ $\{P_2\}$

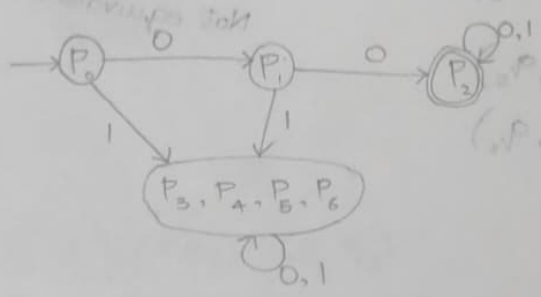
0 ↙ ↘ 1

$\{P_3, P_5, P_5, P_6\}$ $\{P_3, P_4, P_5, P_6\}$

$\therefore$ P_3, P_4, P_5 and P_6 have the same behaviour

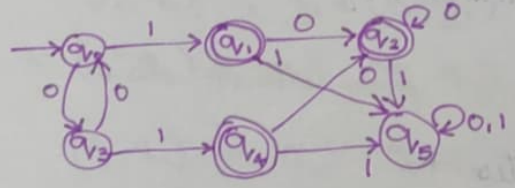
$\therefore$ P_3, P_4, P_5 and P_6 are indistinguishable

P_3, P_4, P_5, P_6 is considered as a single state



Q Minimize the following DFA:

1.



$\{q_0, q_3, q_5\}$ $\{q_1, q_2, q_4\}$

0 ↙ ↘ 1 → Odd one out

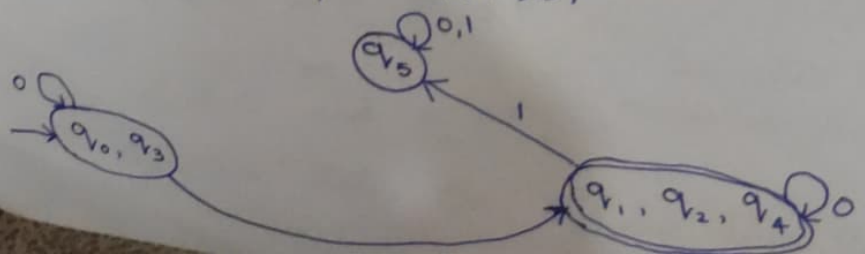
$\{q_3, q_0, q_5\}$ $\{q_1, q_4, \underline{q_5}\}$

$\{q_0, q_3\}$ $\{q_5\}$ $\{q_1, q_2, q_4\}$

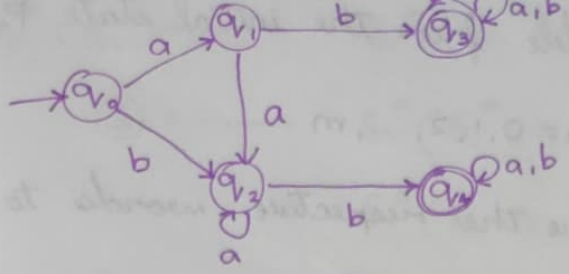
0 ↙ ↘ 0 ↙ ↘ 1

$\{q_3, q_0\}$ $\{q_1, q_4\}$ $\{q_2, q_2, q_2\}$ $\{q_5, q_5, q_5\}$

$\therefore$ The minimized DFA is,



2.



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

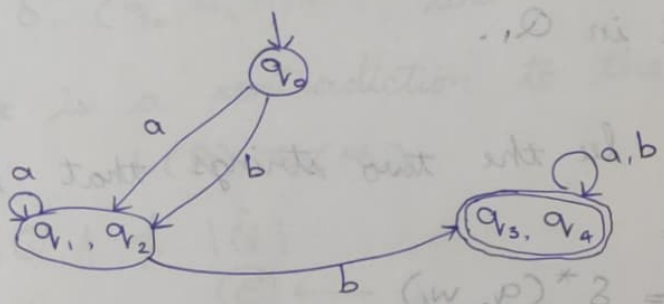
a b (Odd one out)

$\{q_1, q_2, q_2\}$ $\{q_2, q_3, q_4\}$

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

a b a b

$\{q_2, q_2\}$ $\{q_3, q_4\}$ $\{q_3, q_4\}$ $\{q_3, q_4\}$



Correctness of the Algorithm:

Let M be the DFA to be minimized. By applying Hopcroft algorithm on M, let $\hat{M}$ be the resultant DFA.

$L(\hat{M}) = L(M)$

Claim: $\hat{M}$ is minimal

Suppose there exists M_1 such that M_1 is minimal and $|Q_1| \leq |\hat{Q}|$

$L(M_1) = L(M)$

$L(M_1) = L(M) = L(\hat{M})$ ⟶ ①

Let $\hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, P_0, \hat{F})$, $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$

Let $\hat{Q} = \{P_0, P_1, \ldots, P_m\}$, $Q_1 = \{q_0, q_1, \ldots, q_n\}$

$\hat{M}$ has no inaccessible states [∵ we remove all the inaccessible states in the first step of the Hopcroft algorithm]

∴ Every state is reachable from the initial state P_0.

$$\hat{\delta}^*(P_0, w_i) = P_i \quad \forall\, i = 0, 1, 2, \ldots, m \longrightarrow (2)$$

Let $w_0, w_1, w_2, \ldots w_m$ be the respective words to reach P_i from P_0 for $i = 0, 1, 2, \ldots, m$

$$Q_1 = \{q_0, q_1, \ldots, q_n\} \quad [n < m]$$

When $m+1$ words are given as input for M_1 with $n+1$ states where $n < m$, according to pigeon hole principle, at least two words will lead to the same state in Q_1.

Let w_k and w_l be the two strings that end up in the same state.

i.e. $\delta_1^*(q_0, w_k) = \delta_1^*(q_0, w_l) \longrightarrow (3)$

We know,

$$\hat{\delta}^*(P_0, w_k) = P_k$$

$$\hat{\delta}^*(P_0, w_l) = P_l$$

P_k and P_l are distinguishable (by Hopcroft algorithm)

∴ There exists a string x such that,

$$\hat{\delta}^*(P_k, x) \in \hat{F} \text{ and } \hat{\delta}^*(P_l, x) \notin \hat{F} \text{ [or vice versa]}$$

$$\hat{\delta}^*(P_0, w_k x) = \hat{\delta}^*(P_k, x) \in \hat{F}$$

$$\hat{\delta}^*(P_0, w_l x) = \hat{\delta}^*(P_l, x) \notin \hat{F}$$

Or vice versa

$$\delta_1^*(q_0, w_k x) = \delta_1^*(\delta_1^*(q_0, w_k), x)$$

$$= \delta_1^*(\delta_1^*(q_0, w_\ell), x) \quad [\text{From } ③]$$

$$= \delta_1^*(\delta_1^*, q_0, w_\ell x)$$

If $\delta_1^*(q_0, w_k x)$ goes to a final state then $\delta_1^*(q_0, w_\ell x)$ will also go to a final state because they are equal.

Or similarly for a non-final state

$\therefore \hat{\delta}^*(p_0, w_k x) \in \hat{F}$ and $\hat{\delta}^*(p_0, w_\ell x) \notin \hat{F}$

but, $\delta_1^*(q_0, w_k x) \in F_1$ and $\delta_1^*(q_0, w_\ell x) \in \hat{F}_1$

This is a contradiction to the assumption,

$$L(\hat{M}) = L(M_1)$$

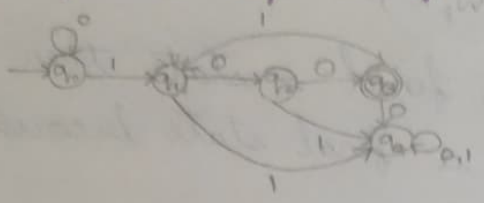
$\therefore |Q_1| \neq |\hat{Q}|$

$\therefore \hat{M}$ is the minimal DFA

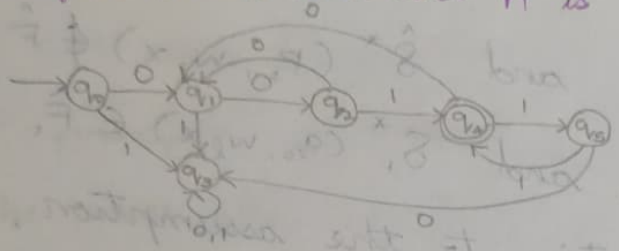
Either

Construct a DFA for the following:

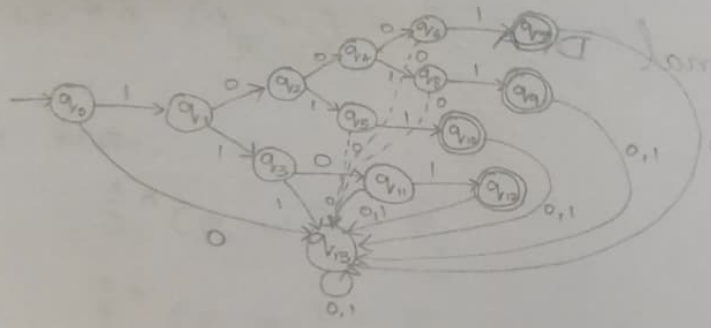
i) L1 : Non-empty strings over {0,1} in which every 1 is immediately followed by exactly two 0s. [≤6 states]



L2: Strings over {0,1} consisting of m consecutive 0s followed by n consecutive 1s where m and n are both positive integers, m is even and n is odd. [≤ 6 states]



L3: Strings over {0,1} that are the binary representation of all prime numbers between 10 and 20 with no leading 0s. [≤ 12 states]



11 → 1011
13 → 1101
17 → 10001
19 → 10011

Minimizing the DFA,

$\{q_0, q_1, q_2, q_4, \underline{q_6}, \underline{q_8}, \underline{q_5}, q_3, \underline{q_{11}}, q_{13}\}$ $\{q_7, q_9, q_{10}, q_{12}\}$

0 ↙

$\{q_{13}, q_2, q_4, q_6, q_{13}, q_{13}, q_{13}, q_{11}, q_{13}, q_{13}\}$

1 ↓

$\{q_1, q_3, q_5, q_8, \underline{q_7}, \underline{q_9}, \underline{q_{10}}, q_{13}, \underline{q_{12}}, q_{13}\}$

$\{q_0, q_1, q_2, \underline{q_4}, \underline{q_3}, q_{13}\}$ $\{q_6, q_8, q_5, q_{11}\}$ $\{q_7, q_9, q_{10}, q_{12}\}$

0 ↙ 1 ↘

$\{q_{13}, q_2, q_4, \underline{q_6}, \underline{q_{11}}, q_{13}\}$ $\{q_1, q_3, q_5, q_8, q_{13}, q_{13}\}$

$\{q_0, q_1, \underline{q_2}, q_{13}\}$ $\{q_3, q_4\}$ $\{q_5, q_6, q_8, q_{11}\}$ $\{q_7, q_9, q_{10}, q_{12}\}$

0 ↙

$\{q_{13}, q_2, \underline{q_4}, q_{13}\}$

$\{q_0, q_1, q_{13}\}\ \{q_2\}\ \{q_3, q_4\}\ \{q_5, q_6, q_8, q_{11}\}\ \{q_7, q_9, q_{10}, q_{12}\}$

0 ↙

$\{q_{13}, q_2, q_{13}\}$

$\{q_0, q_{13}\}\ \{q_1\}$

0 ↙ ↘

$\{q_{13}, q_{13}\}\ \{q_1, q_{13}\}$

$\{q_0\}\{q_{13}\}\{q_1\}\{q_2\}\{q_3, q_4\}$

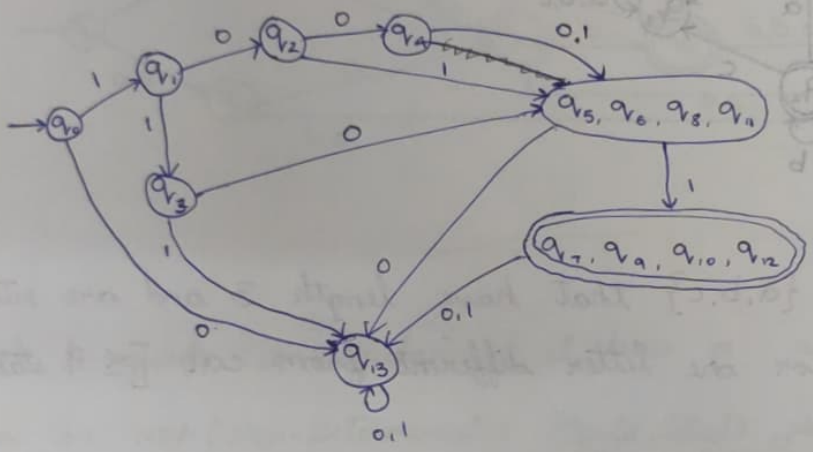
a ↙ ↘

$\{q_{11}, q_6\}\ \{q_{13}, q_8\}$

$\{q_0\}\{q_1\}\{q_2\}\{q_{13}\}\{q_3\}\{q_4\}\{q_5, q_6, q_8, q_{11}\}\{q_7, q_9, q_{10}, q_{12}\}$

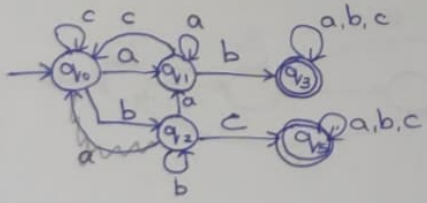
0 ↙ ↘ 0,1 ↓

$\{q_{13}, q_{13}, q_{13}, q_{13}\}\ \{q_{10}, q_7, q_9, q_{12}\}\ \{q_{13}, q_{13}, q_{13}, q_{13}\}$



