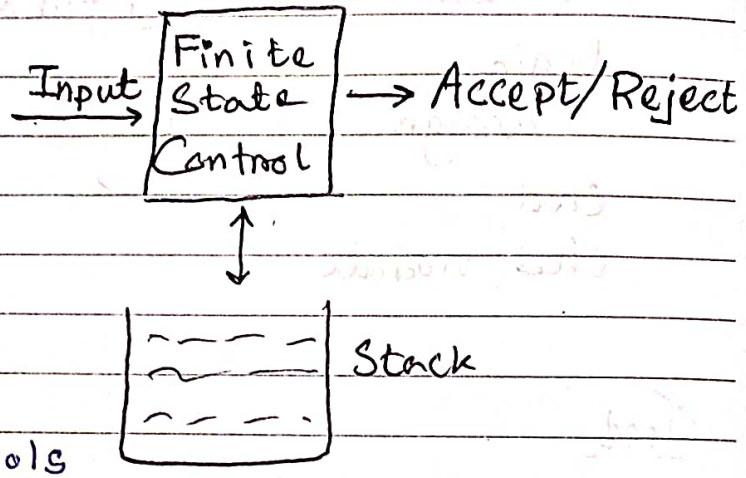
Arpan Basu

Pushdown Automata (PDA):-

P can be represented by 7 tuples.



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

Q = Finite set of States

Σ = Finite set of Input Symbols

Γ = A Finite set of stack Alphabets which are allowed to put into the stack.

δ = Transition Function ~~$\delta = (q$~~ $\delta(q, a, X)$

[q_0 = State in Q $q_0 \in Q$ $a \in \Sigma$ or $a \in \epsilon$
It is assumed not to be an input symbol.

X = stack symbol and $X \in \Gamma$

The output of δ is a finite set of pairs (p, γ) where p is the new state and γ ~~replaces~~ is the string of stack symbols that replaces X at the top of the stack.

$\gamma = \epsilon$ → popped

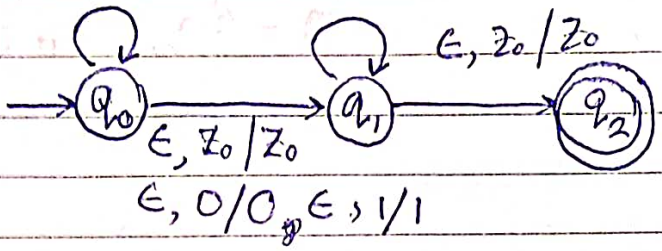
$\gamma = X$ → unchanged

$\gamma = YZ$ → X is ~~now~~ replaced by Z and Y is pushed into the stack.

Order of pushing is right to left]

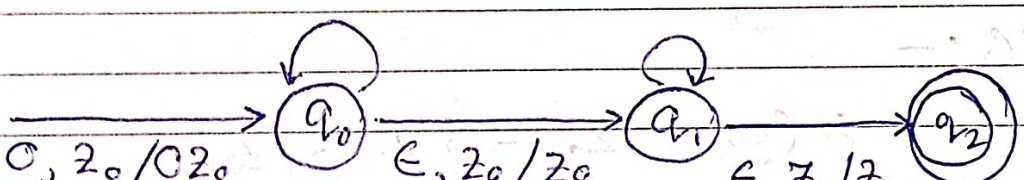
q_0 = Start state
Z_0 = Start symbol of stack
F → set of Accepting / Final state

Eg.



Q. 001100 check if $w = w^R$

$0, 0/\epsilon$ → ϵ means pop
$1, 1/\epsilon$



$\delta(q, a, \gamma) = (p, \gamma')$

If $\gamma = \gamma'$, then stack is empty

Eg. Input String containing even no. of a's $\Sigma = \{a, b\}$

aabbaba

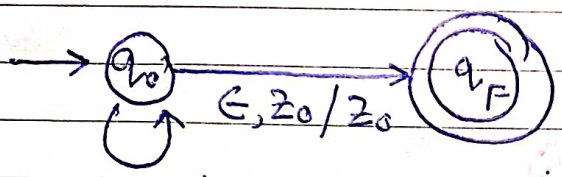
$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

