Construction of finite automata equivalent to a Regular expressio

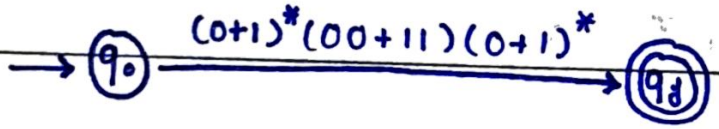
Step 1: Construct a transition system, equivalent to the given regular expression using empty string (Λ) moves with the help of Kleen's theorm

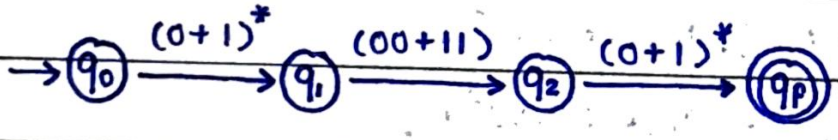
Step 2: Construct the transition table for given system and convert it to a DFA. with removing empty moves. We reduce no. of state is possible.

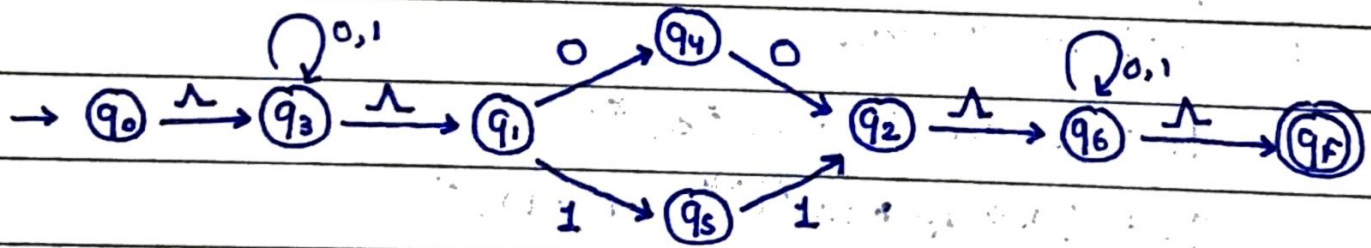
↓ not means minimize remove states which are not relevent.

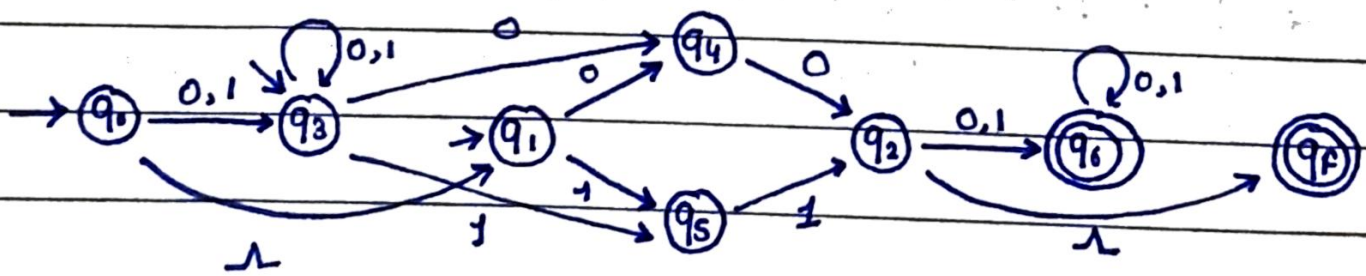
Construct the FA equivalent to r.e.

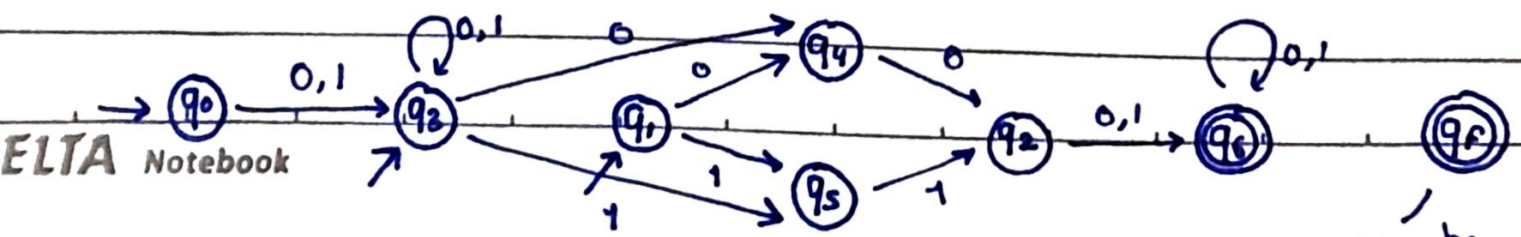
$$(0+1)^*(00+11)(0+1)^*.$$

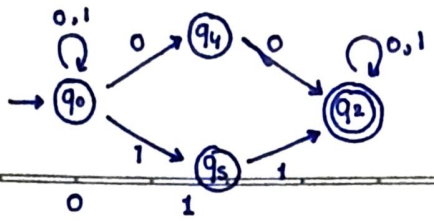












final if asked for DFA, then continue after this.

Date _____

Page No. _____

if for them one initial state then start [q0, q1]

To both initial.

