Conversion of CFG to PDA

1. Convert the given CFG to Greibach Normal form

2. From the start state q_0 of the PDA

→ Add a transition to a new state q_1 with the label $(\lambda, Z_0 \mid S Z_0)$

3. for each production rule of the form $T \rightarrow aUXYZ\cdots$

→ Add a transition from q_1 to q_1 with the label $(a, T \mid UXYZ\cdots)$

4. From state q_1, add a transition to a new final state q_2 with the Label $(\lambda, Z_0 \mid Z_0)$

The PDA begins by artificially pushing the symbol S on to the stack, corresponding to a grammar starting with the variable S in its sentential form. from then on, without the need to change any state, the PDA adds transitions that correspond to each of the production rules in GNF. Finally it adds a transition to the final state, the requirements for which are that the

input should be fully consumed and the stack should be empty. The conflict PDA may leads to non determinism.

Convert the following CFG to PDA

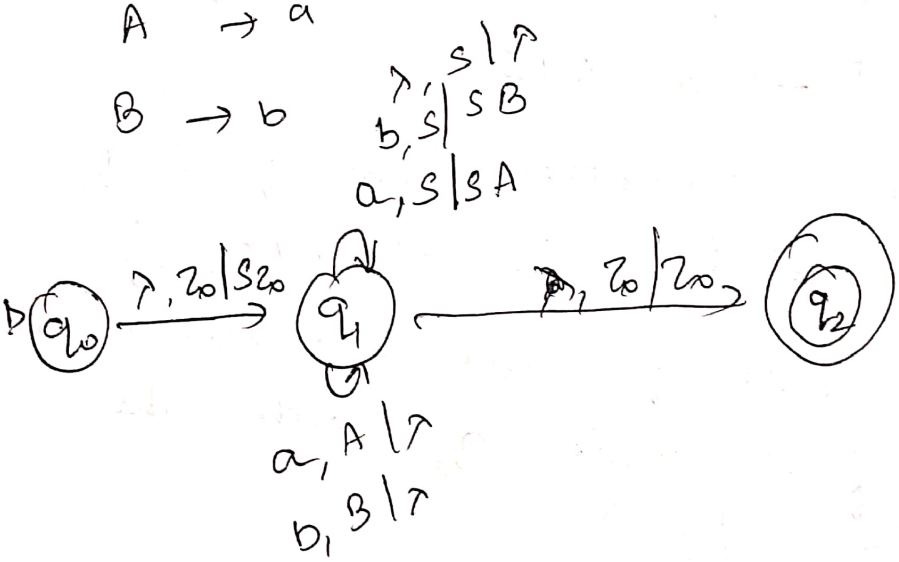
$$S \to aSa \mid bSb \mid \lambda$$

First we need to convert our CFG to GNF.

$$S \to aSA \mid bSB \mid \lambda$$

$$A \to a$$

$$B \to b$$



Let us take a string abaaba

$(q_0, abaaba, Z_0)$

$\vdash (q_1, abaaba, SZ_0)$

$\vdash (q_1, baaba, SAZ_0)$

$\vdash (q_1, aaba, SBAZ_0)$

$\vdash (q_1, aba, SABAZ_0)$

$\vdash (q_1, aba, AB1Z_0)$

$\vdash (q_1, ba, BAZ_0)$

$\vdash (q_1, a, AZ_0)$

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

$\vdash (q_2, Z_0)$ accepted

Let us take rejected string abaa and see how configurations will happen

$(q_0, abaa, Z_0)$

$\vdash (q_\$, abaa, SZ_0)$

$\vdash (q_1, baa, SAZ_0)$

$\vdash (q_1, aa, SBAZ_0)$

$\vdash (q_1, a, SABAZ_0)$

$\vdash (q_1, \lambda, SAABAZ_0)$

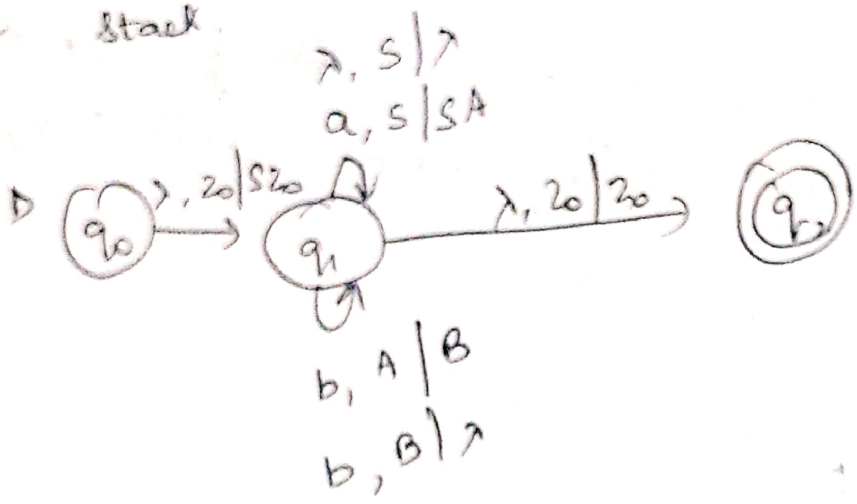
The automata will stays in q_1, even after reading all the input symbols but the stack content is not empty. So the string abaa is rejected.