LA: Strings over {a,b,c} that contain no occurrences of the following substrings : ab, bc [≤ 4 states]

DFA for strings with occurrences of ab or bc



Simplifying.

a ↙ b ↓ ↘ $\{q_0, q_1, q_2\}\ \{q_3, q_5\}$

$\{q_1, q_1, q_1\}\{q_{12}, q_{13}, q_5\}$

$\{q_0\}\{q_1, q_2\}\ \{q_3, q_5\}$

a ↙ b ↓ ↘

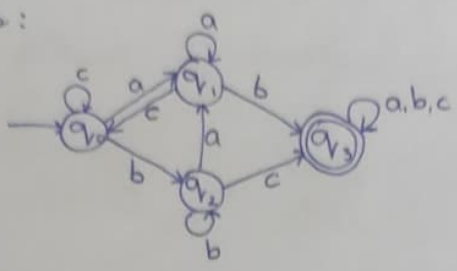
$\{q_1, q_1\}\{q_3, q_2\}$

$\{q_0\}\ \{q_1\}\{q_2\}\ \{q_3, q_5\}$ c →

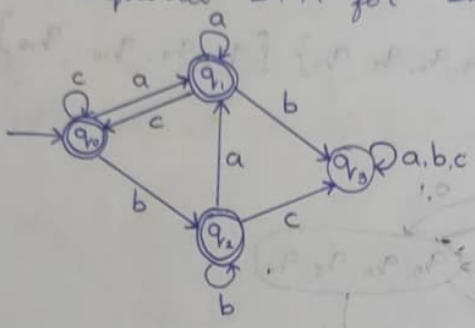
a ↙ b ↓

$\{q_3, q_5\}\ \{q_3, q_5\}\ \{q_3, q_5\}$

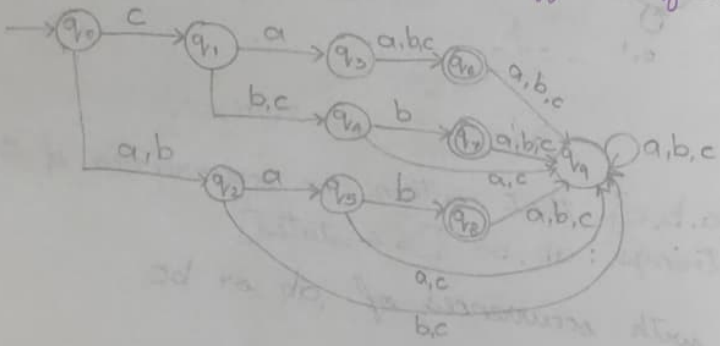
The minimized DFA for strings with occurrences of ab or b are is:



∴ The required DFA for L4 is:



L5: Strings over {a,b,c} that have length 3 and are either exactly cab or one letter different from cab [≤ 9 states]



Minimizing,

$\{q_0, q_1, q_2, q_3, q_4, q_5, q_9\}$ $\{q_6, q_7, q_8\}$

$\{q_2, q_3, q_5, q_8, q_7, q_9, q_9\}$

$\{q_3\}$ $\{q_0, q_1, q_2, q_4, q_5, q_9\}$ $\{q_6, q_7, q_8\}$

$\{q_2, q_5, q_8, q_9, q_7, q_9\}$

$\{q_1\}\{q_3\}$ $\{q_0, q_2, q_4, q_5, q_9\}$ $\{q_6, q_7, q_8\}$

$\{q_2, q_5, q_9, q_9, q_9\}$ $\{q_2, q_7, q_7, q_8, q_9\}$

$\{q_1\}\{q_3\}$ ~~$\{q_4\}$~~ $\{q_0, q_4, q_5\}$ $\{q_0, q_2, q_9\}$ $\{q_6, q_7, q_8\}$

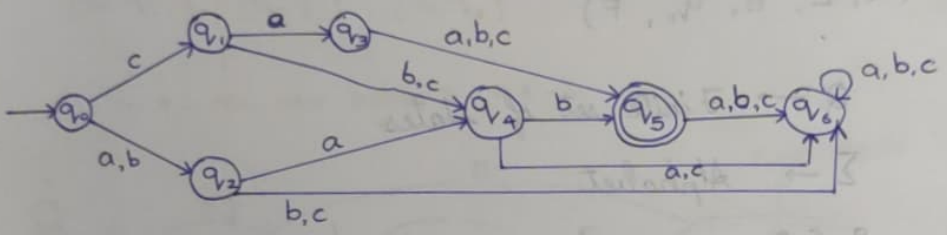
$\{q_9, q_9\}$ $\{q_7, q_8\}$ $\{q_9, q_9\}$

$\{q_2, q_5, q_9\}$

$\{q_1\}\{q_2\}\{q_3\}\{q_4, q_5\}\{q_0, q_6\}\{q_7, q_8\}$

$\{q_0\}\{q_1\}\{q_2\}\{q_3\}\{q_4, q_5\}\{q_7, q_8\}\{q_6\}$

∴ The minimized DFA is:



*. A language is regular if it has a regular expression.

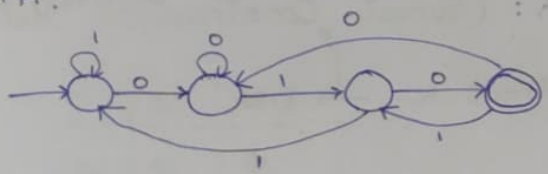
DFA or NFA (Non-Deterministic Finite State Automaton)

Eg:

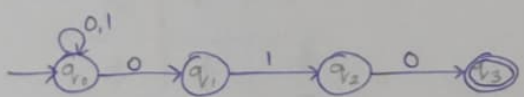
Language of strings ending with 010

i) Regular Expression ⟶ $(0+1)^*010$

ii) DFA:



iii) NFA:



$\delta^*(q_0, 11010) = \{q_0, q_1, q_3\}$

Final state $(q_3) \in \delta^*(q_0, 11010)$

$\Rightarrow$ 11010 is a valid string

*. For a state q_i in an NFA,

$\delta(q_i, a) = \phi$ Input a does nothing

(or)

$\{P\}$ goes to a single state

(or)

$\{P_1, P_2, \ldots, P_m\}$ goes to one of multiple possible states

*. A string w is accepted by an NFA if $\delta^*(q_0, w) \cap F \neq \phi$

*. ∴ The language of an NFA, M, is given by,

$$L(M) = \{w \mid \delta^*(q_0, w) \cap F \neq \phi\}$$

Non-Deterministic Finite State Automaton (NFA)

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

$Q \rightarrow$ Finite set of states

$\Sigma \rightarrow$ Alphabet

$q_0 \in Q \rightarrow$ Initial state

$F \subseteq Q \rightarrow$ Set of final states

$\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ → Power set of Q (Some combination the states in Q)

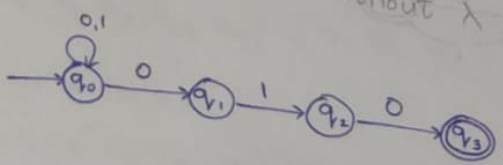
Possible to move from one state to another even if there is no input

*. Every DFA is an NFA

Converting an NFA to DFA: (Subset Construction Method)

NFA with λ → NFA without λ → DFA

Eg:



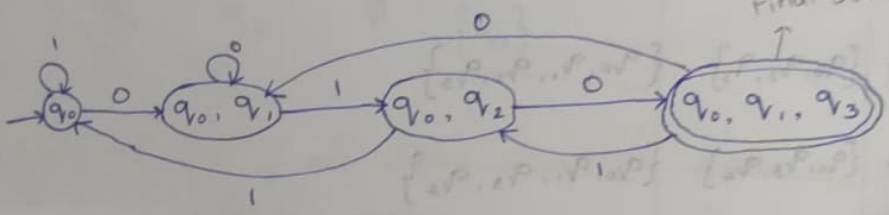
	0	1
q_0	q_0, q_1	q_0
q_1	ϕ	q_2
q_2	q_3	ϕ
q_3	ϕ	ϕ

Subset construction:

	0	1
Initial state → $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

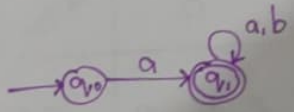
-x- No new subsets -x-

Final state ∴ $q_3 \in \{q_0, q_1, q_3\}$



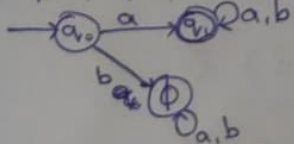
* Subset construction does not always give the minimized DFA

Q



Convert to DFA

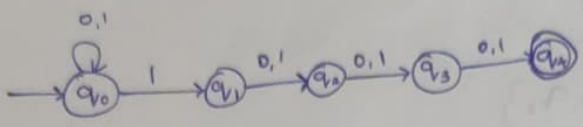
	a	b
q_0	q_1	ϕ
q_1	q_1	q_1
ϕ	ϕ	ϕ



Q Construct an NFA for

from right is $1 \Rightarrow 2^4$ states in the DFA (Cannot be minimized)

Regular Expression → $(0+1)^*.1.(0+1).(0+1).(0+1)$

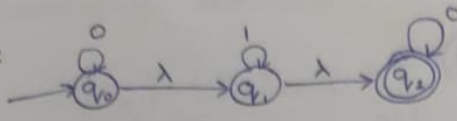


	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_1, q_3, q_4\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3, q_4\}$	$\{q_0, q_1, q_2, q_3, q_4\}$
$\{q_0, q_3, q_4\}$		
$\{q_0, q_1, q_3, q_4\}$		
$\{q_0, q_1, q_3\}$		
$\{q_0, q_3\}$		
$\{q_0, q_2, q_3, q_4\}$		
$\{q_0, q_1, q_2, q_3, q_4\}$		

→ λNFA or ∈NFA

NFA with lambda (epsilon):

Eg:



$0^* 1^* 0^*$

Eliminating the λ,

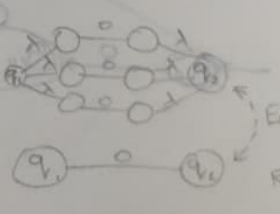
λNFA	0	1	$\overline{\lambda}$ (closure)
q_0	q_0	ϕ	$\{q_0, q_1, q_2\}$
q_1	ϕ	q_1	$\{q_1, q_2\}$
q_2	q_2	ϕ	$\{q_2\}$

*. $\overline{\lambda}(q_i)$ → Set of all states reachable from q_i only by λ (or ∈) moves
or
$\overline{\in}(q_i)$

*. $q_i \in \overline{\lambda}(q_i)$ always

NFA	0	1
Initial State → q_0 *	q_0, q_1, q_2	q_1, q_2
Final State q_1	q_2	q_1, q_2
Final State * q_2	q_2	ϕ

For initial state,
if $\overline{\lambda}(q_0) \cap F \neq \phi$
then $q_0 \in F$



$\delta'(q_0, 0) = \overline{\lambda}(\delta(\overline{\lambda}(q_0), 0))$

$= \overline{\lambda}(\delta(\{q_0, q_1, q_2\}, 0))$

$= \overline{\lambda}(q_0, q_2)$

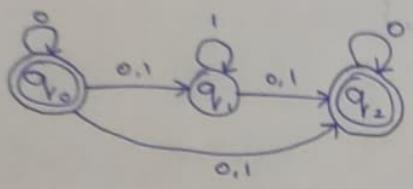
$= \overline{\lambda}(q_0) \cup \overline{\lambda}(q_2)$

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

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

$$\delta'(q_0, 1) = \bar{\lambda}(\delta(\bar{\lambda}(q_0), 1))$$
$$= \bar{\lambda}(\delta(\{q_0, q_1, q_2\}, 1))$$
$$= \bar{\lambda}(q_1)$$
$$= \{q_1, q_2\}^*$$

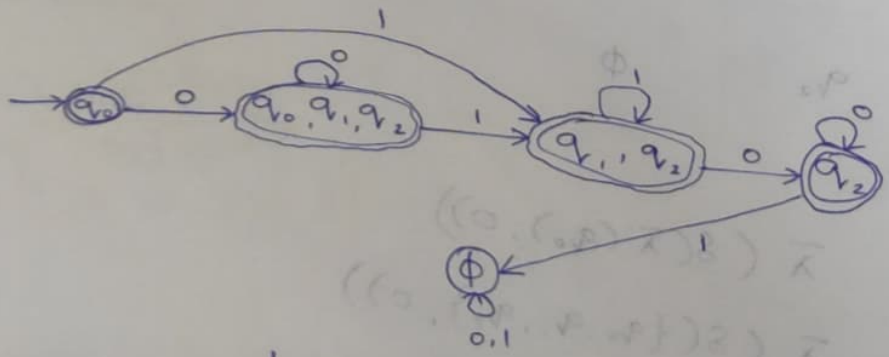
Similarly complete the NFA table

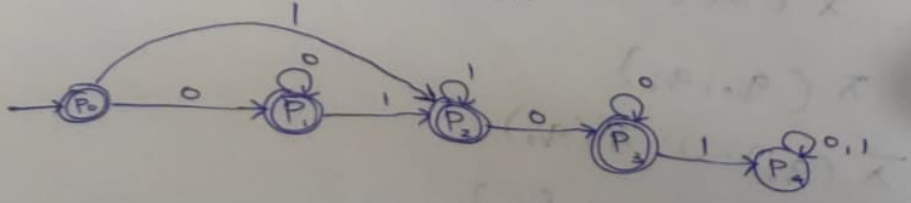


Converting to DFA,

	0	1
* $\{q_0\}$ ($\in$F)	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
* $\{q_0, q_1, q_2\}$ ($\in$F)	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
* $\{q_1, q_2\}$ ($\in$F)	$\{q_2\}$	$\{q_1, q_2\}$
* $\{q_2\}$ ($\in$F)	$\{q_2\}$	$\{\phi$
ϕ	ϕ	ϕ

Final state





Minimizing the DFA.