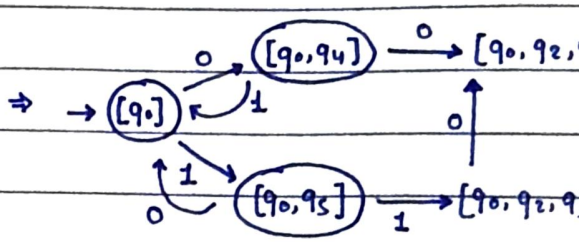
q0 [q0,q4] [q0,q5]

[q0,q4] [q0,q2,q4] [q0]

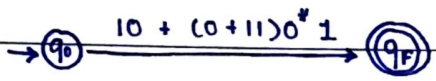
[q0,q5] [q0] [q0,q2,q5]

[q0,q2,q4] [q0,q4,q2] [q0,q5,q2]

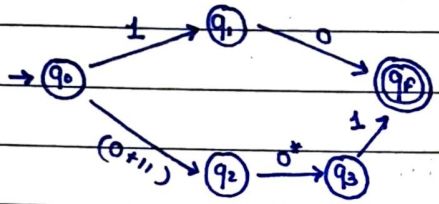
[q0,q2,q5] [q0,q4,q2] [q0,q2,q5]

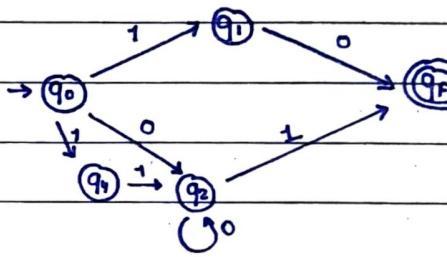


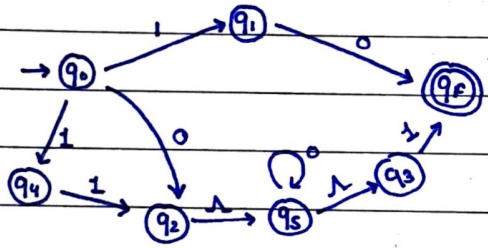
Construct for $10 + (0+11)0^*1$

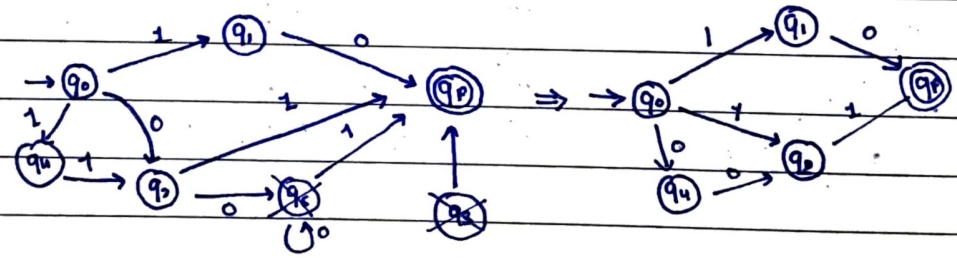


short-cut.







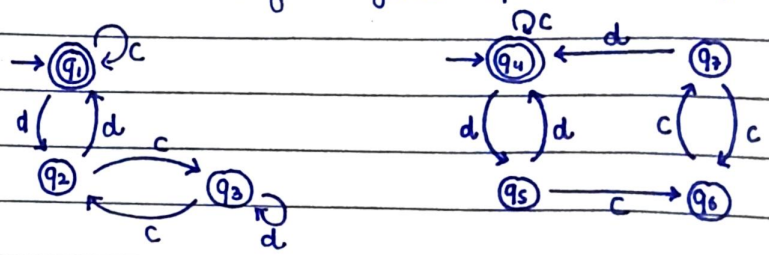


	0	1
q_0	q_4	q_1, q_2
q_1	q_F	ϕ
q_2	ϕ	q_F
q_4	q_2	ϕ

	0	1
q_0	q_4	q_1, q_2
q_1, q_2	q_F	q_F
q_4	q_2	ϕ
q_2	ϕ	q_F

Equivalance of 2 F.A. (Comparison Method)

If regular expression of 2 machin are equal than they are equal.
we can use identity to find equivalent set / regual expression

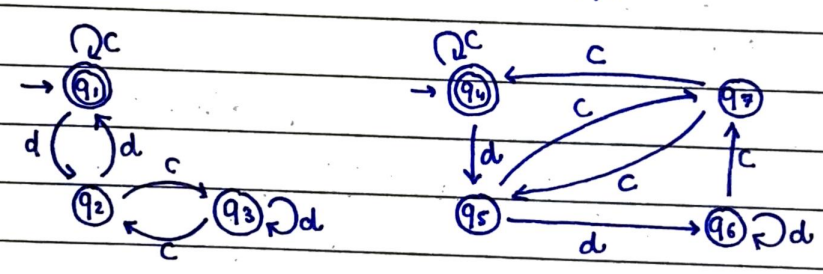


Determine whether M & M' are equivalance?

Using comparision method

check if we end at (final, final) or (non-final, non-final) if other than this than not equivalance.

	M, M'		
	(q_0, q_0')	(q_c, q_c')	(q_d, q_d')
initial →	(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
	(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
	(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
	(q_2, q_7)	(q_3, q_6)	(q_1, q_4)



not equivalance.

Pumping Lemma for Regular Set

Theorem: Let $M=(Q, \Sigma, \delta, q_0, F)$ be a FA with 'n' states. Let 'L' be a regular set accepted by M. Let $\omega \in L$ and $|\omega| \geq m$. If $m \geq n$ then $\exists\ x, y, z$ s.t.

$$\omega = xyz, \quad y \neq \Lambda$$

and $xy^i z \in L \quad \forall\ i \geq 0$.