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

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

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

$\delta(q_0, \epsilon, z_0) = (q_f, z_0)$



$a, z_0 / a z_0$

$a, a / \epsilon$

$b, z_0 / z_0$

$b, a / a$

$Q = \{q_0, q_F\}$

$q_0 = \{q_0\}$

$z = \{z_0\}$

$F = \{q_F\}$

PDA for CFG :-

$S \to aSbb \mid a$

$S \to aSA \mid a$

$A \to bB$

$B \to b$

$\delta(q_0, \epsilon, z_0) = \{q, S, z_0\}$

$\delta(q_1, a, S) = \{(q_1, SA), (q_1, \epsilon)\}$

$\delta(q_1, b, A) = \{(q_1, B)\}$

$\delta(q_1, b, B) = \{q_1, \epsilon\}$

$\delta(q_1, \epsilon, z_0) = \{q_F, z_0\}$

Th:- For any CFL, L there exists an NPDA, M such that $L = L(M)$

Th:- If $L = L(M)$ for some NPDA, M then L is a CFL.

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

Leftmost derivation Since G is in GNF.

$M = (\{q_0, q_1, q_f\}, T, V \cup \{z_0\}, \delta, q_0, z_0, \{q_f\})$

start symbol

$\delta(q_0, \epsilon, Z_0) = \{(q_1, SZ_0)\}$

$(q_1, u) \in \delta(q_1, a, A)$

$A \longrightarrow$ all. belongs to P

$\delta(q_1, \epsilon, Z_0) = (\{q_f, Z_0\})$

when we have finished reading the input string and the stack is empty, then we are reaching the final state through ϵ transition.

$w \in L(G)$

$S \overset{*}{\Rightarrow} a_1 a_2 \ldots a_n A_1 A_2 A_3 \ldots A_n$

$\Rightarrow a_1 a_2 \ldots a_n b B_1 \ldots B_k A_2 A_3 \ldots A_n$

$\because A_1 \longrightarrow b B_1 \cdots B_k$

$(q_1, B_1 \cdots B_k) \in \delta(q_1, b, A_1)$

$(q_1, w, SZ_0) \vdash^* (q_1, \epsilon, Z_0)$

$\vdash \Rightarrow$ instantanious description (ID)

$(q_0, w, Z) \vdash (q_1, w, SZ_0) \vdash^* (q_1, \epsilon, Z_0) \vdash (q_f, \epsilon, Z_0)$

So $L(G) \subseteq L(M)$

Let, $w \in L(M)$

$(q_1, w, SZ_0) \vdash^* (q_1, \epsilon, Z_0)$

$w = a_1 a_2 a_3 \ldots a_n$

$(q_1, a_1a_2a_3\cdots a_n, Sz_0) \overset{*}{\vdash} (q_1, \epsilon, z_0)$

$\Rightarrow (q_1, a_1a_2a_3\cdots a_n, Sz_0) \vdash (q_1, a_2a_3,\cdots a_n, u_1z_0)$

Let $u_1 = Au_2$ $\quad S \to a_1k_1$ $\quad Au_2z_0$

$\vdash (q_1, a_3\cdots a_n, u_3u_2z_0)$

$A \to a_2u_3$

We are getting a sentential form $S \overset{*}{\Rightarrow} a_1a_2u_2u_3$ which is possible for leftmost derivation which means G is in GNF.

$\therefore L(M) \subseteq L(G)$

$\therefore L(G) = L(M)$

ϵ is accepted by NPDA although grammar does not contain ϵ.

07 Sunday

$S \to aA$

$A \to aABC \mid bB \mid a$

$B \to b$

$C \to c$

Transition Functions $\longrightarrow$

$\delta(q_0, \epsilon, z_0) = \{(q_1, Sz_0)\}$

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

$\delta(q_1, a, S) = \{(q_1, A)\}$

$\delta(q_1, a, A) = \{(q_1, ABC)(q_1, \in)\}$

$\delta(q_1, b, A) = \{(q_1, B)\}$

$\delta(q_1, b, B) = \{(q_1, \in)\}$

$\delta(q_1, c, C) = \{(q_1, \in)\}$

aaabc

$S \xrightarrow{lm} aA \Rightarrow aaABC \Rightarrow aaaBBC \Rightarrow aaabC$
$\Rightarrow aaabc$

$M = (\{q_0, q_1, q_f\}, \{a, b, c\}, \{S, A, B, C, Z_0\},$
$\delta, \{q_0\}, \{Z_0\}, \{q_f\})$

Instantaneous description $\longrightarrow$

$(q_0, aaabc, Z_0) \vdash (q_1, aaabc, SZ_0)$
$\vdash (q_1, aabc, AZ_0)$
$\vdash (q_1, abc, ABCZ_0)$
$\vdash (q_1, bc, BCZ_0)$
$\vdash (q_1, c, CZ_0)$
$\vdash (q_1, \in, Z_0)$
$\vdash (q_f, \in, Z_0)$

So the string is accepted by the PDA,
There are certain CFG for which we have NPDA
but not DPA.