(2) $S \to aSA \mid \lambda$

$A \to bB$

$B \to b$

Since the given CFG is already in GNF, you can directly push S onto the stack



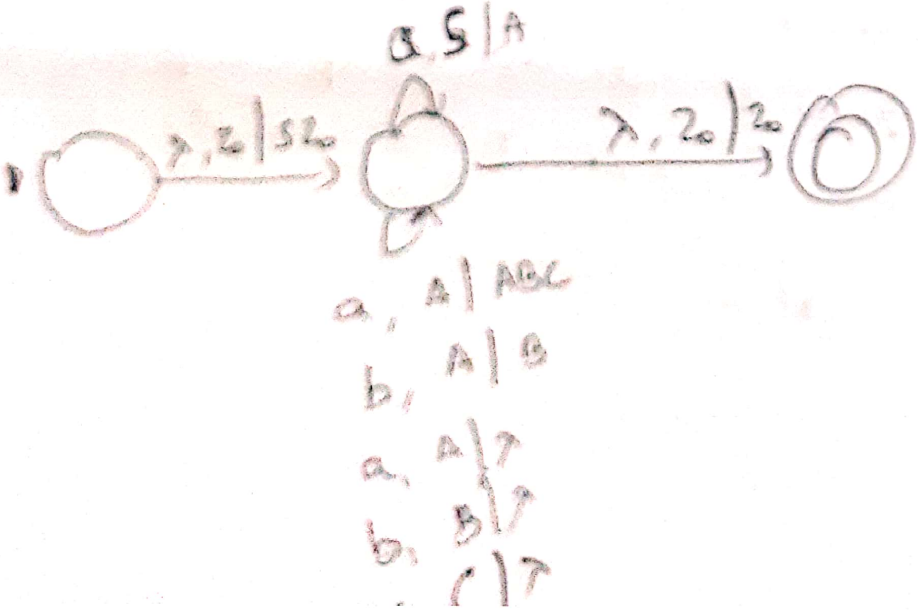
(3) $S \to aA$

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

$B \to b$

$C \to c$

Show how aaabc is accepted



The string aaabc is accepted via

$(q_0, aaabc, Z_0)$

$\vdash (q_1, aaabc, SZ_0)$

$\vdash (q_1, aabc, AZ_0)$

$\vdash (q_1, abc, ABCZ_0)$

$\vdash (q_1, bc, BCZ_0)$

$\vdash (q_1, c, CZ_0)$

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

$\vdash (q_2, \lambda, Z_0)$ - accepted

Normal forms

Chomsky normal form

A CFG G is in Chomsky normal form if all productions are of the form

$X \to YZ$ or

$X \to a$

Every CFG G can be converted to a CFG G' in Chomsky normal form.

Advantage of CNF

⇒ Gives us a way to solve the parsing problem for a CFG G and $w \in \Sigma^*$ does

$w \in L(G)$?

→ If G is in CNF, then the length of derivation of w (if one exists) can be bounded by $2|w|$

Procedure to convert a CFG to CNF

The algorithm for converting CFG to CNF is based on the idea of substitution. If a grammar has a set of productions $A \to aBc$ & $B \to x/y/z$ where x, y & z stands for any piece of a sentential form, we can eliminate the B productions by substituting the RHS of those productions wherever B occurs in other productions. In the A production, we can eliminate B by substituting for B to obtain a new set of productions

$$A \to axc \mid ayc \mid azc$$

Algorithm

1. Eliminate lambda productions

If there is lambda production of the form $A \to \lambda$, we need to eliminate this. Here A is called nullable variable.

Consider the grammar

$S \to aS_1b$, $S_1 \to aS_1b \mid \lambda$

with start variable S. This grammar generates the language $\{a^n b^n \mid n \geq 1\}$.

The lambda production i.e $S_1 \to \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right.

i.e $S \to aS_1b \mid ab$

$S_1 \to aS_1b \mid ab$

This new grammar generates the same language as the original one.

2. Eliminate unit production

Any production of a context free grammar of the form

$A \to B$ where $A, B \in V$ is called a unit production.

To remove unit productions, we use the substitution rule.