Proof: Let $\omega = a_1 a_2 \dots a_m \quad m \geq n$

$\delta(q_0, a_1 a_2 \dots a_m) = q_i$ for $i = 1, 2, \dots, m$

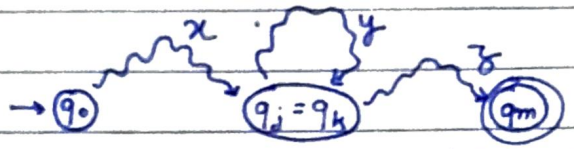
$Q_1 = \{q_1, q_2, \dots, q_m\}$

ie, q_i is the sequence of states in the path with path value $w = a_1 a_2 \dots a_m$. As there are only 'n' distinct states. at least 2 states in q_i must coincide. Among the various pair of states, we take the first pair of repeated states. Let these be $q_j \& q_k$ ($q_j = q_k$). Then, $0 \le j, k \le n$

The string w can be decomposed into 3 substrings, $a_1 a_2 \dots a_j$, $a_{j+1} a_{j+2} \dots a_k$ and $a_{k+1} a_{k+2} \dots a_m$ respectively.

x y z

As $k \le n$; $|xy| \le n$ and $w = xyz$. The path with path value w in the transition diagram M is given below:



$\delta(q_0, a_1, \dots, a_i) = q_i$

The automata M starts from initial state q_0. On applying the string x, it reaches (q_j (or) q_k). On applying y, it comes back to $q_j = (q_k)$. So after the application of y^i ($i \ge 0$); the automaton is in q_j. On applying z, it reaches q_m, is a final state. Hence, $xy^iz \in L$. As every state in q_i is obtained by applying an input symbol, $y \ne \Lambda$.

Application of pumping lemma:

This theorem is used to prove, that certain sets are not regular

step 1: assum that L is regular. let 'n' be number of states in the corresponding FA.

step 2: choose a string 'w' such that $|w| \ge n$. Use "pumping Lemma" to write $w = xyz$ with $|xy| \le n$ & $|y| > 0$.

step 3: Find a suitable integer 'i' s.t. $xy^iz \notin L$. This is a contradiction our assumption & hence, L is not regular

↙ use to only prove not regular. can't use to prove it is regular. as pumping lemma is only a necessary condition

Q1. Show that $L = \{a^{i^2} \mid i \ge 1\}$ is not regular. ↗ use grammar to prove regular

Q2. Show that $L = \{0^i 1^i \mid i \ge 1\}$ is not regular

Solution 1: (i) Let L be a regular set. Let n = number of states in the FA.

(ii) $w = a^{n^2}$; $|w| \ge n$

~~let $w = xyz$; $|xy| \le n$ & $|y| > 0$~~

$|w| = |x| + |y| + |z| = n^2$

by using "length" method

$i = 2$ in xy^iz

$|xy^2z| = |x| + 2|y| + |z|$

$= (|x| + |y| + |z|) + |y|$

$\le n^2 + n$ ($\because |x| + |y| + |z| = n^2$ & $\because |xy| \le n \Rightarrow |y| \le n$)

$< $, as $|y| > 0$

$\Rightarrow n^2 = |xyz| \leq |xy^2z| \leq n^2 + n$
$< n^2 + 2n + 1 = (n+1)^2$

$\Rightarrow n^2 < |xy^2z| < (n+1)^2$

$\Rightarrow xy^2z \notin L$

This contradicts our assumption that L is regular
$\therefore$ By pumping lemma, L is not a regular set.

Solution 2: (i) Let L is regular set. Let n = number of states in the FA
(ii) $\omega = 0^n 1^n$; $|\omega| \geq n$

by using "set" method.

Let $\omega = \underbrace{0^n 1^n}_{xyz}$; $|xy| \leq n$ & $|y| > 0$

$xyyz = 0^k$ or $y = 1^k$ or $y = 0^l 1^k$

$xy^0z = 0^{n-k} 1^n$ $\notin L$ as you two should have equal power

$xy^0z = 0^n 1^{n-k}$ $\notin L$

$xy^0z = 0^{n-l} 1^{n-k}$ $\notin L$.

Closure Properties of Regular Set:

union, concatination, Kleen's closur already done.

to proves from book

R is regular set $\Rightarrow R^T$ is regular $\quad R = aab \Rightarrow R^T = baa$

$\Rightarrow R^c = \Sigma^* \setminus R$ is regular

R reg. set $\Rightarrow M = (Q, \Sigma, \delta, q_0, F)$ s.t $T(M) = R$

$M' = (Q, \Sigma, \delta', F, q_0)$ for reversing initial & final states will interchange & arc use reversed

to prove $\Rightarrow$ show.

R is regular set $\Rightarrow R^c$ (complement) $= \Sigma^* \setminus R$

to prove we just make all non-final state as final & final state as non-final.

UNIT - IV : Context Free Language

$A \to \alpha$; $A \in V_N$, $\alpha \in (V_N$

Derivation Tree (Parse Tree)

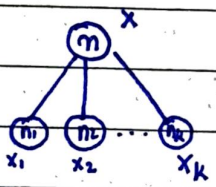
A derivation for a context free grammar (c.f.g) 'G'

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

is a tree satisfying the following condition.

- every vertex has a lable which is a variable or a terminal or empty string (Λ)
- the root has the lable 'S'

- the lable of an internal vertex is a variable. (indu
- if the vertices $n_1, n_2, \ldots, n_k$ return with lables $x_1, x_2, \ldots, x_k$ are decendans of a vertex 'n' wit lable 'x', then

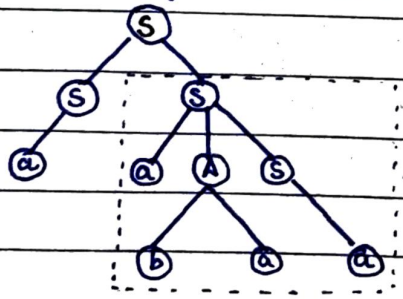


$x \to x_1, x_2, \ldots, x_k$ is a production in G.

- A vertex is a leaf, if its lable is $a \in \Sigma$ or Λ

$G = (\{S, A\}, \{a, b\}, \{S \to aAS \mid a \mid SS;\ A \to SbA \mid ba\}, S)$

for obtaing $\omega = aabaa$ find derivation tree.