$\{P_4\}$ $\{P_0, P_1, P_2, \underline{P_3}\}$

0 ↙ ↘ 1

$\{P_1, P_1, P_3, P_3\}$ $\{P_2, P_2, P_2, \underline{P_4}\}$

$\{P_4\}\{P_3\}$ $\{P_0, P_1, \underline{P_2}\}$

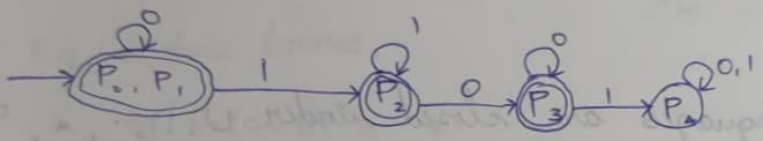
0 ↙

$\{P_1, P_1, \underline{P_3}\}$

$\{P_4\}\{P_3\}\{P_2\}$ $\{P_0, P_1\}$

0 ↙ ↘

$\{P_1, P_1\}$ $\{P_2, P_2\}$

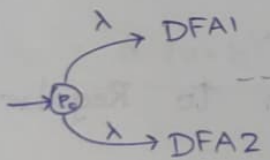
∴ 

$0^*1^*0^*$

---

*. If L_1 and L_2 are regular languages then $L_1 \cup L_2$ is regular.

L_1 → DFA1 , L_2 → DFA2



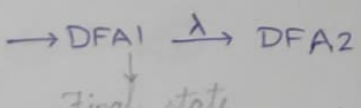
---→ This can be converted to a DFA

⇓

$L_1 \cup L_2$ is regular

*. If L_1 and L_2 are regular, $L_1 \cdot L_2$ is also regular

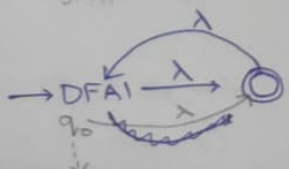
L_1 → DFA1 , L_2 → DFA2



Final state becomes non-final

*. If L is regular, L^* is also regular

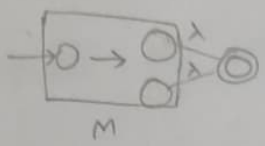
L → DFA1



Initial State of DFA1

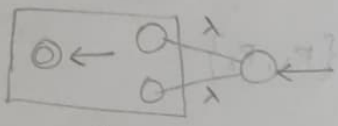
* If L is regular, L^R is also regular

i) Interchange the arrow direction
ii) Swap final state to initial state
iii) Swap initial state to final state



M

Reversal:



M^R

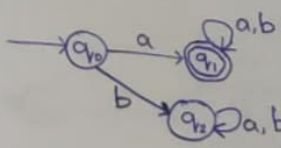
* ∴ Regular languages are closed under ∪, ∩, ·, *, ^{c} and

* $L_1 - L_2 = L_1 \cap L_2^c$

If L_1 and L_2 are regular, then $L_1 - L_2$ is also regu

Converting a Finite State Automaton to Regular Expression:

a) Using Arden's Lemma:



$q_0 = aq_1 + bq_2$ (Or)

Final State ← * $q_1 = aq_1 + bq_1 + \lambda$

$q_2 = aq_2 + bq_2$

Arden's Lemma:

If $X = AX + B$ then $X = A^*B \quad (A \neq \lambda)$

Solving for initial state (q_0),

$$q_2 = (a+b)q_2 + \phi$$

By Arden's lemma,

$$q_2 = (a+b)^* \cdot \phi$$

$$\therefore q_2 = \phi \quad [\because r \cdot \phi = \phi]$$

→ If q_2 is initial state, it will accept nothing [ϕ]

$$q_1 = (a+b)q_1 + \lambda$$

By Arden's lemma,

$$q_1 = (a+b)^* \cdot \lambda$$

$$\therefore q_1 = (a+b)^* \quad [\because r \cdot \lambda = r]$$

→ If q_1 is initial state, it will accept everything [$(a+b)^*$]

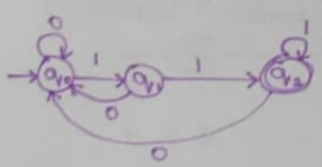
$$q_0 = aq_1 + bq_2$$

$$= a \cdot (a+b)^* + b \cdot \phi$$

$\therefore q_0 = a \cdot (a+b)^*$ → with q_0 as initial state, the FA will accept $a \cdot (a+b)^*$

② Ending with 11

DFA



$$q_0 = 0 \cdot q_0 + 1 \cdot q_1$$

$$q_1 = 0 \cdot q_0 + 1 \cdot q_2$$

$$q_2 = 0 \cdot q_0 + 1 \cdot q_2 + \lambda$$

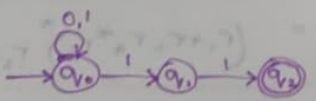
Solving,

$$q_2 = 1 \cdot q_2 + 0q_0 + \lambda$$

$$= 1^* \cdot (0q_0 + \lambda)$$

(PTO)

NFA



$$q_0 = 0q_0 + 1q_0 + 1q_1$$

$$q_1 = 1q_2$$

$$q_2 = \phi + \lambda = \lambda$$

Solving,

$$q_1 = 1 \cdot q_2 = 1 \cdot \lambda = 1$$

$$q_0 = (0+1)q_0 + 1 \cdot q_1 = (0+1)q_0 + 11$$

$$q_0 = (0+1)^* \cdot 11 \quad [\text{Arden's lemma}]$$

DFA

$q_1 = 0 \cdot q_0 + 11^*(0q_0 + \lambda)$

$q_1 = (0 + 11^*0)q_0 + 11^*\lambda$

$\therefore q_1 = (0 + 11^*0)q_0 + 11^*$

$q_0 = 0 \cdot q_0 + 1 \cdot q_1$

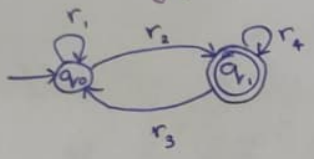
$= 0 \cdot q_0 + 1[(0+11^*0)q_0 + 11^*]$

$= q_0 + 1(0+11^*0)q_0 + 111^*$

$= (0 + 10 + 111^*0)q_0 + 111^*$

$q_0 = (0 + 10 + 111^*0)^* 111^*$

Generalising,



R $q_0 = r_1 q_0 + r_2 q_1$

$q_1 = r_3 q_0 + r_4 q_1 + \lambda = r_4^*(r_3 q_0 + \lambda)$ [Arden's theorem

$\therefore q_0 = r_1 q_0 + r_2 r_4^*(r_3 q_0 + \lambda)$

$= r_1 q_0 + r_2 r_4^* r_3 q_0 + r_2 r_4^*$

$= (r_1 + r_2 r_4^* r_3) q_0 + r_2 r_4^*$

$\therefore q_0 = (r_1 + r_2 r_4^* r_3)^* r_2 r_4^*$ [Arden's Lemma]
(R)

If we start with q_0,

$q_0 = r_1^* r_2 q_1$

$q_1 = r_3 r_1^* r_2 q_1 + r_4 q_1 + \lambda$

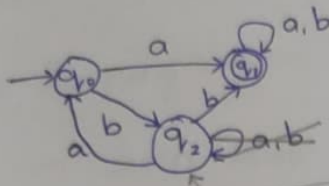
$= (r_3 r_1^* r_2 + r_4) q_1 + \lambda$

$q_1 = (r_3 r_1^* r_2 + r_4)^*$

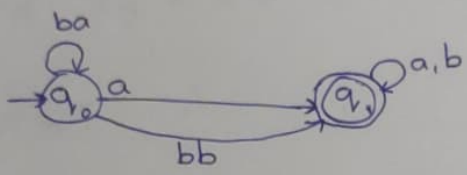
$q_0 = r_1^* r_2 (r_3 r_1^* r_2 + r_4)^*$
(R)

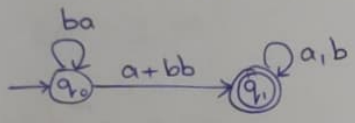
. Similarly for  , the regular expression R = r

b) State Elimination Method:



Choose a non-final and non-initial state and compress





→ Generalised form

∴ Using the generalised formula,

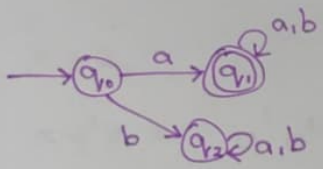
$$R = r_1^* r_2 (r_3 r_1^* r_2 + r_4)^*$$

ϕ [∵ $r_3 = \phi$]

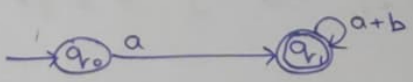
$$\therefore R = (ba)^* (a+bb)[(a+b)]^*$$

$r_1 = ba$
$r_2 = a+bb$
$r_3 = \phi$
$r_4 = a+b$

(Q)



Compressing q_{r_2},



$r_1 = \phi$

$r_2 = a$

$r_3 = \phi$

$r_4 = a+b$

$$R = r_1^* r_2 (r_3 r_1^* r_2 + r_4)^*$$

$$= \phi^* a (a+b)^* = a(a+b)^*$$

$\phi^* = \lambda$

Pumping Lemma

(If L is an infinite regular language, then there exists an integer m such that any string $w \in L$ with $|w| \geq m$ can be decomposed into three parts $w = xyz$, such that

$$|xy| \leq m$$

$$|y| \geq 1$$

$$xy^k z \in L \quad \forall \; k \geq 0)$$

Taking the contrapositive, $P \rightarrow q \equiv \neg q \rightarrow \neg P$

(8) Show that $\{a^n b^n : n \geq 0\}$ is not regular

$\{a^n b^n : n \geq 0\}$ is not regular

Suppose $L = \{a^n b^n : n \geq 0\}$ is a regular language.

By Pumping lemma, there exists ~~a decomposition~~ $w = xyz$ such ~~m~~ such that any string $w \in L$ with $|w| \geq m$, there exists a decomposition $w = xyz$ such that $|xy| \leq m$, $|y| \geq 1$ and $xy^k z \in L \quad \forall k \geq 0$

$w_m = a^m b^m$

$|w| = 2m \geq m$

Let $a^m b^m = xyz$

Since $|xy| \leq m$,

$x = a^i$, $y = a^j$ $\Big\}$ $i + j \leq m$

$\therefore z = a^{m-i-j} b^m$

$xy^k z \in L \quad \forall k \geq 0$

Let $k = 2$,

$xy^2z = a^i a^{2j} a^{m-i-j} b^m$

$= a^{i+2j+m-i-j} b^m = a^{j+m} b^m$

$xy^2z \in L \Rightarrow j + m = m \Rightarrow j = 0 \Rightarrow |y| = 0$

But, $|y| \geq 1 \rightarrow$ Contradiction

$\therefore L$ is not regular

Q Show that $L = \{w : |w|_a = |w|_b\}$ is not regular

$w = a^m b^m$

$|w| = 2m \geq m$

$a^m b^m = xyz$

Since $|xy| \leq m$,

$x = a^i, y = a^j$

$z = a^{m-i-j} b^m$

$xy^k z \in L \quad \forall k \geq 0$

Let $k = 2$,

$xy^2 z \in \notin L$ (See previous problem)

$\Downarrow$

$|y| = 0 \rightarrow$ Contradiction

Q Show that $L = \{ww^R\}$ is not regular

Take $w = a^m b^m a^m$

Q $\Sigma = \{0\}$

$L = \{0^{n^2} : n \geq 0\}$ $L = \{\lambda, 0, 0^4, 0^9, 0^{16}, \ldots\}$

Show that L is not regular

Let $w = 0^{m^2}$

$|w| = m^2 \geq m$

Let $0^{m^2} = xyz$

Since $|xy| \leq m$,

$x = 0^i, y = 0^j$

$z = 0^{m^2-i-j}$ $i+j \leq m$

$xy^k z \in L \quad \forall k \geq 0$

Let $k = 2$,

$xy^2 z = 0^i 0^{2j} 0^{m^2-i-j} = 0^{i+2j+m^2-i-j}$

$= 0^{m^2+j}$

$m^2 < m^2 + j \quad [\because j > 0] \; [\because |y| \geq 1] \longrightarrow ①$