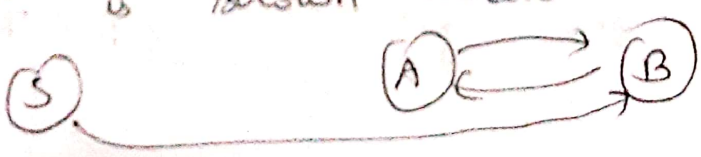
Remove all unit productions from

$S \to Aa \mid B$

$B \to A \mid bb$

$A \to a \mid bc \mid B$

The dependency graph for the unit productions is shown below



$S \to B$, $B \to A$ & $A \to B$ are unit productions.

By seeing dependency graph, we can eliminate unit productions by substitution rule.

$S \to A$.

$B \to a \mid bc \mid bb$

$A \to a \mid bc \mid bb$

Replace B with its productions

$S \to Aa \mid a \mid bc \mid bb$

The final non unit productions are

$S \to Aa \mid a \mid bc \mid bb$

$B \to a \mid bc \mid bb$

$A \to a \mid bc \mid bb$

3. Removal of useless productions

One invariably wants to remove productions from grammar that can never take part in any derivation.

For example $S \to asb \mid \lambda \mid A$

$A \to aA$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production

leaves the language unaffected.

It is very important to perform the above three operation in exactly the same order. i.e "Lambda Units are Useless".

1. Convert a given CFG to CNF

$S \to aSa|bSb|A|\lambda$

$A \to a|b|\lambda$

Step 1 : Eliminate λ-production

S & A are both nullable variables

$S \to aSa|bSb|A|aa|bb$

$A \to a|b$

Note: The modified grammar cannot generate the null string λ. If it is necessary to include λ in the language, a separate rule $S \to \lambda$ can retained.

Step 2 : Eliminate unit productions

i.e $S \to A$ is the only one unit prod

$S \to aSa|aa|a|b|bSb|bb$

$A \to a|b$

step 3 : Eliminate the useless variable A,

we get

$S \to aSa|aa|a|b|bSb|bb$

finally we need to rewrite the production by introducing new intermediate variables.

$S \to AZ$

$A \to a$

$Z \to SA$

$S \to AA$

$S \to BW$

$B \to b$

$W \to SB$

$S \to BB$

$S \to a$

$S \to b$

$S \to \lambda$

---

(2). $S \to abS \mid baS \mid \lambda$

Step1: The given grammar ~~does not have~~ has any lambda production i.e $S \to \lambda$

$S \to abS \mid ab \mid baS \mid ba \mid \lambda$

Step2: No unit production

Step3: No useless production.

Therefore we can convert the grammar to CNF.

$S \rightarrow AD$

$D \rightarrow BS$

$S \rightarrow BC$

$C \rightarrow AS$

$S \rightarrow AB$

$S \rightarrow BA$

$B \rightarrow b$

$A \rightarrow a$

$S \rightarrow \lambda$

(5) $S \rightarrow AS \mid AAAS$

$A \rightarrow SA \mid aa \mid b$

Soln: The start variable S is itself useless. It does not generate any terminal string. The language of the grammar is empty. The equivalent CNF grammar has no production rules at all.

Greibach normal form.

The idea of GNF is to make parsing or derivation linear, that is every step in the derivation should introduce exactly one terminal symbol from the input string. Every production in GNF looks like either the following.

$A \rightarrow aBCD \ldots$

or

$A \rightarrow a$

There is no particular algorithm for converting a given CFG to GNF.

1) Convert the given CFG to GNF.

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$

Soln: $S \rightarrow aSB \mid bSA$

$A \rightarrow a$

$B \rightarrow b$

for $S \rightarrow SS$, we need to somehow introduce a terminal at the beginning i.e we can follow substitution rule $S \rightarrow aSBS \mid bSAS$.