Yield of Derivation Tree

It is defined as the concatination of the lables of the leaves, without repetation from left to right ordering

Sub tree of a derivation tree

A sub-tree of a derivation tree 'T' is a subtree is

- whose root is some vertex 'v' of T
- whose vertices are the decendent of v together with their lables
- whose edges are those connecting the desendent of 'v'.

Theorm: Let $G = (V_N, \Sigma, P, S)$ be a CFG. Then, $S \overset{*}{\Rightarrow} \alpha$ iff there is a ~~decision~~ derivation tree ~~verify~~ with yield 'α'.

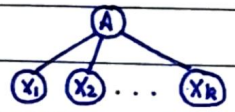
Assume $S \overset{*}{\Rightarrow} \alpha$ ↓ Derivation Tree

induction on ~~number of steps~~ in $S \overset{*}{\Rightarrow} \alpha$

Assume derivation tree 'α' ↓ $S \overset{*}{\Rightarrow} \alpha$

induction on number of internal vertices in derivation tree

Base: $A \to X_1, X_2, \ldots, X_k \in V_N$



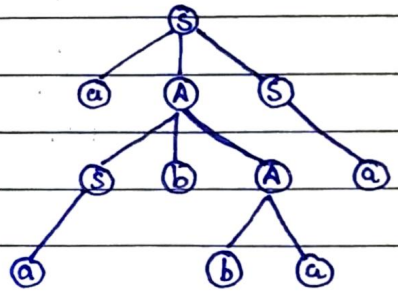
$\leq k$

$S \Rightarrow X_1, X_2, \ldots X_k \overset{k-1}{\Rightarrow} \alpha$

Q. Grammar G $P = \{S \to aAS \mid a,\ A \to SbA \mid SS \mid ba\}$
$\omega = aabbaa \in L(G)$ & construct derivation tree.

$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS$
$\Rightarrow aabbaa$

"leftmost derivation"



aabbaa = yield.

Leftmost & Rightmost Derivation

A derivation $A \overset{*}{\Rightarrow} \omega$ is called a leftmost derivation if we apply production only to the leftmost variable at every step. (A) $\to \alpha$

$S \Rightarrow aASb$ here if we are replacing only leftmost variable first then leftmost.
$ab\underline{S}bSb$

vice-versa for rightmost derivation

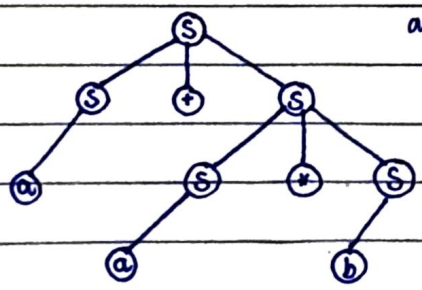
Ambiguity in CFGs

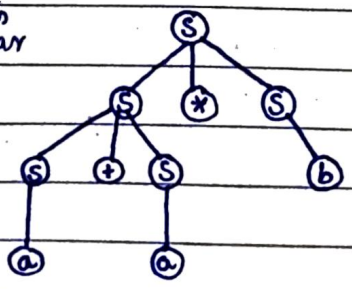
A terminal string $\omega \in L(G)$ is ambiguous if $\exists$ 2 or more derivation trees for ω. (or $\exists$ 2 or more leftmost derivations of ω)

$G = (\{S\}, \{a, b, +, *\}, P, S)$

where $P = \{S \to S+S \mid S*S \mid a \mid b\}$ check if $\omega = a+a*b$ is ambiguous.

ambiguous grammar





as two different derivation tree generates same yield which means the string is ambiguous

yield = a + a * b ← ambiguous → yield = a + a * b

Ambiguous Grammar

A CFG 'G' is ambiguous if ∃ some string $w \in L(G)$, which is ambiguous.

CFG Grammar → Reduced Grammar

$G = (\{S, A, B, C, E\}, \{a, b, c\}, P, S)$ where

$P = \{S \to AB,\ A \to a,\ B \to b,\ B \to C,\ E \to c \mid \Lambda\}$

$w \in L(G)$

C ⇒ doesn't have any production, such kind of production is redundant leads to dead state

① $S \to \ldots\ B \to C \xrightarrow{\times} w$ (can't help to derive terminal)

② $S \to AB \to AC \to \ldots\ E$ (×)

we can't reach 'E' variable every we should remove this from grammar

who to reduce grammar ??

$G \xrightarrow{\text{reduced}} G'$ using '4' steps / theorem.

Def^n: 1. Let $G = (V_N, \Sigma, P, S)$ be a CFG. G is said to be reduced (or non-redundant) if every symbol in $V_N \cup \Sigma$ appears in the course of derivation of some terminal string.

Theorem: For every CFG 'G' ∃ a reduced grammar 'G'' which is equivalent to G.

The reduced grammar can be constructed in 2 steps:-

Step 1: Construct a grammar G_1 equivalent to G so that every variables in G_1 derives some terminal string.

Step 2: Construct G' equivalent to G_1 so that every symbol in G' appears in some sentential form of G'.

CONSTRUCTION (STEP 1)

(a) construction of V_N'

Define $W_i \subseteq V_N$ by recursion:-

$W_1 = \{A \in V_N \mid A \to w;\ w \in \Sigma^*\}$

$W_{i+1} = W_i \cup \{A \in V_N \mid \exists$ some production $A \to \alpha$ $\alpha \in (\Sigma \cup W_i)^*\}$

$W_i \subseteq W_{i+1}\ \forall i$

stop at $W_k = W_{k+1}$. "gives us variables in new grammar"

(b) construction of P'

$P' = \{A \to \alpha \mid A, \alpha \in (V_N' \cup \Sigma)^*\}$

consider previous question $S \to AB,\ A \to a,\ B \to b,\ B \to C,\ E \to C$

step 1: $W_1 = \{A, B, E\}$ ← all variable having RHS, only terminal or string of it.