Deterministic Pushdown Automata (DPDA) :-

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if it is an automaton in definition of NPDA subject to the ~~roje~~ restrictions that:

for every $q \in Q$, $a \in \Sigma \cup \{\in\}$ and $b \in \Gamma$

i) $\delta(q, a, b)$ contains atmost 1 element

ii) if $\delta(q, \in, b)$ is not empty, then $\delta(q, c, b)$ must be empty for every $c \in \Sigma$..

The first of this condition simply requires that for any given input symbol and any stack top, atmost 1 move can be made.

When a $\in$-move is possible for some configuration, no input consuming alternative is available.

A Language L is said to be deterministic context-free language iff there exists a DPDA, M such that $L = L(M)$

$L = \{a^n b^n : n \geq 0\}$ → This is a DCFL

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$

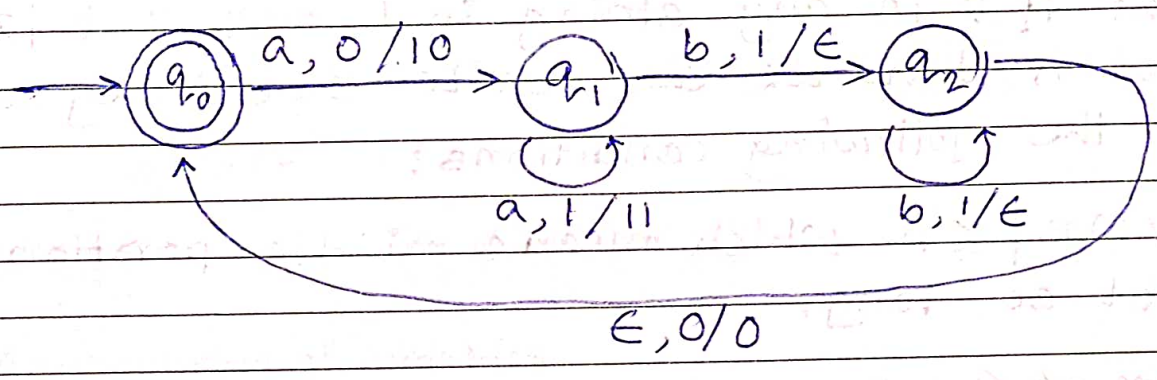
$\delta(q_0, a, 0) = (q_1, 10)$

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

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

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

$\delta(q_2, \epsilon, 0) = (q_0, 0)$



Pumping Lemma for CFL :-

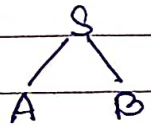
Suppose we have a parse tree according to a CNF grammar, $G = (V, T, S, P)$ and suppose that the yield of the parse tree is a terminal string w. If the length of the longest path in the parse tree is n then $|w| \leq 2^{n-1}$

Basis: $n \geq 1$

$|w| = 1$, $2^{1-1} = 2^0 = 1$

Induction :-

$S \rightarrow AB$



$n > 1$

Pumping Lemma for CFL:→

Statements of pumping Lemma →

Let L be a CFL. Then there exists a constant n such that if z is any string in L such that |z| is atleast n then we can write $z = uvwxy$ subject to the following conditions:

i) $|vwx| \leq n$ which means middle portion is not so long.

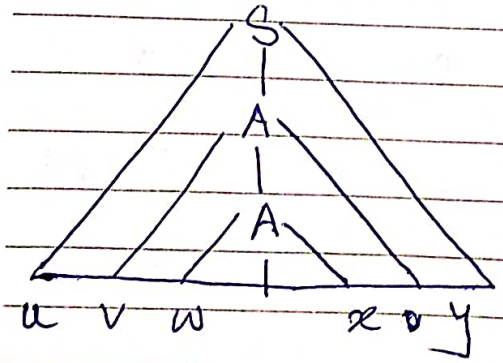
ii) $vx \neq \epsilon$ since v and x are the pieces to be pumped. This condition says that atleast one of the strings being pumped is not empty.

iii) $\forall i \geq 0$, uv^iwx^iy is in L. The strings v and x may be pumped any no. of times including 0 and the resulting string will be a member of L.