2) $S \rightarrow AB$

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

Solution: Only S production is not in GNF. Substituting for A we get

$S \rightarrow aAB \mid bBB \mid bB$

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

Parsing with Chomsky normal form

CYK algorithm

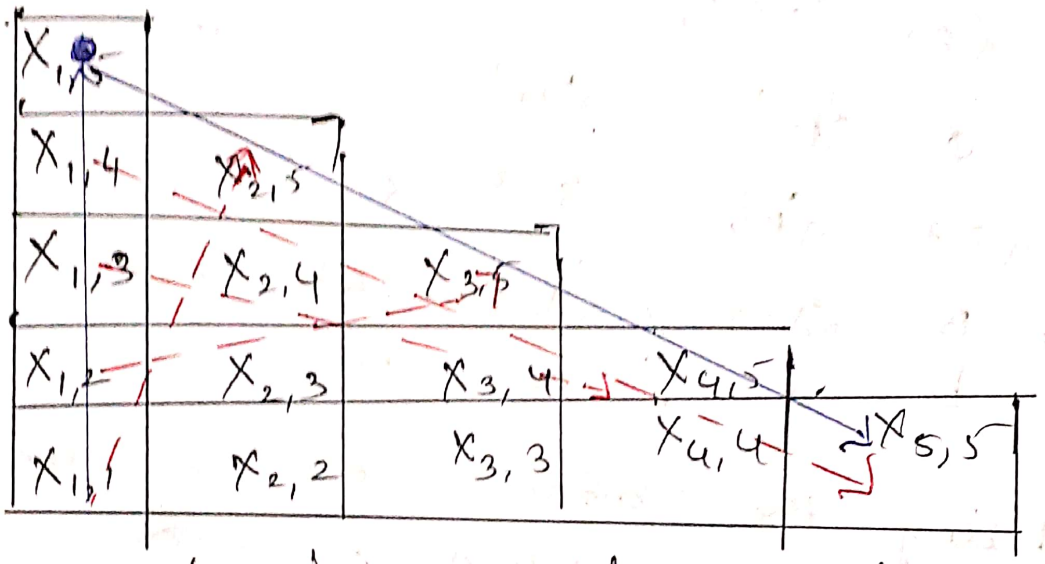
J. Cocke, D. Younger & T. Kasami independently developed an algorithm to recognize whether the given string belongs to the language of the grammar & not. The grammar should be in Chomsky normal form. It uses a dynamic programming or table filling algorithm. We need to construct a triangular table. For example if the string length is 5, then the table looks like this, where each row corresponds to one length of substrings. The bottom row corresponds to string of length 1, second row from bottom row - strings of length 2 & so on, the top row corresponds to string length of w.

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
w_1	w_2	w_3	w_4	w_5

$X_{i,j}$ is the set of variables A such that $A \to w_{ij}$ is a production of G.

→ Compare at most n pairs of previously computed sets.

$$(X_{i,i}, X_{i+1,j}), (X_{i,i+1}, X_{i+2,j}) \cdots (X_{i,j-1}, X_{jj})$$



For example if we want compute $X_{1,3}$ then we need to compare atmost 2 pairs of previously computed sets.

$$X_{1,3} = (X_{i,i}, X_{i+1,j})(X_{i,i+1}, X_{i+2,j})$$

$$= (X_{1,1}, X_{2,3})(X_{1,2}, X_{3,3})$$

$$X_{1,5} = (X_{1,1}, X_{2,5}) \cup (X_{1,2}, X_{3,5}) \cup (X_{1,3}, X_{4,5}) \cup (X_{1,4}, X_{5,5})$$

If $X_{1,5}$ is filled with S then we say that $w \in L(G)$

① Show the CYK algorithm with the following example G

$S \to AB \mid BC$ $\quad C \to AB \mid a$

$A \to BA \mid a$

$B \to CC \mid b$

w is baaba. Check whether the given w is in L(G)?

Soln: The given grammar is already in CNF, we can directly construct the table. Here the |w| is 5

5	{S,A,C}			
4	{φ}	{S,A,C}		
3	φ	{B}	{B}	
2	{S,A}	{B}	{S,C}	{S,A}
1	B	{A,C}	{A,C}	{B}	{A,C}
	b	a	a	b	a

Filling the bottom row is direct, we should check which production gives b & a i.e b can be obtained by B & a can be obtained by A & C.

The second from bottom row is to fill with $X_{1,2} = (X_{1,1}, X_{2,2})$

$= X_{1,1}, X_{2,2}$

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

Take a cross product of these two sets. we get {BA} {BC}. Check which variable gives BA & BC.

There are two S & A i.e S → BC & A → BA

So $X_{3,2}$ is filled with {S, A}. Follow the same procedure to fill the remaing cells in second from bottom row

$X_{2,3} = \{\{A,C\} \otimes \{A,C\}\}$

$= \{AA, CC, AC, CA\}$

Only CC can be obtained from B.

So, $X_{2,3} - \{B\}$

$X_{3,4} = \{A,C\}, \{B\}$

$= \{AB, CB\}$

AB can be obtained from S & C

$X_{4,5} = \{B\} \{A,C\}$

$= \{BA\}, \{BC\}$

$= \{S, A\}$

Next is row 3 can be filled as

$X_{1,3} = (X_{1,1}, X_{2,3}) \cup (X_{1,2}, X_{3,3})$

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

$= \{BB\} \cup \{SA, AA, SC, AC\}$

$= \{BB, SA, AA, SC, AC\}$

$= \phi$

$X_{2,4} = (X_{2,2}, X_{3,4}) \cup (X_{2,3}, X_{4,4})$

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

$= \{AS, AC, CS, CC, BB\}$

Only CC can be obtained in the 1st through B. $\therefore X_{2,4} = \{B\}$

The last row i.e row 5 is filled with S, it indicates that the given string belongs to $L(G_1)$

(2) Apply CYK algorithm for to determine whether the string ababa is in $L(G_1)$.