$W_2 = W_1 \cup \{S\}$

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

$W_3 = W_2$ stop

$V_N' = \{A, B, E, S\}$

$\therefore P' = \{S \to AB,\ A \to a,\ B \to b,\ E \to c\}$

Step 2: Construction of W_i

$W_1 = \{S\}$

$W_{i+1} = W_i \cup \{X \in V_N \cup \Sigma \mid \exists$ a production on $A \to \alpha$ with $A \in W_i$ & α containing $X\}$

stop $W_k = W_{k+1}$

$V_N' = V_N \cap W_k$ & $\Sigma' = \Sigma \cap W_k$

$P' = \{A \to \alpha \mid A \in W_k\}$

Now from previous: (G) $P = \{S \to AB, A \to a, B \to b, B \to C, E \to c\}$

↓ step 1

$W_1 = \{S\}$

$W_2 = \{S\} \cup \{A, B\}$

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

(G') $P = \{S \to AB, A \to a, B \to b, E \to c\}$

↓ step 2

$W_3 = \{S, A, B\} \cup \{a, b\}$

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

$W_4 = W_3$ stop

(G'') $V_N'' = \{S, A, B\}$

$\Sigma'' = \{a, b\}$ $P'' = \{S \to AB, A \to a, B \to b\}$

reduced grammar ↵

Q. find the reduced grammar equivalent to G whose production are:-

$S \to AB \mid CA$, $B \to Bc \mid AB$, $A \to a$, $C \to aB \mid b$

Step 1: $W_1 = \{A, C\}$ → RHS have combination of terminal & variable

$W_2 = \{A, C, S\}$ ↵

$W_3 = W_2$ stop ; $V_N' = \{S, A, C\}$

$P' = \{A \to a, C \to b, S \to CA\}$

Step 2: $W_1 = \{S\}$

$W_2 = \{S, C, A\}$

$W_3 = \{S, C, A, a, b\}$

$V_N'' = \{S, A, C\}$

$P'' = \{A \to a, C \to b, S \to CA\}$

$G' = (V_N'', \{a, b\}, P'', S)$ is reduced grammar equivalet to given grammar.

Q. Reduce the grammar:-

$S \to aAa$; $A \to Sb \mid bCC \mid DaA$, $C \to abb \mid DD$, $E \to aC$; $D \to aDA$

Step 1: $W_1 = \{C\}$, $W_2 = \{C, A, E\}$, $W_3 = \{C, A, E, S\}$

($C \to abb$; $A \to bCC$; $E \to aC$)

$W_4 = \{C, A, E, S\} = W_5$ stop

$G_1 = (\{S, A, C, E\}, \{a, b\}, \{S \to aAa, A \to Sb \mid bCC, C \to abb, E \to aC\}, S)$

step 2: $W_1 = \{S\}$, $W_2 = \{S, A, a\}$, $W_3 = \{S, A, a, b, C\}$

$W_4 = \{S, A, C, a, b\} = W_3$ stop.

$G_1 = (\{S, A, C\}, \{a, b\}, \{S \to aAa, A \to Sb / bCC, C \to abb\}, S)$

Elimination of Null and unit production

The production of the form $A \to \Lambda$ $(A \in V_N)$ is called as a Null production

A unit production

Theorm: If $G = (V_N, \Sigma, P, S)$ is a CFG, then we can find a CFG - G_1 having no null productions s.t. $L(G_1) = L(G) - \Lambda$.

Def^n: A variable 'A' in a CFG is "nullable" if $A \overset{*}{\Rightarrow} \Lambda$

$C \to AB \to \Lambda$

$\Rightarrow$ $A \to \Lambda$ $B \to \Lambda$

Proof: Construction:

(i) Find the nullable variable recursively:

$W_1 = \{A \in V_N \mid A \to \Lambda \text{ is in } P\}$

$W_{i+1} = W_i \cup \{A \cup V_N \mid \exists \text{ a production on } A \to \alpha, \alpha \in W_i^*\}$

$W_{k+1} = W_k$ stop

2. Construction of P'

(i) Any production whose RHS doesn't have any nullable variables is included in P'.

(ii) If $A \to X_1 X_2 \ldots X_k$ is in P; the production of the form $A \to \alpha_1 \alpha_2 \ldots \alpha_k$ are included in P' where $\alpha_j = X_i$, if $X_i \in W_k$; $\alpha_i = X_i$ or Λ if $X_i \in W$ and $\alpha_1 \alpha_2 \ldots \alpha_k \neq \Lambda$ By (2); we get several production in P'. The productions are obtained either by not erasing.

Q. $G = (V_N, \Sigma, P, S)$ where

$P = \{S \to aS / AB, A \to \Lambda, B \to \Lambda, D \to b\}$

construct a grammar G_1 without null productions s.t.

$L(G_1) = L(G) - \{\Lambda\}$

step 1: Set of Nullable variables

$W_1 = \{A, B\}$; $W_2 = \{A, B, S\}$; $W_3 = W_2$ stop

step 2: (i) $D \to b$ is included in P'

(ii) $S \to aS$ gives rise to $S \to aS$ and $S \to a$

$S \to AB$ gives rise to $S \to AB$ and $S \to A$, $S \to B$ (either $B \to \lambda / A$-

$P' = \{ S \to aS \mid AB \mid A \mid B,\ D \to b \}$

it is not reduced, we only start from removing null-production then we reduced it.

removal of unit-production

Theorm: If G is a CFG we can find a CFG, G_1 which has no null or unit productions.

Proof: I-step: removal of null productions.

II-step 1: Define $W_i(A)$ recursively $\forall A \in V_N$

$W_0(A) = \{A\}$

$W_{i+1}(A) = W_i(A) \cup \{B \in V_N \mid C \to B \text{ is in } P \text{ with } C \in W_i(A)\}$

stop $W_k(A) = W_{k+1}(A)$

~~$W_k(A) = W(A)$~~

step 2: Construction of A-productions:

The A-production are either

(a) the non-unit production in G'

(b) $A \to \alpha$ where $B \to \alpha$ is in G with $B \in W(A)$ & $\alpha \notin V_N$.