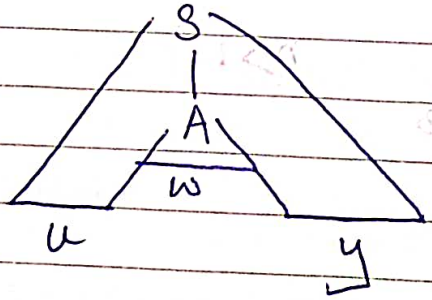
Proof:-

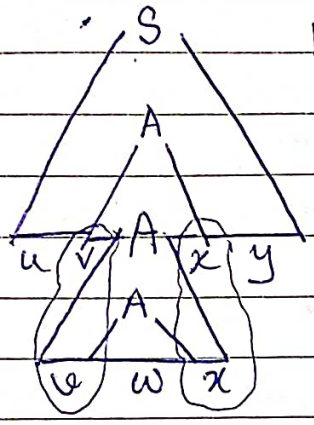
$L(A) = L$

z is a very long string in L



where i=0,





Here $i = 2$

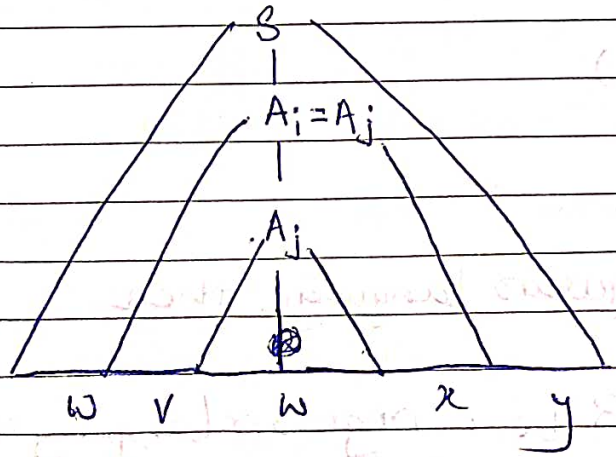
$L = \{\epsilon\}$

$m \longrightarrow$ number of variables

$n = 2^m$

Any parse tree whereas longest path is of length m, or less that would generate a ~~yeild~~ yield of length $2^{m-1} = n/2$ or less

~~Every sufficiently long string in L must have a long path in the parse tree. The largest path in the parse tree for z where k is~~ atleast m and path is of length $k+1$. $k \geq m$



$m+1 \longrightarrow$ one variable repeated

$m+2 \longrightarrow$ 2 variable repeated

denoted by ;

~~Eg.~~ $L = \{a^n b^n c^n : n > 0\}$

$z = uvwxy$

$u = \epsilon$ $\quad w = v^i x y$

$x = \epsilon$ $\quad v = aa$

$vx \neq \epsilon$ $\quad x = bb$

$y = cc$

if $i=1$, $w = aabbcc$

$i=2$, $w = aaaabbcc$ ×

So L is not CFL.

Properties of CFL ⟶

i) Substitution: L be a CFL over some Alphabet Σ and s is a substitution on Σ such that $s(a)$ is a context free language for each a in Σ. Then $s(L)$ is a CFL.
(substitution Theorem)

CFL is closed under ⟶
1. Union
2. Concatenation
3. Closure (*) and (+)
14 Sunday 4. Homomorphism

If L is a CFL, so is L^R

If L is a CFL and R is a regular language then $L \cap R$ is CFL,

Let L_1, L_2, L be CFLs and R be a regular language. Then,
i) $L - R$ is CFL.
ii) $\overline{L}$ is not CFL.
iii) $L_1 - L_2$ is not CFL

Let L be a CFL and h be the homomorphism. Then $h^{-1}(L)$ is a CFL.

Proof:- i) Union → CFL is closed under CFL.

Let G_1 and G_2 be two CFG.

$G_1 = (V_1, \Sigma_1, R_1, S_1)$ $G_2 = (V_2, \Sigma_2, R_2, S_2)$

Without loss of generality, assume that they have disjoint sets of non-terminals.

$G_2 (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R, S)$

$R = R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}$

$\therefore L(G) = L(G_1) \cup L(G_2)$

$S \overset{*}{\underset{G}{\Rightarrow}} w$ iff $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$

Concatenation →

$w = xy$

$L(G) \supseteq L(G_1) \cdot L(G_2)$

w is derived from G_1 and G_2 in 0 or more steps.