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

5	{S, A, C}				
4	{B}	{B}			
3	{B}	{S, C}	{B}		
2	{S, C}	{A, S}	{S, C}	{A, S}	
1	{A, C}	{B}	{A, C}	{B}	{A, C}
	a	b	a	b	a

$X_{1,1} = a, \{A, C\}$

$X_{2,2} = b \ \{B\}$

$X_{3,3} = a \ \{A, C\}$

$X_{4,4} = b \ \{B\}$

$X_{5,5} = a \ \{A, C\}$

Row 2

$X_{1,2} = (X_{11}, X_{22})$

$= \{A, C\} \times \{B\}$

$= \{AB, CB\}$

$= \{S, C\}$

$X_{23} = (X_{22}, X_{33})$

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

$= BA, BC$

$= \{A, S\}$

$X_{34} = (X_{33}, X_{44})$

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

$= \{AB, CB\}$

$= \{S, A\}$

$X_{45} = X_{44}, X_{55}$

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

$= \{BA, BC\}$

$= A, S$

Row 3

$X_{1,3} = (X_{11}, X_{23}) \cup (X_{1,2}, X_{3,3})$

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

$= \{AA, AS, CA, CS\} \cup \{SA, SC, CA, CC\}$

$= B$

$X_{24} = (X_{2,2} \cdot X_{3,4}) \cup (X_{2,3} \cdot X_{4,4})$

$\{B\}\{S,C\} \cup \{A,S\} \cdot \{B\}$

$= \{BS, BC, AB, SB\}$

$= \{S, C\}$

$X_{35} = (X_{33} \cdot X_{45}) \cup (X_{3,4} \cdot X_{5,5})$

$= \{A,C\}\{A,S\} \cup \{S,C\} \cdot \{A,C\}$

$= \{AA, AS, CA, CS\} \cup \{SA, SC, CA, CC\}$

$= B$

$X_{14} = (X_{1,1} \cdot X_{2,4}) \cup (X_{1,2} \cdot X_{3,4}) \cup (X_{1,3} \cdot X_{4,4})$

~~$\cup (X_{1,4} \cdot X_{5,5})$~~

$= \{A,C\}\emptyset, \{S,C\} \cup \{S,C\}\{S,C\} \cup \{B\}\{B\}$

$= \{AS, AC, CS, CC\} \cup \{SS, SC, CS, CC\} \cup \{BB\}$

$= B$

$X_{25} = (X_{2,2} \cdot X_{3,5}) \cup (X_{2,3} \cdot X_{4,5}) \cup (X_{2,4} \cdot X_{5,5})$

$= \{B\}\{B\} \cup \{A,S\}\{A,S\} \cup \{S,C\} \cdot \{A,C\}$

$= \{BB\} \cup \{AA, AS, SA, SS\} \cup \{SA, SC, CA, CC\}$

$= B$

$X_{15} = (X_{1,1} \cdot X_{2,5}) \cup (X_{1,2} \cdot X_{3,5}) \cup (X_{1,3} \cdot X_{4,5}) \cup$

$(X_{1,4} \cdot X_{5,5})$

$= \{A,C\}\{B\} \cup \{S,C\}\{B\} \cup \{B\} \cdot \{A,S\} \cup$

$\{B\}\{A,C\}$

$= \{AB, AC\} \cup \{SB, CB\} \cup \{BA, BS\} \cup \{BA, BC\}$

$= S, A, C$

Closure properties of Context free languages

1. Union of CFLs

The set union of two context free languages is a context free language.

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

$L_2 = \{w w^R\}$

we know how to construct CFG for L_1 & L_2 respectively

$S_1 \rightarrow aS_1b \mid \lambda$

$S_2 \rightarrow aS_2a \mid bS_2b \mid \lambda$

Union i.e

$L = \{a^n b^n\} \cup \{w w^R\} = S \rightarrow S_1 \mid S_2$

S_1 & S_2 are start variables of L_1 & L_2 respectively when we take union, the grammar S is new start variable.

2. Context free languages are closed under Concatenation

L_1 is context free
L_2 is context free
} $L_1 . L_2$ is context free

Language	Grammar
$L_1 = \{a^n b^n\}$	$S_1 \to aS_1b \mid \lambda$
$L_2 = \{ww^R\}$	$S_2 \to aS_2a \mid bS_2b \mid \lambda$
Concatenation	
$L = \{a^n b^n\} \cdot \{ww^R\}$	$S \to S_1 S_2$

3. Star closure of CFLs

The star closure of a context free language is a CFL.