$|y| \leq |xy| \leq m$

$j \leq m < 2m + 1$

$m^2 + j < m^2 + 2m + 1$

$m^2 + j < (m+1)^2 \longrightarrow ②$

Combining ① and ②,

$m^2 < m^2 + j < (m+1)^2$

m^2 and $(m+1)^2$ are consequtive square numbers

$\Downarrow$

$m^2 + j$ cannot be a square number

$\therefore 0^{m^2+j} \notin L$

$\therefore L$ is not regular

---

② $\Sigma = \{0\}$

Show that

$L_1 = \{0^{n!} : n \geq 0\}$

$L_2 = \{0^p : p \text{ is prime}\}$

$L_1 = \{0^{n!} : n \geq 0\}$

Let $w = 0^{m!}$

$|w| = m! \geq m$

Let $0^{m!} = xyz$

Since $|xy| \leq m$, $i + j \leq m$

$x = 0^i \; ; \; y = 0^j$

$z = 0^{m!-i-j}$

$xy^kz \in L \quad \forall \; k \geq 0$

Let $k = 2$,

$xy^2z = 0^i 0^{2j} 0^{m!-i-j} = 0^{m!+j} \in L$

$m! < m! + j$ $[\because j > 0]$ $[\because |y| \geq 1]$

$|y| \leq |xy| \leq m$

$j \leq m < mm!$

$m! + j < m! + mm!$

$m! < m! + j < m!(1+m)$

$m! < m! + j < (m+1)!$

$m!$ and $(m+1)!$ are consecutive factorial numbers, therefore $m! + j$ cannot be a factorial number

$\therefore 0^{m!+j} \notin L$

$\therefore L_1$ is not regular

$L_2 = \{0^p : p$ is a prime number$\}$

Let $w = 0^p$ where $p > m$

Let $0^p = xyz$

$x = 0^i$; $y = 0^j$ $\quad i + j \leq m$

$z = 0^{p-i-j}$

$xy^kz \in L \quad \forall\ k \geq 0$

$\therefore xz \in L \Rightarrow |x| + |z|$ is a prime number

$0^i 0^{p-i-j} = 0^{p-j} \Rightarrow p - j$ is a prime number

Let $q_1 = |x| + |z|$

Consider $xy^{q_1}z \in L$

$|xy^{q_1}z|$ is prime since $xy^{q_1}z \in L$

$\therefore |x| + |y^{q_1}| + |z|$ is prime

$\Rightarrow q_1 + |y^{q_1}|$ is a prime

$\Rightarrow q + q|y|$ is prime

$\Rightarrow q(1+|y|)$ is prime

$q(1+|y|)$ is clearly not prime ~~has~~ as it has the factors q and $1+|y|$. NOTE: $1+|y| \neq 1$ $[\because |y| \geq 1]$

$\therefore |xy^q z| = q(1+|y|)$ is not prime

$\therefore xy^q z \notin L$

$\therefore L_2$ is not regular

(Q) ~~r =~~ $L = \{a^n b^m a^{n+m}\}$;

$a^n b^m a^{n+m}$ is not regular → The values of n and m cannot be remembered

$L = \{a^n b^m a^{n+m}\}$

Let $w = a^n b^m a^{n+m}$ $\quad [|w| = 2(n+m) \geq m]$

Let $a^n b^m a^{n+1} = xyz$

$x = a^i b^j$; $y = b^k a^{\cancel{n+}1}$

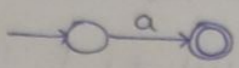
(Q) $|w|_a > |w|_b$

Take $w = b^m a^{2m}$ $\quad [a^m b^{m-1}$ will not work]

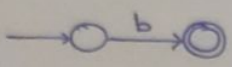
Regular Expression to Finite State Automaton:

$(a+b^*)\ ab^*\ (a+b)^*$

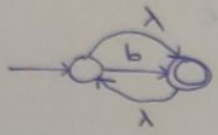
a



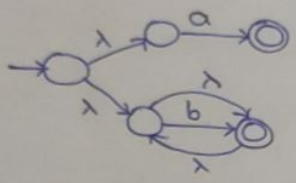
b



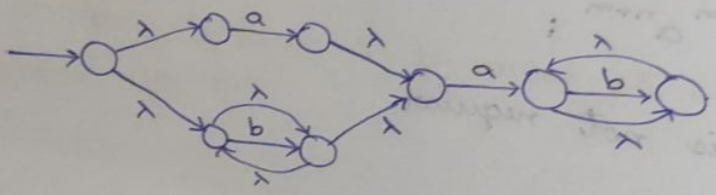
b^*



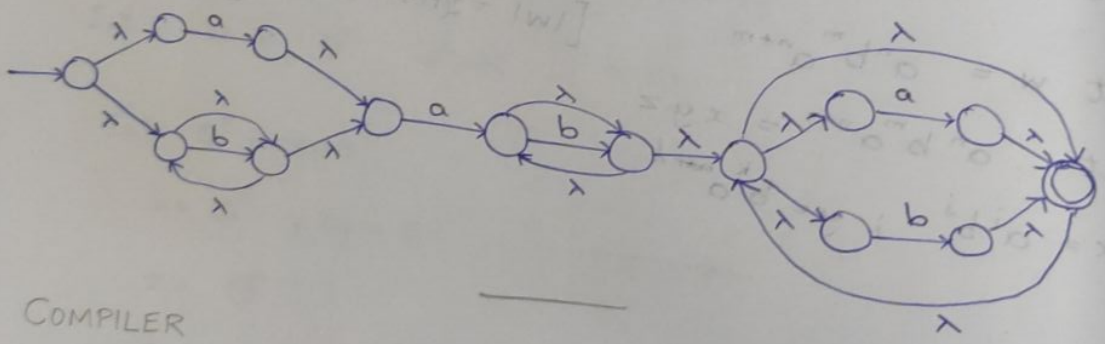
$(a+b^*)$



$(a+b^*)\cdot a\cdot b^*$



$(a+b^*)\ ab^*\ (a+b)^*$



COMPILER

Preprocessing → Lexical Analysis → Syntax Analysis → Semantic Analysis

↓

Intermediate C

↓

Code Optimizat

Q Given two regular languages L_1, L_2, how to check if they are the same?

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \phi$$

Q $L_1 = \{a^n b^n : n \neq 100\}$

Show L_1 is not regular

If $L_1 \cap L_2 = L_3$, L_1 and L_3 are regular, L_2 may or may not be regular

Let $L_2 = \{a^{100} b^{100}\}$

$L_3 = \{a^n b^n : n \geq 0\}$

$L_3 = L_1 \cup L_2$

We know L_2 is regular [It is finite]

$r_2 = aaa\overset{100}{\ldots}a\,bbb\overset{100}{\ldots}b$

We know L_3 is not regular.

If L_1 is regular, $L_1 \cup L_2$ is also regular

$\Downarrow$

L_3 is regular

$\Downarrow$

Contradiction

$\therefore L_1$ is not regular

Q Is $\{a^n b^n : n \geq 0\} \cup \{a^n b^m : n, m \geq 2\}$ regular?

$\{a^n b^n : n \geq 0\} \subseteq \{a^n b^m : n, m \geq 0\}$

Non-Regular Language is subset of Regular Language

$\{a^n b^m : n, m \geq 0\}$ is regular

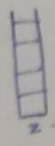
$\therefore \{a^n b^m : n, m \geq 2\}$ is regular

$\{a^n b^m : n, m \geq 2\} \cup \{\lambda, ab\}$ is also regular

Regular

$\{\lambda, ab\}$: Regular (Finite)

PUSHDOWN AUTOMATON (PDA)



→ Indicates empty stack

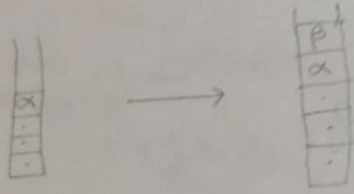
→ Memory

Context Free Language

Regular Languages ⊆ Context Free Languages

$\delta(q_0, a, \alpha) = (q_1, \beta\alpha)$

- α: Top of the stack → Popped
- β: Pushing



$\delta(q_0, a, \alpha) = (q_1, \lambda)$

- α: Pops α
- λ: Pushes nothing
- Popping α

*. Multiple values can be pushed at once, but only one value can be popped at a time.

Definition:

A pushdown automaton PDA is a septuple,

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

- $Q \rightarrow$ Set of states
- $\Sigma \rightarrow$ Alphabet
- $\Gamma \rightarrow$ Stack alphabet
- $q_0 \in Q \rightarrow$ Initial state
- $Z \in \Gamma \rightarrow$ Denotes empty stack
- $F \in Q \rightarrow$ Set of final states
- $\delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

Eg: $\Sigma \subseteq \Gamma = \Sigma \cup \{\}$

Push Operation (Pushing β):

$$\delta(q_i, a, \alpha) = (q_j, \beta\alpha) \quad \text{where } \beta \in \Gamma^*$$

Pop Operation (Popping α):

$$\delta(q_i, a, \alpha) = (q_j, \lambda)$$

No Push No Pop:

$$\delta(q_i, a, \alpha) = (q_j, \alpha)$$

Example:

$w = aabb$

$\delta(q_0, a, z) = (q_1, az) \rightarrow ①$

$\delta(q_1, a, a) = (q_1, aa) \rightarrow ②$

$\delta(q_1, b, a) = (q_2, \lambda) \rightarrow ③$

$\delta(q_2, b, \overset{a}{\hat{b}}) = (q_2, \lambda) \rightarrow ④$

$\delta(q_2, \lambda, z) = (q_f, z) \rightarrow ⑤$

~~q_0~~

$(q_0, \underline{a}abb, z) \xrightarrow{①} (q_1, \underline{a}bb, \underline{a}z) \xrightarrow{②} (q_1, \underline{b}b, aaz)$

Top of the stack

Next letter to be read

(Current State, Word, Stack)

$\downarrow ③$

$(q_2, \underline{b}, \underline{a}z)$

$\downarrow ④$

$(q_2, \underline{\lambda}, \underline{z})$

$\downarrow ⑤$

(q_f, z)

Reached final state

Stack is empty

Valid String

Q

$L = \{a^n b^n : n \geq 1\}$ → Deterministic PDA

PDA Rules:

$\delta(q_0, a, z) = (q_1, 1z)$ → Pushes 1 to the stack when q_0 sees 'a'

$\delta(q_1, a, 1) = (q_1, 11)$ → Pushes 1 to the stack when q_1 sees 'a'

$\delta(q_1, b, 1) = (q_2, \epsilon)$ → Pops 1 when q_1 receives b

$\delta(q_2, b, 1) = (q_2, \epsilon)$

$\delta(q_2, \lambda, z) = (q_f, z)$

↳ Reached the end of the string

Q $L = \{a^n b^n : n \geq 0\}$ → Non-Deterministic PDA

$\delta(q_0, \lambda, z) = (q_f, z)$

Followed by PDA rules for Q $L = \{a^n b^n : n \geq 1\}$

Q $L = \{a^n b^{2n} : n \geq 1\}$

$\delta(q_0, a, z) = (q_1, 11z)$

$\delta(q_1, a, 1) = (q_1, 111)$

} Pushing two 1s to the stack for each 'a' received by q_0 and q_1

$\delta(q_1, b, 1) = (q_2, \epsilon)$

$\delta(q_2, b, 1) = (q_2, \epsilon)$

$\delta(q_2, \lambda, z) = (q_f, z)$

Q $L = \{a^{2n} b^n : n \geq 1\}$

$\delta(q_0, a, z) = (q_1, z)$

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

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

$\delta(q_1, a, 1) = (q_0, 11)$

} 1 is pushed only for even a's

(Pop 1 from the stack for each b)

Q $|w|_a = |w|_b$ → Deterministic PDA

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

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

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

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

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

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

Q Palindromes

Non-Deterministic Push down Automaton

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

$Q \rightarrow$ Set of states

$\Sigma \rightarrow$ Alphabet

$\Gamma \rightarrow$ Stack Alphabets

$q_0 \in Q \rightarrow$ Initial state

$F \subseteq Q \rightarrow$ Set of final states

$z_0 \in \Gamma \rightarrow$ Empty stack

$\delta \rightarrow Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$

Q

Construct PDA for $\{w\#w^R : w \in \{a,b\}\}$ and give instantaneous description for ab # ba and ab#ab

PDA Rules: → Deterministic

$$\delta(q_0, a, z) = (q_0, 0z)$$
$$\delta(q_0, b, z) = (q_0, 1z)$$
$$\delta(q_0, a, 0) = (q_0, 00)$$
$$\delta(q_0, b, 0) = (q_0, 10)$$
$$\delta(q_0, a, 1) = (q_0, 01)$$
$$\delta(q_0, b, 1) = (q_0, 11)$$

$$\left.\begin{aligned}\delta(q_0, a, z|0|1) &= (q_{0}, \\ \delta(q_0, b, z|0|1) &= (q_{0},\end{aligned}\right\}$$

Non-Deter

For WW^R,

$$\delta_*(q_0, \#, z|0|1) = (q_1, z|0|1) \longrightarrow \begin{cases}\delta(q_0, a, 0) = (q_1 \\ \delta(q_0, b, 1) = (q_1\end{cases}$$

$$\delta(q_1, a, 0) = (q_1, \lambda)$$
$$\delta(q_1, b, 1) = (q_1, \lambda)$$
$$\delta(q_1, \lambda, z) = (q_f, z)$$

Non-Deterministic ⇒ Bra

$\delta(q_0, a, 0)$

(q_1, λ) $(q_0, 00)$

DETERMINISTIC PUSHDOWN AUTOMATA

NPDA : $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{(Q \times \Gamma^*)}$ < (Not allowed)

DPDA : $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to (Q \times \Gamma^*) \cup \phi$ > (Allowed)

Such that,

$\delta(q, a, x) \neq \phi$ implies $\delta(q, \lambda, x) = \phi$ and vice versa

* All NPDA cannot be converted to DPDA
* $L(DPDA) \subset L(NPDA)$
  ↳ Proper Subset

FORMAL GRAMMARS

<Sentence> → <Noun Phrase> <Verb Phrase>

<Noun Phrase> → <Article> <Noun>

(or)

<Article> <Adjective> <Noun>

<Verb Phrase> → <Verb>

(or)

<Verb> <Adverb>

<Noun> → dog | cat | door

<Verb> → barks | runs | locks

<Article> → a | an

<Adjective> → black | fat

<Adverb> → fastly | loudly

Eg: i) a dog runs fastly ✓

ii) cat runs fastly ✗ [No article]

Automata ↔ Accepts strings
RegEx → Represents strings

Definition:

A formal grammar (generates strings) is a quadruple $G = (V, T, S, P)$

where

V → Finite set of variables (Non-Terminals)

T → Finite set of terminals $[V \cap T = \phi]$

$S \in V$ → Start symbol

P → Set of production rules of the form $\alpha \rightarrow \beta$

Eg: Grammar:

i) $S \rightarrow aS$

ii) $S \rightarrow aA$

iii) $A \rightarrow bB$

iv) $B \rightarrow bB$

v) $B \rightarrow \lambda$

P

$V = \{S, A, B\}$; $T = \{a, b\}$ $S = S$

Does this grammar generate aab?