New grammar,

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \to S_1 S_2\}, S)$

Kleene's closure:-

$L(G_1)^*$ $S \to \epsilon$

$S \to SS_1$

Grammar, $G = (V_1 \cup \{S\}, \Sigma_1, R_1 \cup \{S \to \epsilon, S \to SS\}, S)$

Intersection of CFL and RL is CFL: –

$L = L(M_1)$. $M_1 = (K_1, \Sigma, \Gamma_1, \Delta_1, Z_0, S_1, F_1)$

$R = L(M_2)$. $M_2 = (K_2, \Sigma, \delta, S_2, F_2)$

$M = (K, \Sigma, \Gamma, \Delta, Z_0, S, F)$

$K = K_1 \times K_2$ $\quad S = (S_1, S_2)$

$\Gamma = \Gamma_1$ $\quad F = F_1 \times F_2$

$((q_1, a, \beta), (p_1, \gamma)) \in \Delta_1$

$(q_2, a) \in \delta_0$

$\Delta = ((\delta(q_1, q_2), a, \beta), (p_1, \delta(q_2, a), \gamma))$

$((q_1, \epsilon, \beta), (p, \gamma)) \in \Delta_1$

$((q_1, q_2), \epsilon, \beta), ((p_1, q_2), \gamma)) \in \Delta$

Context-Sensitive Grammar

Identified by Linear bounded Automata (LBA).
↓
[It is a kind of turing machine with some restriction]

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

Type-1 Grammar/~~CFG~~ CSG :-

A grammar $G = (V, T, S, P)$ is said to be context sensitive if all productions are of the form of $x \rightarrow y$ where $x, y \in (V \cup T)^+$ and $|x| \leq |y|$.

If the production rules follows the above, It is called Type-1 production.

$S \rightarrow \epsilon$. This production is also allowed in Type-1 grammar but in this case S does not appear on the RHS of the production.

This is also called non-contracting grammar. The length of the successive sentential form can never decrease.

All productions can be written in a normal form,

$$xAy \longrightarrow xvy$$

This is equivalent to saying that the production $A \rightarrow v$ can be applied only in the situation where A occurs in a context of the string x on the left and the string y on the right.

Eg. $abAbcd \longrightarrow abAbbcd$

Left and right context must not change.

$A\underline{C} \longrightarrow A$ $C \longrightarrow \epsilon$	LC of C is A RC of C is ϵ

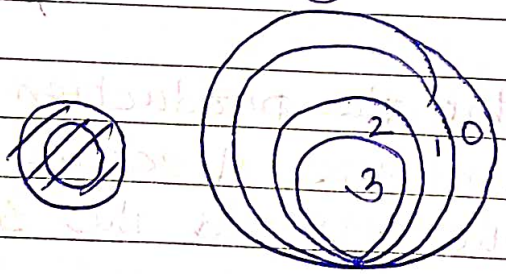
Note:- In a CSG, when we have a production $S \to \epsilon$, we are including ϵ in the CSL. So, except this production $S \to \epsilon$, no other production will decrease the length of the working string.

Type-0 Grammar:— It is any phrase structure grammar without any restrictions.

THEOREMS:—

1) Let G be the Type-0 grammar. Then we can find an equivalent grammar G_1 in which each production is either of the form $\alpha \to \beta$ where α, β are strings of variables only, or of the form $A \to a$ where $A \in V, a \in T$.

So, G_1 is of Type 1/Type 2/Type 3 according as G is of Type-1 or Type-2 or Type-3



1/2/3 are subsets of type-0.

Proof:-

Let G be a grammar with productions. Let G_1 be a grammar constructed considering a production $\alpha \to \beta$ from G.

where α and β have some terminals. In both α and β, we replace every terminal by a new variable C_a ($C_a \to a$) and we get α' and β'.

Thus corresponding to every $\alpha \to \beta$, where α, β contains some terminals, we construct $\alpha' \to \beta'$ and productions of the form $C_a \to a$ for every terminal appearing in $\alpha \to \beta$.

The construction is performed for every $\alpha \to \beta$. The productions for G_1 and are the new productions obtained from the above started construction.

For G_1, the variables are the variables of G together with new variables (C_a). The terminals and start symbol are those of G. So G_1 satisfies required condition and its equivalent to G. $L(G) = L(G_1)$

CSG is called monotonic, or length increasing.

Theorem 2:- Every monotonic grammar G is equivalent to Type-1 grammar.

Proof:- By applying the above theorem, Let G_1 be obtained. We construct $G' \equiv G_1$ as follows.

Consider a production

$A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n$ with $n \geq m$ in G_1. If $m = 1$, then the above production is of Type-1 with left and right context being $\in$. If $m \geqslant 2$ corresponding to the production $A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n$