$L = \{a^n b^n\}^*$ can be $S \to SS_1 \mid \lambda$

The result of the star closure of a CFL is a CFL because if G_1 is a CFG with G_1.language $= L_1$ & start symbol S_1 then we can define new CFG with a start symbol S by adding the production rule

$$S \to S_1 \mid SS_1 \mid \lambda$$

while retaining all other productions rules of the grammar.

for example $L = a^n b^n \mid n \geq 0$ we can obtain the grammar for a new kind of nesting language. as

$$S \to S_1 \mid SS_1 \mid \lambda$$

$$S \to aS_1b \mid \lambda$$

i.e $\{a^n b^n\}^*$ are $\{a^5 b^5\}^*$

$S \Rightarrow aS_1$

$\Rightarrow aS_1b$

$\Rightarrow aaS_1bb$

$\Rightarrow aaaS_1bbb$

$\Rightarrow aaabbb$

For example strings like a^3b^3 a^4b^4 which belong to the language, can be obtained using

$S \rightarrow SS_1$

$\Rightarrow S_1S_1$

$\Rightarrow aS_1bS_1$

$\Rightarrow aaS_1bbS_1$

$\Rightarrow aaaS_1bbbS_1$

$\Rightarrow aaabbbS_1$

$\Rightarrow aaabbbaS_1b$

$\Rightarrow aaabbbaaS_1bb$

$\Rightarrow aaabbbaabb$

i.e $(((\,)\,)\,)(\,(\,)\,)$ - flat nesting

$(\,(\,)(\,)\,)$ - proper nesting

$(\,(\,(\,)\,)\,)$ - simple nesting

4. Complement of CFLs

The complement of a CFL may not be context free language.

CFL is a subset of $a^* b^* c^*$ where either the no. of a's is not equal to the no. of b's or the no. of b's is not equal to no. of c's i.e

$n_a \neq n_b$ or $n_b \neq n_c$.

The complement of this language, is however $a^n b^n c^n$ which is not a CFL for this language we cannot able to construct a grammar or PDA.

5. Intersection of CFL's

CFL's are not closed under Intersection.

Ex: $L_1 = \{a^n b^n c^m\}$ $\quad L_2 = \{a^n b^m c^m\}$

Grammar $S \to AC$ $\quad S \to AB$

$A \to aAb \mid \lambda$ $\quad A \to aA \mid \lambda$

$C \to cC \mid \lambda$ $\quad B \to bBc \mid \lambda$

$L_1 \cap L_2$ i.e $= \{a^n b^n c^n\} = L$ is not a CFL.

6. Set Difference

CFL's are not closed under set difference. i.e $L_1 - L_2$ can be written set theoretically as

$$L_1 - L_2 = \overline{L_1 \cap L_2}$$

We know that intersection, complement are not closed, so difference is also not closed under CFL's

Decidable properties of CFLs

1. Is a given CFL is empty?

for example consider the CFG

$S \rightarrow aA \mid bB \mid C$

$A \rightarrow aA \mid Ab \mid bB$

$B \rightarrow bB \mid Ba \mid aA$

$C \rightarrow bA \mid aB \mid aSb \mid bSa$

This CFG represents an empty language because any sentential form derived by the S variable in this grammar contains only variables that can never be replaced by terminals.

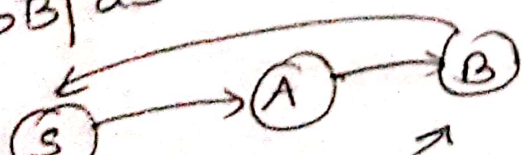
2. Is a given CFL finite or infinite?

for example the following grammar generates an infinite language because of the cyclic dependency of variables i.e A & B by S, A and B by A & B by A & B by B & S by B

$S \rightarrow aA \mid bB \mid C$

$A \rightarrow Ab \mid bB$

$B \rightarrow bB \mid aS$



3. Does a given string w belong to a CFL.

Using CYK algorithm we can check whether string $w \in L(G)$.

Undecidable properties of CFL.

1. Are two CFL's are same?
2. Is a given CFG G is ambiguous?

Problems on properties of CFL

1. S.T the Language

$L = \{a^n b^n \mid n \geq 0, n \neq 100\}$ is CF

we know that $L = \{a^n b^n \mid n \geq 0\}$ is context free & we also know that $L_1 = \{a^{100} b^{100}\}$ is regular.

$\bar{L_1} = \{(a+b)^*\} - \{a^{100} b^{100}\}$ is regular

Now $a^n b^n \cap \bar{L_1}$ is context free.