$S \overset{i}{\Rightarrow} aS \overset{ii}{\Rightarrow} aaA \overset{iii}{\Rightarrow} aabB \overset{v}{\Rightarrow} aab\lambda$

∴ It can generate aab

→ $a \cdot a^* \cdot b \cdot b^*$

Eg: 1 $S \xrightarrow{} aABb$

2 $A \to aA$

3 $A \to \lambda$

4 $B \to Bb$

5 $B \to \lambda$

$aa^*b^*b \equiv aa^*bb^*$

$V = \{S, A, B\}$

$T = \{a, b\}$

Can these production rules produce aabb?

$S \xrightarrow{1} aABb \xrightarrow{2} aaABb \xrightarrow{3} aaBb \xrightarrow{4} aaBbb \xrightarrow{5} aabb$ ✓

abab ? ⇒ NOT POSSIBLE

Eg: $S \to aSa \mid bSb \mid \lambda$ ⇒ All even palindromes

$S \to aSa \mid bSb \mid a \mid b$ ⇒ All odd palindromes

$S \to aSa \mid bSb \mid a \mid b \mid \lambda$ ⇒ All palindromes

*. A language L is context-free if there exists a PDA M or CFG G such that

(Context-Free Grammar)

$L(G) = L$

[or] $L(M) = L$

Types of Grammars:

Every grammar is a,

$\alpha \longrightarrow \beta$

$\alpha \in (V \cup T)^* . V . (V \cup T)^*$ (V: Variable, T: Terminal)

$\beta \in (V \cup T)^*$

Unconditional Grammar
(or)
Type-0 Grammar
(or)
Phase-Structure Grammar

$\alpha \longrightarrow \beta$

$\alpha \in (V \cup T)^* . V . (V \cup T)^*$

$\beta \in (V \cup T)^*$

$|\alpha| \leq |\beta|$ except for $S \longrightarrow \lambda$ $[|\alpha|=1 > |\beta|=0]$

Type-1 Grammar
(or)
Context-Sensitive Grammar

$\alpha \longrightarrow \beta$

$\alpha \in V \quad [|\alpha|=1]$

$\beta \in (V \cup T)^* \quad [|\beta| \geq 1]$

Type-2 Grammar
(or)
Context-Free Grammar

$\Rightarrow |\alpha| \leq |\beta|$

Type-2 $\subseteq$ Type-1 $\subseteq$ Type-0

$\alpha \longrightarrow \beta$

$\alpha \in V$

$\beta \in T^* \cup T^* . V . T^*$

Type-3 Grammar
(or)
Linear Grammar

$\alpha \longrightarrow \beta$

$\alpha \in V$

$\beta \in T^* \cup VT^*$

Left-Linear Grammar

$\alpha \longrightarrow \beta$

$\alpha \in V$

$\beta \in T^* \cup T^*V$

Right-Linear Grammar

*. A grammar is said to be regular if it is either right linear or left linear.

Left Linear $\longleftrightarrow$ Right Linear

Derivations in a Grammar:

Q

$S \rightarrow aABC$

$A \rightarrow aA \mid \lambda$

$B \rightarrow Bb \mid \lambda$

$C \rightarrow aC \mid \lambda$

Derive aabaa

Leftmost Derivation

$S \Rightarrow aABC$

$\Rightarrow aaABC$

$\Rightarrow aaBC$

$\Rightarrow aaBbC$

$\Rightarrow aabC$

$\Rightarrow aabaC$

$\Rightarrow aabaaC$

$\Rightarrow aabaa$

Rightmost Derivation

$S \Rightarrow aABC$

$\Rightarrow aABaC$

$\Rightarrow aABaaC$

$\Rightarrow aABaa$

$\Rightarrow aABbaa$

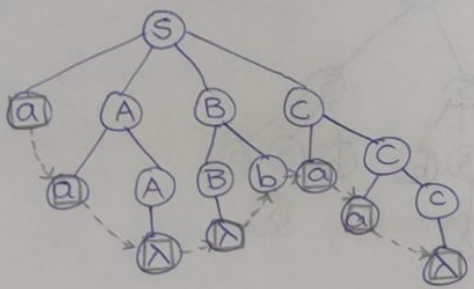
$\Rightarrow aAbaa$

$\Rightarrow aaAbaa$

$\Rightarrow aabaa$

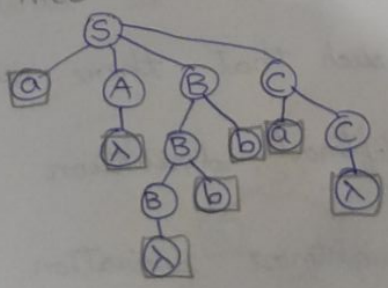
Parse Tree

(Applicable for CFG)



Q abba

Parse Tree



Leftmost Derivation

$S \Rightarrow aABC$

$\Rightarrow aBC$

$\Rightarrow aBbC$

$\Rightarrow aBbbC$

$\Rightarrow abbC$

$\Rightarrow abbaC$

$\Rightarrow abba$

Rightmost Derivation

$S \Rightarrow aABC$

$\Rightarrow aABaC$

$\Rightarrow aABa$

$\Rightarrow aABba$

$\Rightarrow aABbba$

$\Rightarrow aAbba$

$\Rightarrow abba$

Q

Grammar:

$S \rightarrow aSb$

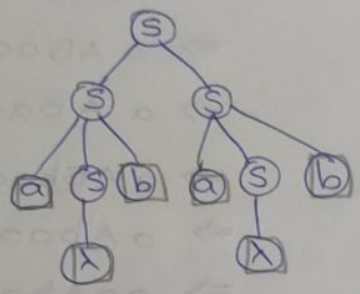
$S \rightarrow bSa$

$S \rightarrow SS$

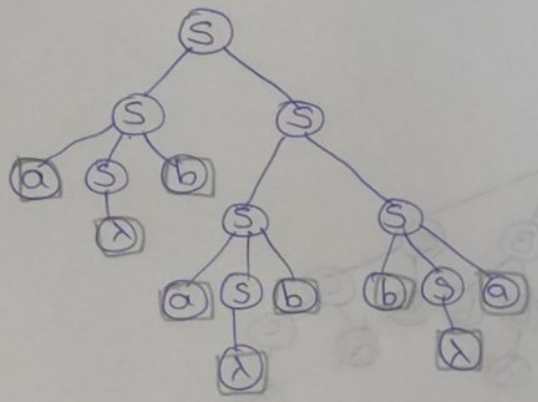
$S \rightarrow \lambda$

i) abab ii) ababba

i) abab



ii) ababba



Ambiguity in Grammar:

A grammar is said to be ambiguous if there exists a string w, such that there exists at least:

i) Two distinct leftmost derivation

or

ii) Two distinct rightmost derivation

or

iii) Two distinct parse trees

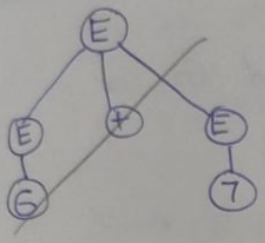
to derive w using rules of G.

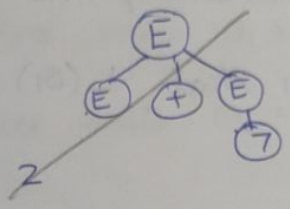
Q Show that,

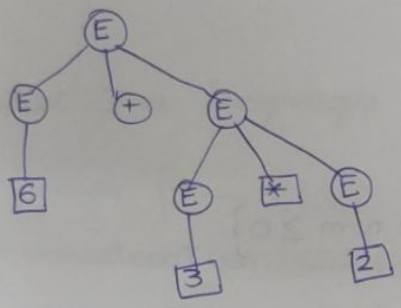
$E \rightarrow E+E$

$E \rightarrow E*E$

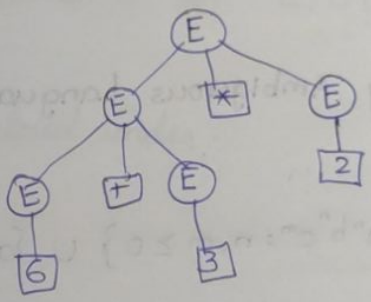
$E \rightarrow 0|1|2|\ldots|9$ is ambiguous







6 + 3 * 2



6 + 3 * 2

---

* A language is inherently ambiguous if $\forall$ G of where $L(G) = L$, G is ambiguous

Eg: $\{a^n b^n c^m : n,m \geq 0\} \cup \{a^n b^m c^m : n,m \geq 0\}$

Q Show that the following grammar is ambiguous

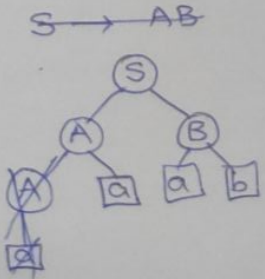
$S \rightarrow AB \mid aB$

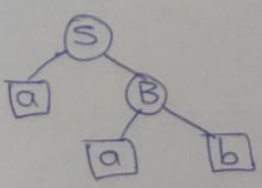
$A \rightarrow Aa \mid a$

$B \rightarrow ab$

Let $w = aab$

$S \rightarrow AB$





Unambiguous Grammar and Language:

*. A grammar is unambiguous if for all strings $w \in L(G)$, there exists exactly one derivation tree that derives w.

Eg: $G = \{S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda\}$ → Each step is unique

*. A language L is unambiguous if there exists at least one grammar G_1 such that $L(G_1) = L$ and G_1 is unambiguous.

Inherently Ambiguous Language:

Eg:

$$\underbrace{\{a^n b^n c^m : n, m \geq 0\}}_{L_1} \cup \underbrace{\{a^n b^m c^m : n, m \geq 0\}}_{L_2}$$

Grammar for L_1

$S_1 \rightarrow AB$

$A \rightarrow aAb \mid \lambda$

$B \rightarrow cB \mid \lambda$

Grammar for L_2

$S_2 \rightarrow CD$

$C \rightarrow aC \mid \lambda$

$D \rightarrow bDc \mid \lambda$

Combining,

$S \rightarrow AB \mid CD$

$A \rightarrow aAb \mid \lambda$

$B \rightarrow cB \mid \lambda$

$C \rightarrow aC \mid \lambda$

$D \rightarrow bDc \mid \lambda$

Q Show that Context Free Languages are closed under union

Let L_1 be a CFL

L_2 be a CFL

L_1 is CFL $\Rightarrow$ There exists $G_1 = (V_1, \Sigma_1, S_1, P_1)$

L_2 is CFL $\Rightarrow$ There exists $G_2 = (V_2, \Sigma_2, S_2, P_2)$

$S \rightarrow S_1 | S_2$ will give the CFL for $L_1 \cup L_2$

---

*. Context Free Languages are closed under:

i) Union $[S \rightarrow S_1 | S_2]$

ii) Concatenation $[S \rightarrow S_1 . S_2]$

iii) Kleene Star $[S \rightarrow SS_1 | \lambda]$

If L_1 is context-free, then there exists $G_1 = (V_1, \Sigma_1, S_1, P$ and L_1^* is also regular

iv) Reversal $[X \rightarrow \alpha_1 \alpha_2 \ldots \alpha_{n-1} \alpha_n$ becomes $X \rightarrow \alpha_n \alpha_{n-1} \ldots \alpha_2 \alpha_1]$

*. Context-Free Languages are not closed under intersection and complement.

Q Prove that CFL are not closed under intersection

Consider $L_1 \cap L_2$

$$\{a^n b^n c^m\} \cap \{a^n b^m c^m\} = \{a^n b^n c^n\} : n \geq 0\}$$

Proved later

Q) Prove CFL are not closed under complement.

Suppose CFL are closed under complement.