Q. $P = \{ S \to AB,\ A \to a,\ B \to c \mid b,\ C \to D,\ D \to E,\ E \to a \}$
eliminate unit production

Sol^n: No null production

1. $\to W_0(S) = \{S\}$, $W_1(S) = \{S\}$; $W(S) = \{S\}$
2. $\to W_0(A) = \{A\}$, $W_1(A) = \{A\}$; $W(A) = \{S\}$
3. $\to W_0(B) = \{B\}$, $W_1(B) = \{B, C\}$, $W_2(B) = \{B, C, D\}$
   $W_3(B) = \{B, C, D, E\} = W_4(B)$; $W(B) = \{B, C, D, E\}$
4. $\to W_0(C) = \{C\}$, $W_1(C) = \{C, D\}$, $W(C) = \{C, D, E\}$
   $W(D) = \{D, E\}$
   $W(E) = \{E\}$.

step 2: $\{ S \to AB,\ A \to a,\ B \to b,\ E \to a;\ B \to a$ $B \to C \to D \to E \to a$ (non-terminal)
$C \to a,\ D \to a \}$
this doesn't have

Reduce grammar, Null, unit.

Pumping Lemma for CFL

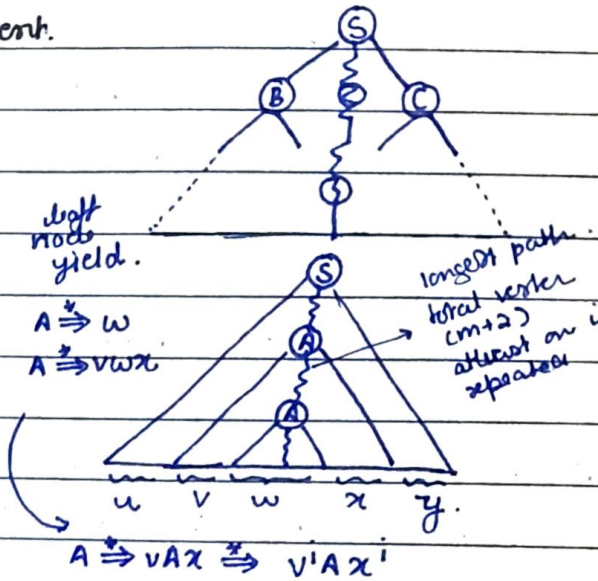
Let L be a CFL. Then, we can find a natural no. 'n' s.t.

1. Every $z \in L$ with $|z| \geq n$ can be written as $uvwxy$ for some strings u, v, w, x & y.
2. $|vx| \geq 1$ means atleast one should be present.
3. $|vwx| \leq n$
4. $uv^iwx^iy \in L \ \forall i \geq 0$.

Proof: We decide $\Lambda \in L$ or not. When $\Lambda \in L$ then consider $L - \{\Lambda\}$ and construct a grammar $G = (V_N, \Sigma, P, S)$ in CNF generating $L - \{\Lambda\}$

Let $|V_N| = m$ and $n = 2^m$. To prove that n is the required natural no. we start with $z \in L$, $|z| \geq 2^m$, and construct a derivation tree 'T' of z. If the length of longest path in 'T' is atmost 'm' then yield of $T = |z| \leq 2^{m-1}$

But, $|z| \geq 2^m > 2^{m-1}$



Hence, T has a path 'P' of length greater than or equal to $(m+1)$. P has atleast $(m+2)$ vertices, and only the last vertex is a leaf. Thus, in P all the levels except last one are variables. As $|V_N| = m$, some label is repeated. We chose the repeated label as follows:

We start with the leaf of P & travel along P upwards. We stop when some label 'B' is repeated. Let V_1 & V_2 are the 2 vertices with label 'B'; V_1 belong / being near to the root. In P, the portion of the path from V_1 to leaf has only one label namely B which is repeated, and so its length is atmost $(m+1)$. Let T_1 & T_2 be the subtrees with V_1 & V_2 as root and z_1 & w as yields respectively. As P is the longest path in T, the portion of P from V_1 to leaf is the longest path in T_1 & is of length at most $(m+1)$. Thus $|z| \leq 2^m$.

As z & z_1 are yields of T & T_1 respectively (T_1 is a subtree of T). We can write $z = uz_1y$ As z_1 & w are the yields of T_1 & T_2 respectively (T_2 is subtree of T_1), we can write $z_1 = vwx$. Also $|vwx| > |w|$. So $|vx| \geq 1$. Thus, we have $z = uvwxy$. As T is a S-tree and T_1 & T_2 are B-trees; we get

$S \overset{*}{\Rightarrow} uBy$. $B \overset{*}{\Rightarrow} vBx$ and $B \overset{*}{\Rightarrow} w$.

As $S \overset{*}{\Rightarrow} uBy \Rightarrow uwy$; $uv^iwx^iy \in L$

For $k \geq 1$, $S \overset{*}{\Rightarrow} uBy \overset{*}{\Rightarrow} uv^kBx^ky \Rightarrow uv^kwx^ky \in L$

$\mathcal{L} = \{a^n b^n c^n \mid n \geq 1\}$ show that it is ¬CFL, not a CFL.

To disprove, the point it is CFL we use pumping lemma of CFL.

Let $z \in \mathcal{L}$, let 'n' be a natural number s.t. $|z| \geq n$.