2. Show that set of all strings over $\{a, b, c\}$ that are either even palindromes or odd palindromes is context free.

The set of all even palindroms over the given alphabet is context free

$S \to aSa \mid bSb \mid cSc \mid \lambda$

The set of all odd palindromes is

also context free i.e

$$S \to aSa \mid bSb \mid cSc \mid a \mid b \mid c$$

We know that the union of two CFL is context free. Therefore the given language is context free

3. Show that the union of two deterministic context free languages is context free and non deterministic.

Soln: Let $L_1(G)$ and $L_2(G)$ be two context free languages generated by the grammar G_1 & G_2 respectively. Let S_1 and S_2 are the start symbols of G_1 & G_2 The union of the two languages is defined by

$$L = L_1 \cup L_2 = \{w \mid w \in L_1 \text{ & } w \in L_2\}$$

Let G be the grammar with S as new start state. We combine the production rules of both the grammar and add a new rule.

$$S \to S_1 \mid S_2$$

The grammar is still context free and therefore CFL are closed under union. The language is non deterministic as S can either go to S_1 or S_2.

4. Why are deterministic push down automata not equivalent to non deterministic push down automata?

Soln: Because NPDA's can have multiple stack configurations which cannot be simulated in a single stack, given the nature of the stack data structure.

5. Consider the statement, "If the grammar of a programming language is in GNF then its compiler can parse any program in linear time". Is this true or false? Justify your answer.

Soln: Not true. Because if it is in GNF, there may be multiple productions with the same LHS variable and the same terminal symbol on the RHS. This leads to nonlinear time.

Pumping lemma for CFL.

If a language L is context free then there exists some integer $p \geq 1$ called pumping length, such that every string w in L that has a length of p or more symbols i.e with $|w| \geq p$ can be written as

$$w = uvxyz$$

with substrings u, v, x, y and z, such that 1) $|vxy| \leq p$

2. $|vy| \geq 1$ and

3. $uv^n xy^n z \in L \quad \forall\ n \geq 0$

This is similar to pumping lemma for regular languages, but the pumping happens at two places.

L is CFL

So,

There exists a P,

for all strings w belonging to L,

there exists $w = uvxyz$ (with conditions)

for all n, $uv^n xy^n z \in L$

We cannot use this to prove that a language is CFL. Every CFL does follow this pumping lemma.

To prove that the language is CFL, we construct a PDA or develop a CFG.

The language $\{a^n b\, a^n b \mid n \geq 0\}$ is CFG as we can build a PDA for this language or we can construct a CFG.

We instead use the contra positive version of this lemma to show that the language is not CFL.

Assume that the language L is CFL

$\forall$ P,

$\exists$ a string $w \in L$

$\forall$ $w = uvxyz$ (with some rules)

$\exists$ n, $uv^n xy^n z \in L$

Adversary claims that a language L is CFL
I ask him for a magic number p. I have no control over it. This is universal.
Then I make up a string - thinking a lot. This is existential.
Then the adversary breaks the string into 5 parts. I have no control over it. He does follow the rules..
Then I select a value of n such that the pumped string is not in L - thus arriving at a contradiction.
That shows that L is not a CFL.

Observe that
I should select a good string to win this duel.
My string should be pumpable outside the language no matter how the adversary breaks the strings into parts.
It should work for every possible division of the string by the adversary.

1. S.T the language $a^n b^n c^n$ is not a context free.
Assume the language is context free and apply pumping lemma. write the language L in terms of pumping Constant p i.e $L = a^p b^p c^p$

We Know that vxy can be almost p symbols i.e

$|vxy| \leq p$

There are several cases possible

Case 1: vxy is made up only of a's

case 2: vxy is made up of some a's and some b's

case 3: vxy is made up only of b's

case 4: vxy is made up of some b's and some c's

case 5: vxy is made up only of c's

case	vxy	v and y : i	n_a	n_b	n_c	Remark
1	Only a's.	aa^* ≥ 2	↑	=	=	Too many a's
2	a's & b's	aa^*bb^* ≥ 2	↑	↑	=	Too few c's
3	Only b's	bb^* ≥ 2	=	↑	=	Too many b's
4	b's and c's	bb^*cc^* ≥ 2	=	↑	↑	Too few a's
5	Only c's	cc^* ≥ 2	=	=	↑	Too many c's

In any case, the string with the repeating pattern vxy, according to the Pumping lemma cannot be longer than p. As a result, it is not long enough to span all the three symbols a's b's and c's in the chosen string w. Pumping the string generates non members of the language in every case,

thereby contradicting our assumption that the language is context free.

2. Show that $L = \{ww \mid w \in \{a,b\}^*\}$ is not context free. Indicate the universal and existential quantifiers clearly.

Soln: For all p, the pumping length given by the adversary,

Let us choose the string $w = a^p b^p a^p b^p$.