Let L_1 and L_2 be CFL

According to De Morgan's law,

$$(L_1 \cap L_2)^c = L_1^c \cup L_2^c$$

$$L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$$

We know $L_1 \cap L_2$ is not closed. ~~under~~ But,

$(L_1^c \cup L_2^c)^c$ is closed [$\because$ Union is closed and we assume ^{c} to be closed]

$\therefore$ $(L_1^c \cup L_2^c)^c$ is closed

$\therefore$ It is a contradiction

$\therefore$ complement is not closed for CFL

Minimizing Ambiguity (Might become unambiguous)

Eg: $S \longrightarrow aABb \mid \lambda$

$A \longrightarrow aA \mid \lambda$

$B \longrightarrow Bb \mid C$

$C \longrightarrow bB \mid \lambda$

i) Eliminating λ-productions (except for $S \longrightarrow \lambda$);

ii) Eliminating unit productions ($V_1 \longrightarrow V_2$)

iii) Eliminating useless productions

i) Eliminating λ- productions:

Eliminating $A \rightarrow \lambda$,

$S \longrightarrow aABb \mid \lambda \mid aBb$

$A \longrightarrow aA \mid a\lambda$

$B \longrightarrow Bb \mid C$

$C \longrightarrow bB \mid \lambda$

Eliminating $C \to \lambda$,

$S \to aABb \mid aBb \mid \lambda$

$A \to aA \mid a$

$B \to Bb \mid C \mid \lambda$

$C \to bB$

Eliminating $B \to \lambda$,

$S \to aABb \mid aBb \mid \lambda \mid aAb \mid ab$

$A \to aA \mid a$

$B \to Bb \mid C \mid b$

$C \to bB \mid b$

∴ After eliminating λ, we have:

$S \to aABb \mid aBb \mid aAb \mid ab \mid \lambda$

$A \to aA \mid a$

$B \to Bb \mid b \mid C$

$C \to b \mid bB$

ii) Eliminating unit-productions:

Eliminating. $B \to C$

$S \to aABb \mid aBb \mid aAb \mid ab \mid \lambda$

$A \to aA \mid a$

$B \to Bb \mid b \mid bB \mid b$

$C \to bB \mid b$

iii) Eliminating useless-production:

★. If a production rule involves an undefined variable, then it is useless. (Only the part with the undefined variable is useless)

★. If a variable recursively points to itself, then it is useless.
Eg: $A \to aA$

★. Inaccessible variables are useless.

NORMAL FORMS

CHOMSKY NORMAL FORM

Used for parsing a string

GREIBACH NORMAL FORM

Used for converting Context-free grammar to Push-Down Automata

*. A regular grammar is said to be in left-linear normal form if every rule is of the form:

$A \rightarrow \lambda$

$A \rightarrow x$

$A \rightarrow By$

[$A \in V$, $B \in V$, $x \in T$ & $y \in T$]

(Left-linear normal form will have only one terminal after the variable)

Eg:

$S \rightarrow Aaba$
$A \rightarrow Bbb \mid abb$
$B \rightarrow \lambda$

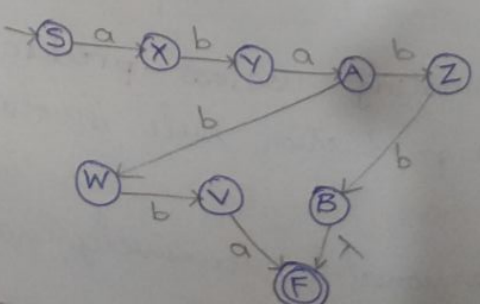
} Regular Left-Linear Grammar

⇩

$S \rightarrow Xa$
$X \rightarrow Yb$
$Y \rightarrow Aa$
$A \rightarrow Zb$
$Z \rightarrow Bb$
$A \rightarrow Wb$
$W \rightarrow Vb$
$V \rightarrow a$
$B \rightarrow \lambda$

} Left-Linear Normal Form

Equivalent NFA:



: A regular grammar is said to be in right-linear normal form if every rule is of the form:

$A \to \lambda$
$A \to x$
$A \to yB$

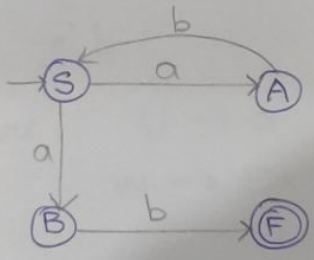
$[A, B \in V$
$x, y \in T]$

Eg: $S \to abS \mid ab$

$\Downarrow$

$S \to aA$
$A \to bS$
$S \to aB$
$B \to b$

} Right-Linear Normal Form



CHOMSKY NORMAL FORM (CNF)

A CFG (Context Free Grammar) is said to be in CNF if every rule is of the form:

$S \to \lambda$ $\quad S \in V$ (Start Variable)
$X \to a$ $\quad X, Y, Z \in V$
$X \to YZ$ $\quad a \in T$

Convert Context-Free Grammar to Chomsky Normal Form

Eg:

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \lambda$

i) Minimizing ambiguity.

Eliminating λ productions:

Eliminate $B \rightarrow \lambda$,

$S \rightarrow ASA \mid aB \mid a$

$A \rightarrow B \mid S \mid \lambda$

$B \rightarrow b$

Eliminate $A \rightarrow \lambda$,

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$ → $S \rightarrow S$ is a useless production

$A \rightarrow B \mid S$

$B \rightarrow b$

Eliminating unit productions:

Eliminate $A \rightarrow B$,

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid S$

$B \rightarrow b$

Eliminate $A \rightarrow S$,

$S \rightarrow ASA \mid aB \mid a \mid SA \mid \&AS$

$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

ii) Let $U \rightarrow a$
$V \rightarrow b$

iii) For any rule that is not in CNF, replace the terminal by U and V respectively.

$S \rightarrow ASA \mid UB \mid AS \mid SA \mid a$
$A \rightarrow ASA \mid UB \mid AS \mid SA \mid a \mid b$
$B \rightarrow b$

iv) Let $W \rightarrow SA$

$S \rightarrow AW \mid UB \mid AS \mid SA \mid a$
$A \rightarrow AW \mid UB \mid AS \mid SA \mid a \mid b$
$B \rightarrow b$

} Chomsky Normal Form

where, $U \rightarrow a$
$W \rightarrow SA$
$V \rightarrow b$ is not used

Q $L = \{a^n b^m c^{n+m} : n, m \geq 1\}$

CFG:

$S \rightarrow aAc$
$A \rightarrow aAC \mid bBc$
$B \rightarrow bBc \mid \lambda$

i) Minimizing ambiguity.

Eliminating λ productions:

Eliminate $B \rightarrow \lambda$,

$S \rightarrow aAc$
$A \rightarrow aAc \mid bBc \mid bc$
$B \rightarrow bBc \mid bc$

Eliminating unit productions.

None

Eliminating useless productions.

None

ii) Let $U \rightarrow a$

$V \rightarrow b$

$W \rightarrow c$

iii) $S \rightarrow UAW$

$A \rightarrow UAW \mid VBW \mid VW$

$B \rightarrow VBW \mid VW$

iv) Let $X \rightarrow UA$

$Y \rightarrow VB$

$\therefore S \rightarrow XW$

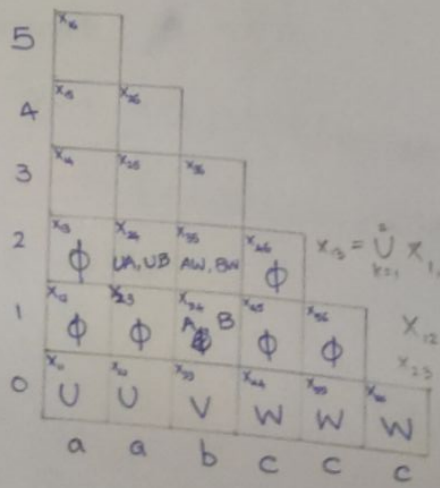
$A \rightarrow XW \mid YW \mid VW$

$B \rightarrow YW \mid VW$

CYK Algorithm : (Cocke - Younger - Kasami) [Check whether the CNF grammar can derive a specific string]

Q Does the above grammar in CNF derive aabccc?

$X_{ij} \rightarrow X_{i,k} \mid X_{k+1,j} \quad \{i \le k \le$



$X_{13} = \bigcup_{k=1}^{2} X_{1,k} X_{k+1,3} = X_{11} X_{23} \cup X_{12} X_{33} = \phi \cup \phi = \phi$

$X_{12} = X_{11} X_{22} = UU \Rightarrow \phi ; \quad X_{34} = X_{33} X_{44} = VW$

$X_{23} = X_{22} X_{33} = UV \Rightarrow \phi ;$

* If X_{1n} contains the starting variable S, then w is accepted

Q) Check whether $w \in L(G_1)$ for

i) w = baaba

G_1:

$S \rightarrow AB|BC$

$A \rightarrow BA|a$

$B \rightarrow CC|b$

$C \rightarrow AB|a$

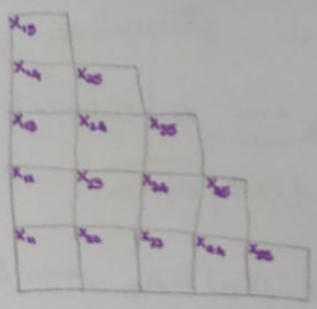
ii) w = abaabb

G_1:

$S \longrightarrow AA|AS|b$

$A \longrightarrow SA|AS|a$

i)



GREIBACH NORMAL FORM (GNF):

A CFG is said to be in GNF if every rule is of the form:

$$A \rightarrow x\beta$$

where, $x \in T$, $\beta \in V^*$

Eg:

$S \rightarrow aAB \mid aB$
$A \rightarrow aBBA \mid b$
$B \rightarrow bBA \mid a$

Convert CFG to GNF:

Eg:

$S \rightarrow aABb$
$A \rightarrow aA \mid \lambda$
$B \rightarrow bB \mid \lambda$

i) Minimizing ambiguity.

Eliminating λ productions:

Eliminating $A \rightarrow aA \mid \lambda$,

$S \rightarrow aABb \mid aBb$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid \lambda$

Eliminating $B \rightarrow \lambda$,

$S \rightarrow aABb \mid aBb \mid aAb \mid ab$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

ii) Let,

$U \rightarrow a$

$V \rightarrow b$

S ⟶

iii) Replace a and b with U and V respectively except when the terminal is in the first position.

$S \rightarrow aABV \mid aBV \mid aAV \mid aV$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$V \rightarrow b$

Q Convert to GNF:

$S \rightarrow AB$

$A \rightarrow aA \mid Bb \mid a$

$B \rightarrow bB \mid a$

Already ambiguity is minimized.

$B \rightarrow bB \mid a$ ⟶ Already in GNF

$A \rightarrow aA \mid a \mid bBb \mid ab$ [$\because B \rightarrow bB \mid a$]

$S \rightarrow aAB \mid aB \mid bBbB \mid abB$ [$\because A \rightarrow aA \mid a \mid bBb \mid ab$]

Let $U \rightarrow a$

$V \rightarrow b$

$S \rightarrow aAB \mid aB \mid bBVB \mid aVB$

$A \rightarrow aA \mid a \mid bBV \mid aV$

$B \rightarrow bB \mid a$

$V \rightarrow b$

Q Convert to GNF:

$S \rightarrow aA|aB$

$A \rightarrow aA|a$

$B \rightarrow bB|aC$

$C \rightarrow BC$

No λ-productions

No unit productions

$C \rightarrow BC$ is a useless production $\Rightarrow$ Eliminate $C \rightarrow BC$

$S \rightarrow aA|aB$

$A \rightarrow aA|a$

$B \rightarrow bB|aC$

~~$C \rightarrow BC$~~

Now $B \rightarrow bB$ is a useless production,

$S \rightarrow aA|aB$

$A \rightarrow aA|a$

~~$B \rightarrow bB$~~

$\therefore$ $S \rightarrow aA$

$A \rightarrow aA|a$

Q

Convert to GNF: [Left-Recursion Problem] $A \rightarrow A\beta$

$\beta \in (V \cup T)^*$

$S \rightarrow Sa|Sb|aa|bb$

No λ productions

No unit productions

No useless productions

$A \rightarrow A\alpha_1|A\alpha_2|\ldots|A\alpha_n|\beta_1|\beta_2|\ldots|\beta_m$

$A \rightarrow \beta_1|\beta_2|\ldots|\beta_m|\beta_1 Z|\beta_2 Z|\ldots|\beta_m Z$

$Z \rightarrow \alpha_1|\alpha_2|\ldots|\alpha_n|\alpha_1 Z|\alpha_2 Z|\ldots|\alpha_n Z$

$S \rightarrow aa|bb|aaZ|bbZ$

$Z \rightarrow a|b|aZ|bZ$

Let $U \rightarrow a$; $V \rightarrow b$

$S \rightarrow aU|bV|aUZ|bVZ$

$Z \rightarrow a|b|aZ|bZ$

⑧ Convert to GNF:

$S \to AB$

$A \to Aa \mid ABa \mid b$

$B \to BBa \mid ab$

No λ-productions

No unit productions

No useless productions

$A \to b \mid bZ$

$Z \to a \mid Ba \mid aZ \mid BaZ$

$B \to ab \mid abZ_1$

$Z_1 \to Ba \mid BaZ_1$

$A \to b \mid bZ_2$

$Z_2 \to a \mid Ba \mid aZ_2 \mid BaZ_2$

Replacing initial variables with their expansions,

$B \to ab \mid abZ_1$

$Z_1 \to aba \mid abZ_1a \mid abaZ_1 \mid abZ_1aZ_1$

$A \to b \mid bZ_2$

$Z_2 \to a \mid aZ_2 \mid aba \mid abZ_1a \mid abaZ_2 \mid abZ_1aZ_2$

$S \to bB \mid bZ_2B$

Let $U \to a$

$V \to b$,

$S \to bB \mid bZ_2B$

$A \to b \mid bZ_2$

$Z_2 \to a \mid aZ_2 \mid aVU \mid aVZ_1U \mid aVUZ_2 \mid aVZ_1UZ_2$

$B \to aV \mid aVZ_1$

$Z_1 \to aVU \mid aVZ_1U \mid aVUZ_1 \mid aVZ_1UZ_1$

$U \to a$

Q

$S \rightarrow Aab \mid aS \mid a$

$A \rightarrow SSa \mid bS \mid ab$

Convert into CNF and GNF

$X_{ij} = X$

Aab

↓

bSab

baab

Check whether babab and aabb can be derived from G.