$z = a^n b^n c^n \Rightarrow |z| = 3n > n$

$z = uvwxy$; let $|vx| \geq 1$ & $|vwx| \leq n$ $\quad \because |vx| \geq 1$

$uv^i wx^i y \notin \mathcal{L}$ for some i.

either prog uv^0wx^0y or uv^2wx^2y

$v = a^i \quad x = b^j$; either $i \neq 0$ or $j \neq 0$

$uv^0wx^0y = a^{n-i} b^{n-j} c^n$

for $i \neq j$ $uv^0wx^0y \notin \mathcal{L}$.

$\mathcal{L} = \{a^p \mid p = \text{prime no.}\}$ is not a CFL.

CFGs $\longrightarrow$ CNF

$A \to \alpha$ $\quad (A \to a, A \to BC)$

$\in V_N \quad \in (V_N \cup \Sigma)^*$

GNF (Greebach Normal Form)

CFGs $\longrightarrow$ GNF

$A \to a\alpha$

$\in V_N \quad \in \Sigma \quad \in (V_N)^*$

Lemma 1: Let $G = (V_N, \Sigma, P, S)$ be a CFG,

Let $A \to B\psi$ be an A production in P.

Let B-production be $B \to \beta_1 | \beta_2 | \beta_3 | \ldots | \beta_j$.

Define: $P_1 = (P - \{A \to B\psi\} \cup \{A \to \beta_i \psi \mid 1 \leq i \leq j\})$

Then, $G_1 = (V_N, \Sigma, P, S)$ is a CFG equivalent to G.

Lemma 2: Let $G = (V_N, \Sigma, P, S)$ be a CFG. Let the set of A-productions be $A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_r | \beta_1 | \beta_2 | \ldots | \beta_s$ (β_i's don't start with A)

Let Z be a new variable.

Let $G_1 = (V_N \cup \{Z\}, \Sigma, P, S)$ where P_1 is defined as:

(i) $A \to \beta_1 Z | \beta_2 Z | \beta_3 Z | \ldots | \beta_s Z | \beta_1 | \beta_2 | \ldots | \beta_s$

(ii) $Z \to \alpha_1 | \alpha_2 | \ldots | \alpha_r | \alpha_1 Z | \alpha_2 Z | \ldots | \alpha_r Z$

(iii) The productions of other variables are as in P.

Then G_1 is a CFG equivalent to G.

Reduction to GNF:

(i) Eliminate null & unit production & reduce G to its CNF. Rename the variables as $A_1, A_2, \ldots, A_n$ with $S = A_1$.

(ii) To get productions in the form $A_i \to \alpha\psi$ or $A_i \to A_j\psi$ where $j > i$, convert A_i production ($i = 1, 2, \ldots, n-1$) in the form $A_i \to A_j\psi$ s.t. $j > i$

(iii) Consider $A_1 \to A_1\psi$ productions, & remove left recursion using lemma 2. by using new variable Z_1.

(iv) Finally, convert productions in the form $A_1 \to a$ or $A_1 \to A_j\psi'$, where $j \neq 1$.

Q. Remove / Reduce to GNF $S \to AA|a$, $A \to SS|b$

Sol^n : ① The CFG in CNF.

② Renaming variables : $A_1 \to A_2A_2|a$; $A_2 \to A_1A_1|b$

~~new check~~ : $A_1 \to a$ $A_i \to A_j\alpha$, $j>i$ ~~occurs~~.

$A_1 \to A_2A_2|a$ $A_2 \to j$ $j>1$. ✓

$A_2 \to A_1A_1|b$ × $j \not> 1$. is not in form $A_i \to A_j\varphi$ $j>i$.

③ $A_1 \to A_2A_2|a$; $A_2 \to aA_1|A_2A_2A_1|b$.

$A_2 \to aA_1|b$ $A_2 \to A_2A_2A_1$ left-recursion

introduce New variable Z_2,

$A_2 \to aA_1|b|aA_1Z_2|bZ_2$

$Z_2 \to A_2A_1|A_2A_1Z_2$.

$A_1 \to A_2A_2|a$ → not GNF

$A_2 \to aA_1|b|aA_1Z_2|bZ_2$ → are in GNF

$Z_2 \to A_2A_1|A_2A_1Z_2$ → not GNF.

using to convert A_1 & Z_2 in GNF

$Z_2 \to aA_1A_1|bA_1|aA_1Z_2A_1|bZ_2A_1Z_2$

$aA_1A_1Z_2|bA_1Z_2|aA_1Z_2A_1Z_2|bZ_2A_1Z_2$ are in GNF.

$A_1 \to aA_1A_2|bA_2|aA_1Z_2A_2|bZ_2A_2|a$ are in GNF.

Reduction to GNF ($S \to AB$, $A \to BS|b$, $B \to SA|a$).

① The CFG in CNF.

② Renaming variables :

$A_1 \to A_2A_3$ $2>1$

$A_2 \to A_3A_1|b$ $3>1$

$A_3 \to A_1A_2|a$ $1 \not> 3$

$\Rightarrow$ $A_3 \to A_2A_3A_2|a$ $2 \not> 3$

$A_3 \to bA_3A_2|a|A_3A_1A_3A_2$

in GNF ↓ using left recursion.

$A_3 \to bA_3A_2|a|aZ_3|bA_3A_2Z_3$

$Z_3 \to A_1A_3A_2|A_1A_3A_2Z_3$

$A_1 \to A_2A_3$ → Not GNF

$A_2 \to A_3A_1|b$ → Not GNF

$A_3 \to bA_3A_2|a|aZ_3|bA_3A_2Z_3$ → are in GNF

$Z_3 \to A_1A_3A_2|A_1A_3A_2Z_3$ → Not GNF.

$A_2 \to bA_3A_2A_1|aA_1|aZ_3A_1|bA_3A_2A_3A_1|b$ → are in GNF

$A_1 \to bA_3A_2A_1A_3|aA_1A_3|aZ_3A_1A_3|bA_3A_2A_3A_1A_3|bA_3$ → are in GNF.

$Z_3 \to$ .... 10 element.

A push down automata is defined as 7-tuple

$(Q, \Sigma, \nabla, \delta, q_0, z_0, F)$ where

(i) Q - is a finite non-empty set of states

(ii) Σ - is a finite not-empty set of input symbol

(iii) ∇ - is a finite non-empty set of push down symbol in pds (push down stor = stack)

(iv) q_0 - is a special state called initial state.

(v) z_0 - is a special push down symbol called as initial symbol on the pds.

(vi) F - set of finial state $F \subseteq Q$.

(vii) δ - is the transition function

∇^*: string from pds

$$\delta : Q \times (\Sigma \cup \{\Lambda\}) \times \nabla \to Q \times \nabla^*$$

reading symbol

reading topmost symbol in pds

state

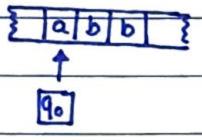
$(q, a, Z) = (q', \beta)$ $\beta \in \nabla^*$

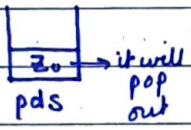
↳ next state.

pushdown symbol

$\beta \in \nabla^*$ $\nabla = \{z_0, z_1, z_2\}$

$\delta(q_0, a, z_0) = (q_1, z_1 z_2)$





we only proceed / travell / tra the string yf,

$\delta(q_0, a, z) = (q_1, z z_2)$ → only move if matc

① $\delta(q, a, Z)$ is finite subset of $Q \times \Sigma^*$ ($Q \times \nabla^*$) the elements of $\delta(q, a, Z)$ are the form of (q', β) where $q' \in Q$ & $\beta \in \nabla^*$ this $\delta(q, a, Z)$ may be any empty set.

② ~~The behaviour of push down automata is non-deterministic~~

③ Due to the defination of δ, the pda, can make a transition without reading any input symbol. such a move is called as empty move or empty transition (provided $\delta(q, \Lambda, Z)$ is defined for some $q \in Q$ & $Z \in \nabla$)

$\delta(q, ab, Zz_0) = \delta(q, \Lambda ab, Z z_0)$ → if $\delta(q, \Lambda, Z)$ → is defined as $= (q', \beta)$

$= \delta(q', ab, \beta z_0)$

④ The push down automata cannot make a transition when pds is empty, in such case pda haults.

⑤ ~~When $\beta = z_1 z_2 \ldots z_k$ is in ∇^*, then z_1 is the topmost symbol in the pushdown store~~

Instantaneous Description (ID)

If $A = (Q, \Sigma, \nabla, \delta, q_0, z_0, F)$

then ID of this pda is defined by (q, x, β) where $q \in Q$, $x \in \Sigma^*$ & $\beta \in \nabla^*$

Initial ID: initial ID is similar to initial state defined as (q_0, x, z_0) $q_0 \in Q$, $x \in \Sigma^*$, $z_0 \in \Gamma^*$

Date _____

Page No. _____

Move Relation: transition relation shown as

$\delta(q_0, aabc, z_0) \vdash \delta(q_1, abc, z_1 z_2) \vdash \delta(q_2, bc, z_1 z_2)$

given $\delta(q_0, a, z_0) = (q_1, z_1 z_2)$

$\delta(q_1, a, z_1) = (q_2, z_1)$

 # Results: (i) If $(q_1, x, \alpha) \vdash^* (q_2, \lambda, \beta)$ → given sub transition for substring

then $\forall y \in \Sigma^*$

$(q_1, xy, \alpha) \vdash^* (q_2, y, \beta)$ & conversely

(ii) If $(q, x, \alpha) \vdash^* (q', \lambda, \psi)$; then $\forall \beta \in \Gamma^*$

$(q, x, \alpha\beta) \vdash^* (q', \lambda, \psi\beta)$.

 # Acceptance by pda

Let $A = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a pda.

(i) By final state

The set accepted by pda by final state is

$T(A) = \{\omega \in \Sigma^* | (q_0, \omega, z_0) \vdash^* (q_f, \lambda, \alpha)$ for some $q_f \in F$ & $\alpha \in \Gamma^*\}$

(ii) By empty (null) store

The set accepted by Null store is defined as

$N(A) = \{\omega \in \Sigma^* | (q_0, \omega, z_0) \vdash^* (q, \lambda, \lambda)$ for some $q \in Q\}$

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

Q. $\delta(q, a, z_0) = \{(q, a z_0)\}$

$\delta(q, b, z_0) = \{(q, b z_0)\}$

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

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

$\delta(q_0, b, a) = \{(q_0, ba)\}$

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

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

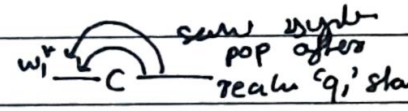
$\delta(q_0, c, b) = \{(q_1, b)\}$

$\delta(q_0, c, z_0) = \{(q_1, z_0)\}$

$\delta(q_1, a, a) = \{(q_1, \lambda)\}$

$\delta(q_1, b, b) = \{(q_1, \lambda)\}$

$\delta(q_1, \lambda, z_0) = \{(q_f, z_0)\}$



find the language accepted by the pda above:

we check if a assumed string, $\omega = bacab$ is accepted.

$\omega = bacab$ $\Gamma = \Sigma \cup \{z_0\}$ but as c is never given to pda $\{a, b, z_0\}$

$\delta(q_0, bacab, z_0) \vdash^* \delta(q_0, acab, bz_0) \vdash^* \delta(q_0, cab, abz_0) \vdash^*$

$\delta(q_1, ab, abz_0) \vdash^* \delta(q_1, b, bz_0) \vdash^* \delta(q_1, \lambda, z_0) \vdash^* \delta(q_f, \lambda, z_0)$

This accepts $T(A) = \{\omega c \omega^T | \omega \in \{a, b\}\}$