For all possible divisions the adversary makes, we should be able to pump it out

$w = uvxyz \quad |vxy| \leq p \;\&\; |vy| \geq 1$

Case 1: vxy in a^p, on pumping, no of a's will be more in the left half

Case 2: vxy in b^p, on pumping, no of b's will be more in the left half

Case 3: vxy has no of a's & no of b's

On pumping number of a's & number of b's in first half will be more than number of a's & number of b's in second half.

So, this language is not context free.

3. Show that the following languages are context free. You can do this by writing a context free grammar or a PDA, or you can use the closure properties for CFL.

1. $L = a^n c b^n$. We can easily say the L is context free by constructing CFG

i.e $S \to aSb$

$S \to c$

2. $L = \{a^m b^n c^p d^q \quad n=q \text{ or } m \le p \text{ or } m+n = p+q\}$

Take a union of three simple CFL

$L_1 = a^m b^n c^p d^q \quad n = q$

$L_2 = a^m b^n c^p d^q \quad m \le p$ } left to reader

$L_3 = a^m b^n c^p d^q \quad m+n = p+q$

L_3 $S \to aSd \mid A \mid B$

$A \to aAc \mid C$

$B \to bBd \mid C$

$C \to bCc \mid \lambda$

For all productions, we generate one 'a' or 'b' on left side, and one 'c' or 'd' on the right side, which can guarantee that $m+n = p+q$. A cover cases that $m \ge q$ while B cover case $m \le q$ &

finally take a union of all three languages i.e $L_1 \cup L_2 \cup L_3$ i.e the grammar is $S \to S_1 | S_2 | S_3$ where S_1, S_2 & S_3 are start variable of L_1, L_2 & L_3 languages respectively.

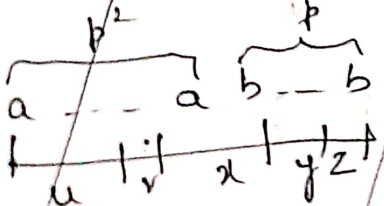
4. Show that the following language is not context free $L = \{a^i b^{i^2} \mid i > 0\}$

Soln: Assume $L = \{a^n b^{n^2} \mid n > 0\}$ is context free & apply pumping lemma & write L in terms of p

$$a^{p^2} b^p = uvxyz \qquad |vxy| \le p \qquad |vy| \ge 1$$

we examine all the possible locations of string vxy in $w = a^{p^2} b^p$.

case 1. v is in a^p & y is in b^p



$v = a^{k_1}$ $\quad y = b^{k_2}$ $\quad 1 \le k_1 + k_2 \le p$

Another case where $k_1 \neq 0$ & $k_2 \neq 0$

$v = a^{k_1}$ $\quad y = b^{k_2}$ $\quad 1 \le k_1 + k_2 \le p$

clearly $|w| \ge p$ & $w \in L$. Therefore $w = uvxyz$ & $|vy| \ge 1$ & $|vxy| \le p$ & $i \ge 0$ $\quad uv^i xy^i z \in L$. Let us consider two cases

case 1. $vxy = a^k$ i.e it contains no b.

Then $w' = uv^2xy^2z = a^{q+k}b^p$ & $k \geq 1$.
Therefore $w' \notin L$, a Contradiction

Case 2: $vy = a^j b^k$ & $k \geq 1$ i.e it contains atleast one b. Then consider $w' = uPoxy^0z$.
We have

$n_a(w) = q - j = p^2 - j$ & $1 \leq j \leq p-1$ &

$n_a(w) = p - k$ & $1 \leq k \leq p$ so

$(n_b(w))^2 = (p-k)^2 = p^2 - 2kp + k^2$

clearly $kp - j > k^2 - kp$ because

$kp - j \geq p - (p-1) = 1$ & $k^2 - kp = k(k-p) \leq 0$

Now it is easy to verify that $p^2 - j > p^2 - 2kp + k^2$. Therefore $w \notin L$, Contradiction to our assumption. So the Language is not context free.

5. S.T the $L = \{ xx^R yy^R zz^R, x,y,z \in \{a,b\}^* \}$ is context free

Soln: Since we know CFG for xx^R i.e even palindrome, & the Context free language are closed under Concatenation.

6 Suppose that L is context free & R is regular

(a) Is $L - R$ necessarily Context free

(b) Is $R - L$ necessarily Context free

Soln. (a) $L - R$ is context free

$L - R = L \cap \overline{R}$ (the complement of R)

$\overline{R}$ is regular (since the regular languages are closed under complement) & the intersection of a CFL & a RL is context free, so $L - R$ is context free

(b) $R - L$ need not be context free.

$R - L = R \cap \overline{L}$. But $\overline{L}$ may not be context free, since the context free languages are not closed under complement.