Minimizing ambiguity.

Eliminating λ-productions:

~~Eliminate~~ None

Eliminating unit-productions:

None

No useless productions

Let $U \rightarrow a$

$V \rightarrow b$

~~$S \rightarrow A$~~

Let $X \rightarrow ~~ab~~ UV$

$Y \rightarrow SS$

$S \rightarrow AX \mid US \mid ~~UV~~ a$

$A \rightarrow YaU \mid VS \mid UV$

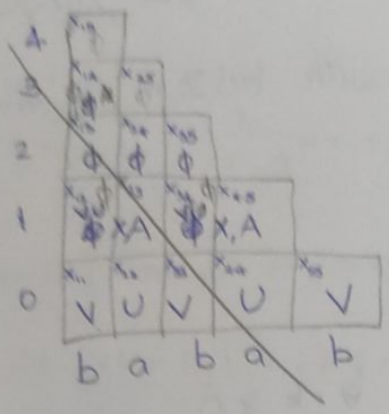
$U \rightarrow a$

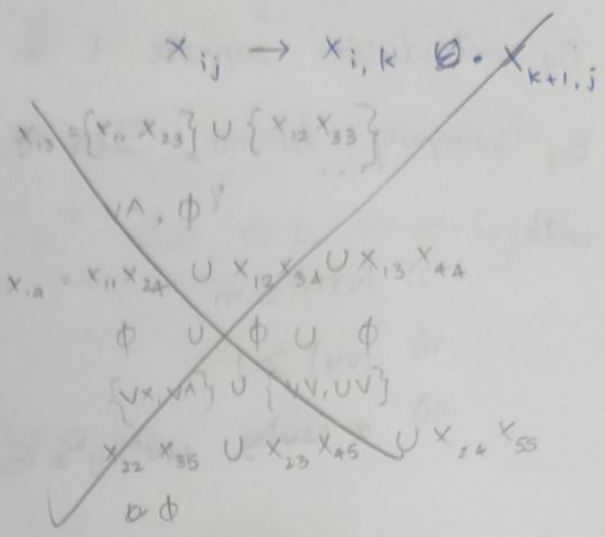
$V \rightarrow b$

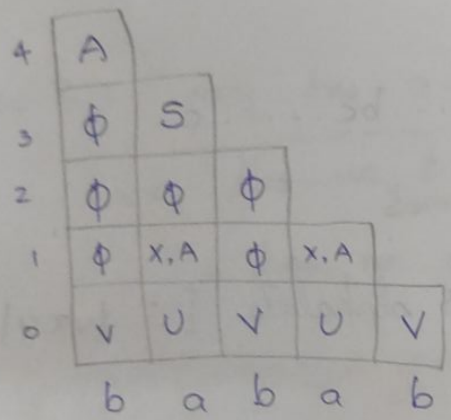
} CNF

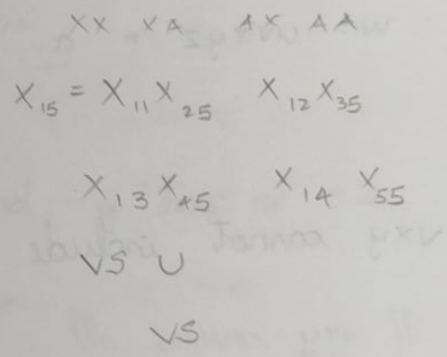
~~Checking~~

Checking if babab can be derived:









Pumping Lemma for CFL

Let L be an infinite CFL, then there exists m such that for all $w \in L$ with $|w| \geq m$, there exists a decomposition $w = uvxyz$ such that,

i) $|vxy| \leq m$

ii) $|v| + |y| \geq 1$ [or, $|vy| \geq 1$]

iii) $uv^k xy^k z \in L \quad \forall \; k \geq 0$

Q Show that $L = \{a^n b^n c^n : n \geq 0\}$ is not context-free

By contradiction, suppose L is a context-free language

By pumping lemma, $\exists m \; \forall w$ with $|w| \geq m$, $\exists w = uvxyz$

with :

i) $|vxy| \leq m$

ii) $|vy| \geq 1$

iii) ~~$uv^k xy^k z$~~ $uv^k xy^k z \in L \; \forall \; k \geq 0$

Let $w = a^m b^m c^m$

$|w| = 3m \geq m$

$w = uvxyz = \underbrace{a \ldots a}_{m \text{ times}} \underbrace{b \ldots b}_{m \text{ times}} \underbrace{c \ldots c}_{m \text{ times}}$

vxy cannot include all 3 alphabets [$\because |vxy| \leq$

If vxy covers all 3 alphabets, it is of the form

$vxy = \ldots \underbrace{ab \ldots bc}_{m \text{ times}} \ldots$

$|vxy| \geq m+2$ $\quad \therefore |vxy| \neq m$

$\therefore$ vxy has at most two alphabets

Case 1 : vxy has only as

Case 2 : vxy has as and bs

Case 3 : vxy has only bs

Case 4 : vxy has bs and cs

Case 5 : vxy has only cs

Case 1: vxy has only as

$v = a^i$

$y = a^j$

$uv^2xy^2z = a^{m+i+j} b^m c^m \notin L \Rightarrow m+i+j=m$

$m+i+j = m \Rightarrow i=j=0$

$\Rightarrow |v| + |y| = i+j = 0+0 = 0$

$|vy| \not\geq 1$

$\therefore$ Contradiction

Case 2: vxy has as and bs

$\therefore vxy$ has no cs

$\therefore$ v has no cs and y has no cs

Compare $uvxyz$ and uxz

The number of cs in $uvxyz$ and uxz ~~is~~ are equal since v and y do not have cs.

$uv^0xy^0z = uxz = a^{m-p} b^{m-q} c^m$ of vy (There is some decrease in as and bs due to the removal)

$uxz \in L \Rightarrow a^{m-p} b^{m-q} c^m \in L$

$\therefore m-p = m \Rightarrow p=0$
$m-q = m \Rightarrow q=0$
} This implies there are no as and bs in vy. We already know there are no cs in vy $\Rightarrow |vy| = 0$

$p=0$ and $q=0 \Rightarrow |vy| = 0 \rightarrow$ Contradiction to $|vy| \geq 1$

Q Show that $L = \{www : w \in \Sigma^*\}$ is not a CFL
$[\Sigma = \{0,1\}]$

Let $w = 0^m 1 0^m 1 0^m 1$

$uvxyz = 0^m 1 0^m 1 0^m 1$

Case 1: vxy has no 1's

Case 2: vxy has one 1

[vxy cannot have more than one 1, since $|vxy| \leq m$]

Case 1: vxy has no 1's

$\Rightarrow v$ and y has no 1's

No. of 1's do not change

$uv^0xy^0z = uxz$

$uxz = 0^{m-p}1 0^m 1 0^m 1$ or $0^m 1 0^{m-p} 1 0^m 1$ or

$0^m 1 0^m 1 0^{m-p} 1$ vxy can be in the first, second or third 0^m

$m - p = m \Rightarrow p = 0$

$|vy| = p = 0 \Rightarrow |vy| \neq 1 \Rightarrow$ Contradiction

---

Q $L_1 = \{a^m b^m c^n d^n\}$

$S \rightarrow AB$

$A \rightarrow aAb \mid \lambda$

$B \rightarrow cBd \mid \lambda$

$\therefore L_1$ is a CFL

Q $L_2 = \{a^m b^n c^n d^m\}$

$S \to aSd \mid A$

$A \to bAc \mid \lambda$

$\therefore L_2$ is a CFL

Cross-Dependency Language

Q $L_3 = \{a^m b^n c^m d^n\}$ is not a CFL

*. If L_1 is context-free and L_2 is context free, then $L_1 \cap L_2$ may or may not be context-free.

*. If L_1 is context-free and L_2 is regular, then $L_1 \cap L_2$ is context-free.

PDA cannot do $a^n b^n c^n$

TURING MACHINE

A Turing Machine is a septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$

Where,

$\Gamma \to$ Tape alphabets $\Sigma \subseteq \Gamma$

$\square \in \Gamma$ is a special blank symbol

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

A finite state automaton that can move right to left and edit the symbols as well.

Q Construct a Turing Machine for $\{a^n b^n : n \geq 0\}$

1. $\delta(q_0, a) = (q_1, x, R)$
2. $\delta(q_1, a) = (q_1, a, R)$
3. $\delta(q_1, b) = (q_2, y, L)$
4. $\delta(q_2, a) = (q_2, a, L)$
5. $\delta(q_2, x) = (q_0, x, R)$
6. $\delta(q_1, y) = (q_1, y, R)$
7. $\delta(q_0, y) = (q_3, y, R)$
8. $\delta(q_3, \square) = (q_f, \square, L)$
9. $\delta(q_3, y) = (q_3, y, R)$ (Does not matter)

Termination:

NOTE: Change state if a symbol is rewritten or if there is a change in direction of movement

Instantaneous Description:

i) aabb

$(q_0, \underline{a}abb) \xrightarrow{①} (q_1, \underline{x}abb) \longrightarrow (xq_1, \underline{a}bb) \xrightarrow{②} (xaq_1, \underline{b}b)$

$\downarrow ③$

$(xxq_1, \underline{y}b) \xleftarrow{①} (xq_0, \underline{a}yb) \xleftarrow{⑤} (q_2, \underline{x}ayb) \xleftarrow{④} (xq_2, \underline{a}yb)$

$⑥ \downarrow$

$(xxyq_1, \underline{b}) \xrightarrow{③} (x\,xq_2, \underline{y}y) \xrightarrow{⑦} (x\,q_2, \underline{x}yy) \xrightarrow{⑤} (xxq_0, \underline{y}y)$

$\downarrow ⑧$

$\checkmark \leftarrow (xxyyq_f) \xleftarrow{⑩} (xxyyq_3\square) \xleftarrow{⑧} (xxyq_3, y)$

ii) aab

$(q_0, \underline{a}ab) \xrightarrow{①} (xq_1, \underline{a}b) \xrightarrow{②} (xaq_1, \underline{b}) \xrightarrow{③} (xq_2, \underline{a}y)$

$\downarrow ④$

$(xxyq_1) \xleftarrow{⑥} (xxq_1, y) \xleftarrow{①} (xq_b, \underline{a}y) \xleftarrow{⑤} (q_2, \underline{x}ay)$

$\downarrow q_1, \square$ is not defined

X

Q Turing Machine for palindromes of even length

$(q_0, a|b) \rightarrow (q_1|q_2, \square, R)$

$(q_1, a|b) \rightarrow (q_1, a|b, R)$

$(q_2, a|b) \rightarrow (q_2, a|b, R)$

$(q_1|q_2, \square) \rightarrow (q_3|q_4, \square, L)$

$(q_3, a) \rightarrow (q_5, \square, L)$

$(q_4, b) \rightarrow (q_6, \square, L)$

$(q_5|q_6, a|b) \rightarrow (q_5|q_6, a|b, L)$

$(q_5|q_6, \square) \rightarrow (q_0|q_0, \square, R)$

$(q_0, \square) \rightarrow (q_f, \square, R)$

(8) Turing Machine for $\{ww : w \in \{0,1\}^*\}$

$\delta(q_0, 0) \longrightarrow (q_1, x, R)$

$\delta(q_0, 1) \longrightarrow (q_1, y, R)$

$\delta(q_1, 0|1) \longrightarrow (q_1, 0|1, R)$

$\delta(q_1, \square) \longrightarrow (q_2, \square, L)$

$\delta(q_1, x|y) \longrightarrow (q_2, x|y, L)$

$\delta(q_2, 0|1) \longrightarrow (q_3, x|y, L)$

$\delta(q_3, 0|1) \longrightarrow (q_3, 0|1, L)$

$\delta(q_3, x|y) \longrightarrow (q_0, x|y, R)$

$\delta(q_0, x|y) \longrightarrow (q_4, x|y, L)$

$\delta(q_4, x|y) \longrightarrow \delta(q_4, 0|1, L)$

$\delta(q_4, \square) \longrightarrow \delta(q_5, \square, R)$

$\delta(q_5, 0) \longrightarrow (q_6, x, R)$

$\delta(q_6, 0|1) \longrightarrow (q_6, 0|1, R)$

$\delta(q_6, x) \longrightarrow (q_7, \square, L)$

$\delta(q_7, 0|1) \longrightarrow (q_{57}, 0|1, L)$

$\delta(q_7, x|y) \longrightarrow (q_7, x|y, L)$

$\delta(q_7, \square) \longrightarrow (q_5, \square, R)$

$\delta(q_5, 1) \longrightarrow (q_8, y, R)$

$\delta(q_8, 0|1) \longrightarrow (q_8, 0|1, R)$

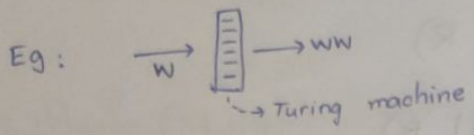
$\delta(q_8, y) \longrightarrow (q_9, \square, L)$

$\delta(q_9, 0|1) \longrightarrow (q_{89}, 0|1, L)$

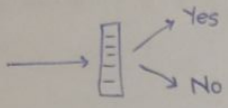
$\delta(q_9, x|y) \longrightarrow (q_9, x|y, L)$

$\delta(q_9, \square) \longrightarrow (q_5, \square, R)$

*. Turing machines can be used as transducers.

Eg: 

*. Turing machine as acceptor:



Turing machine as a copier:

$f(w) = ww$

$\Sigma = \{x\}$

$xxx \rightarrow xxx1 \rightarrow xxx'1 \rightarrow xx111 \rightarrow x x' 111$

$\downarrow$

$x11111$

$\downarrow$

$x'11111$

$\downarrow$

111111

$\downarrow$

$xxxxxx$

Turing machine for multiplication:

$f(4,3) = 12$

$f(1111011) \rightarrow 0000\,c\,0000c \rightarrow 0000c\,\underset{x}{\phi}\,00c$

$\downarrow$

$\underset{y\,y\,y\,y}{\phi\phi\phi\phi}\,c\,x\,00c0000$

$\downarrow$

$0000cx\underset{x}{\phi}0c0000 \leftarrow 0000\,cx\,00c0000$

$\downarrow$

$\underset{y\,y\,y\,y}{\phi\phi\phi\phi}cxx0c00000000 \rightarrow$

→ ϕϕϕϕ C x x x C 000000000000
y y y y

↓

0000 C x x x C 000000000000
12

Turing machine for division :

$x > y$

1111 0 11

x x x x 0 x x

x x x 10 x x

x x x 101 x

x x 1101 x

x x 11011

x 1111011

x 11011 □

$x < y$

1101111

x x 0 x x x x

x 10 x x x x

x 101 x x x

1101 x x x

11011 x x

□ 11011 x x

$x = y$

11011

x x 0 x x

x 10 x x

x 101 x

1101 x

11011

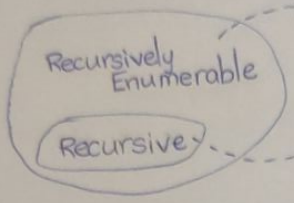
□ 11011

*. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM and let w be a string in Σ^*. Then, the string w is accepted if

$$q_0 w \rightarrow \alpha_1 p \alpha_2 \qquad [p \in F; \alpha_1, \alpha_2 \in \Gamma]$$

*

*.



→ The TM goes into an infinite loop for some $\Sigma^* - L$ → Words that do not belong to

→ At least one TM exists such that it halts for L and $\Sigma^* - L$

*. Set of all Turing machines is countable

*. Set of all languages is uncountable

⇒ ∃ Uncountable languages L for which ∃ no TM

⇒ ∃ infinitely many non RE languages.

Proof: Set of TM is countable

$\delta(q_1, a) = (q_2, x, R)$

$Q = \{q_1, q_2, \ldots, q_k\}$

q_1 is starting state

q_2 is the only final state

$q_1 = 1$	$\square = 1$	$L = 1$
$q_2 = 11$	$a = 11$	$R = 11$
$q_3 = 111$	$b = 111$	
	$x = 1111$	

→ 10 110 110 1111 0 11 00 . . . . . 00 . . . .

, = , , end

Rule 1 Rule 2

Every TM can be mapped to a unique binary number (One-One mapping)

Since binary numbers are countable, the set of TM is also countable

Proof: Set of all languages is uncountable

To show this,

If S is infinitely countable, then 2^S is uncountable. [Cantor Set Theorem]

When,

$S = \Sigma^*$ (set of all words) → Infinitely countable

$2^S = 2^{\Sigma^*}$ (set of all subsets of Σ^*)

(Set of all languages)

$L \subseteq \Sigma^*$

$\Downarrow$

$L \in 2^{\Sigma^*}$

∴ The set of all languages is uncountable

Lemma Proof:

Let S be infinitely countable

If

Suppose 2^S is countable

$S = \{s_1, s_2, s_3, \dots\}$ → Infinitely countable

$2^S = \{\{s_1\}, \{s_2\}, \{s_1, s_2, s_3\}, \{s_3, s_4\}, \dots\}$

$t_1 \quad t_2 \quad t_3 \quad t_4$

	s_1	s_2	s_3	s_4	...
t_1	1	0	0	0	...
t_2	0	1	0	0	...
t_3	1	1	0	0	...
t_4	0	0	1	1	...
⋮	⋮	⋮	⋮	⋮	⋱

$f(t,s) = \begin{cases} 1 & \text{if } s \in t \\ 0 & \text{otherwise} \end{cases}$

$\infty \times \infty$

Let t_d represent the diagonal elements of the ma

$t_d = 11\,01\ldots$

$t_{\bar{d}} = \cancel{1011}\ 0010\ldots$

$t_{\bar{d}} \in 2^S$

$t_{\bar{d}} \neq t_i \quad \forall\, i \geq 1$

$\therefore t_{\bar{d}} \notin 2^S \Rightarrow$ contradiction

$t_{\bar{d}} \neq t_1$ [$\because s_1 \in t_1 ; s_1 \notin t_{\bar{d}}$]

$t_{\bar{d}} \neq t_2$ [$\because s_2 \in t_2 ; s_2 \notin t_{\bar{d}}$]

$t_{\bar{d}} \neq t_3$ [$\because s_3 \notin t_3 ; s_3 \in t_{\bar{d}}$]

* A language L is ~~a regular language~~ recursively enumerable if there exists a Turing machine M such that $L(M) = L$ for all $w \in L$, M halts in a final state. For all $x \notin L$, M either halts in a non-final state or go into an infinite loop.
M accepts L

* A language L is recursive if there exists a Turing machine such that $L(M) = L$ for all $w \in \Sigma^*$. If $w \in L$ it ha in a final state. If $w \notin L$, M halts in a non-final state.
M decides L

Since the set of Turing machines is countable,

Let ~~TM~~ $\overline{\text{TM}} = \{M_1, M_2, M_3, \ldots\}$

$\Sigma = \{a\}$

$\Sigma^+ = \{a, a^2, a^3, \ldots\}$

$L_d = \{a^i \mid M_i \text{ accepts } a^i\}$ [Universal Turing Machine]

$\overline{L}_d = \{a^i \mid M_i \text{ does not accept } a^i\}$

Claim: $\overline{L_d}$ is not recursively enumerable

Let $\overline{L_d}$ be recursively enumerable.

Then, there exists a Turing machine $M_k \in T$ such that

$$L(M_k) = \overline{L_d} \longrightarrow (*)$$

a^k cannot ~~be~~ belong to both L_d and $\overline{L_d}$

Suppose $a^k \in L_d$ ←

M_k accepts a^k

$\Rightarrow a^k \in L(M_k)$ Contradiction

By (*),

$a^k \in \overline{L_d}$ ←

Suppose $a^k \in \overline{L_d}$ ←

By (*),

$a^k \in L(M_k)$ Contradiction

$\Rightarrow M_k$ accepts a^k

$\Rightarrow a^k \in L_d$ ←

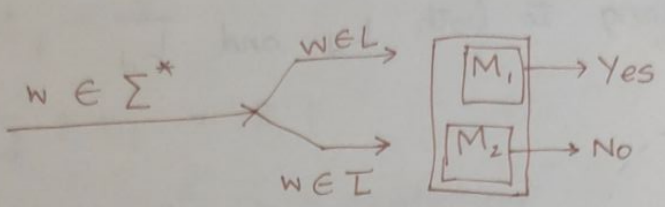
$\therefore a^k \notin L_d$ and $a^k \notin \overline{L_d}$ $\longrightarrow$ Contradiction

$\therefore \overline{L_d}$ is not recursively enumerable.

*. If L and $\overline{L}$ are recursively enumerable, then L is recursive and vice versa.

If L is recursively enumerable, $\exists$ M_1 such that $L(M_1) = L$

If $\overline{L}$ is recursively enumerable, $\exists$ M_2 such that $L(M_2) = \overline{L}$



⇒ ~~Final~~ Halting is possible for all $w \in \Sigma^*$

If $w \in L$, M_1 will halt

If $w \notin L$, M_2 will halt

[$\because$ $w \in L$ or $w \in \overline{L}$]

Claim: L_d is not recursive

Suppose L_d is recursive.

L_d is recursively enumerable and $\overline{L_d}$ is recursively enumerable.

We know $\overline{L_d}$ is not recursively enumerable.

$\therefore$ This is a contradiction.

$\therefore$ L_d is not recursive

UNDECIDABILITY

Decidable Problems:

NFA to DFA

Check if the DFA is empty

Is L(DFA) finite?

Does $w \in L(DFA)$?

Is L(CFG) finite?

Does $w \in L(CFG)$?

Undecidable Problems:

Is a CFG ambiguous? → No guarantee to terminate finitely

Check if the language of two CFGs are equivalent

Does $w \in L(TM)$?

Halting Problem:

Given a Turing machine and a string w, does M halt on w or not.

Proof:

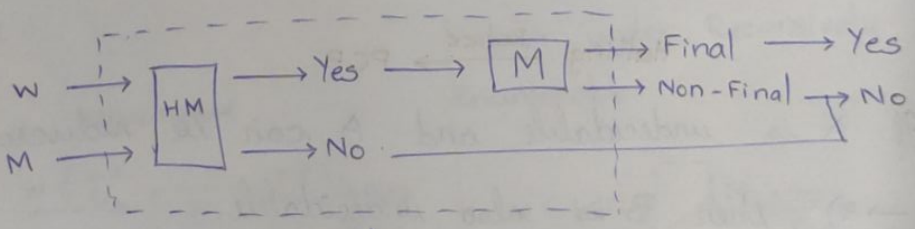
Suppose the halting problem is decidable.

Let L be any recursively enumerable language.

There exists a Turing Machine M for L such that $L(M) = L$

Let HM be the halting machine.

Then,



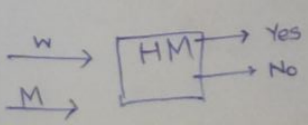
↓

This machine exists ⇒ L is recursive

∴ All recursively enumerable languages can be converted to become recursive using the halting machine.

⇒ Contradiction

Membership Problem [$w \in L(TM)$?]



* Post's Correspondence Problem (PCP): → Undecidable

Given two sets of finite strings of size k, does there exist a permutation m (1,2,3,...,k) such that,

$$S_{11}, S_{12}, \ldots S_{1m} = t_{11}, t_{12}, \ldots t_{1m}$$

Eg:

$$P_1 = \{\underset{s_1}{110}, \underset{s_2}{10}, \underset{s_3}{100}\}$$

$$P_2 = \{\underset{t_1}{11}, \underset{t_2}{010}, \underset{t_3}{10}\}$$

(213)

$$S_2 S_1 S_3 = 10110100$$
$$t_2 t_1 t_3 = 0101110 \quad \times$$

(12)

$$S_1 S_2 = 11010$$
$$t_1 t_2 = 11010 \quad \checkmark$$

Halting $\xrightarrow{\text{Reduce}}$ PCP

*. If A is undecidable and A can be reduced to B, (A → B), then B is also undecidable.

*. A reduction is a polynomial time algorithm,

$$P_1 \xrightarrow[\text{Reduce}]{f} P_2$$

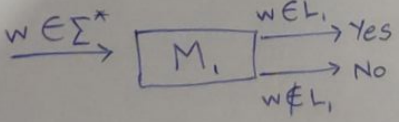
$$\alpha \in P_1 \iff f(\alpha) \in P_2$$

Closure Properties of Recursive and Recursively Enumerable Languages

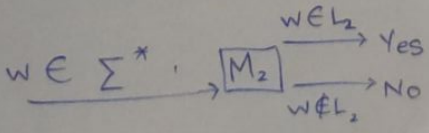
	Recursive	Recursively Enumerable
Union	✓	✓
Intersection	✓	✓
concatenation	✓	✓
Star	✓	✓
Complement	✓	×
Difference	✓	×

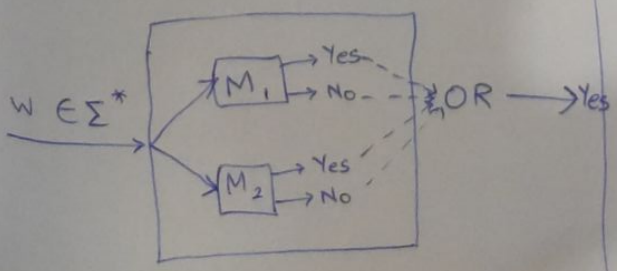
Show $L_1 \cup L_2$ is recursive if L_1 and L_2 are recursive

$\because L_1$ is recursive,

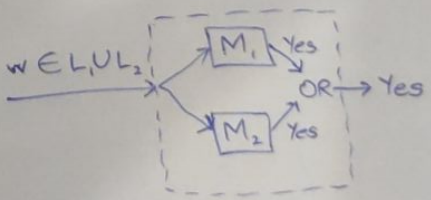


$\because L_2$ is recursive,





Union under Recursively Enumerable



Robustness: The robustness of a mathematical object is measured by its invariance to certain changes.

*. The definition of ~~two~~ Turing machines is robust.

*. Suppose we allow the head to stay put;

i.e: $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$

(L: Left, R: Right, S: Stay)

This feature does not allow Turing machine to recognize additional languages.

Multitape Machine Simulation: