Content free Grammer

Grammer:

- Consist of set of rules to construct a sentences in a language.
- CFG is a method of language representation.
- CFG is used to generate all possible patte of strings in given finite language.
- It consist of set of production rules that generates the string of language.
- The concept of Content free grammer wa invented by the linguist Noam Chomsky in 1956.

- Definition:
  - A Content-free grammer, called CFG consists of four tuples.

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

- V = finite set of variables (non-terminals)
  - V always denoted by 'Capital' letter.
- T = finite set of terminals ($V \cap T = \phi$)
  - T always denoted by small letter and terminals help to generate string.

⇒ it represent regulars as well as non-regulars languages both.

- P= production rules (Substitution rules)
  - represent the recursive definition of a language
  - $S \rightarrow aS$ ⇒ one non-terminal → String of non-terminal
    - ↓ only one variable on left side.
    - ↳ can contain variable & tuple both
- S= Start variable.

Example:

$S \rightarrow aS1$

$S \rightarrow \varepsilon$

$V = S$, $T = aS1 | \varepsilon$

What is terminal?

⇒ The words that cannot be replaced by anything are called terminal.

What is non-terminal?

- finite set of variables.

⇒ Words that must be replaced by other thing we call them non-terminals.

What do you mean by passing the Sentence?

⇒ Determining how a sentence can be formed from the rules of grammer, is called passing the sentence.

What is Semantic & Syntax?

Semantic: Rules that involve the meaning of words called semantic.

Syntax: Rules that do not involve the meaning of words called syntax.

What is derivation & production?

Derivation: The sequence of applications of the rules that produces the finished string of terminals from the starting symbol is called derivation.

Productions:

The grammatical rules are often called productions. They all indicate possible substitutions.

Note: The derivation may or may not unique, by which we mean that by applying productions to the start symbol in two different ways may still produce the same finished product.

What do you mean by production

It is a finite set of rules or productions that represent the recursive definition of a language.

- Each production consists of:

=> A variable that is being defined by the production. The variable is often called the "head of the production"

=> The production symbol is →.

=> A string of zero or more terminals and variables. This string, called the body of the production, represents one way to form a string in the language of the variable of the head.

What is Content free language?

- A language generated by a CFG is called a content-free language.
- Content free language is the set of all strings of terminals that can be produced from the start symbol S using the productions or substitutions

RE to CFG Conversion

(1) RE = a+b

$S \rightarrow a|b$

OR $S \rightarrow a$

$S \rightarrow b$

OR $S \rightarrow X|Y$

$X \rightarrow a$

$Y \rightarrow b$

(2) RE = (a+b)(a+b)

$S \rightarrow AA$

$A \rightarrow a|b$

3) RE = (a+b)a+ab

$S \rightarrow XY$

$X \rightarrow JK$

$J \rightarrow a|b$

$K \rightarrow a$

$Y \rightarrow KB$

$B \rightarrow b$

4) RE = (a+b)(aa+bb)(a+b)

$S \rightarrow ABA$

$A \rightarrow a|b$

Note:

⇒ one capital letter cannot be equal to two small letters

Y ⇒ bb

A → YY | ZZ
Y → a
Z → b

CFG derivation:

Derivation is generation of string from grammer.

We can use CFG derivation to find out:
- which string is produced by grammer.
- Specific string is generated by grammer or not

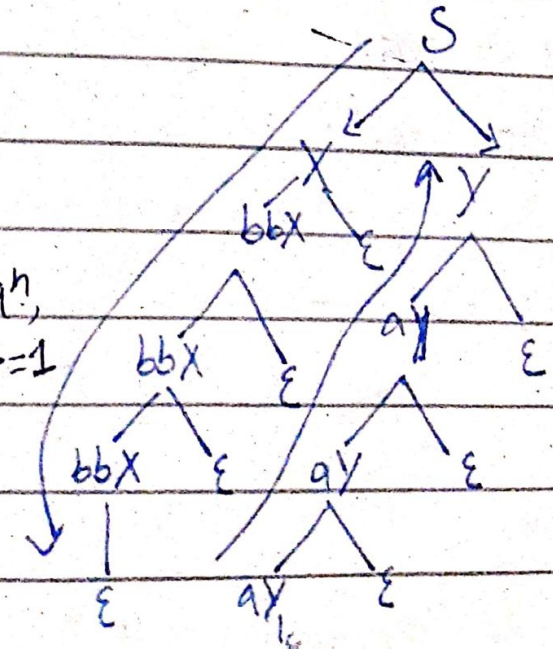
We can draw derivation tree to find out string produced by grammer.

Find out the string generated by given CFG.

S → XY
X → bbX | ε
Y → aY | ε

In general: $b^{2n}a^n$, n >= 1

String generated by given grammer will be:



Find out is "baab" valid string of given CFG?

$S \rightarrow ABA$

$A \rightarrow a|b$

$B \rightarrow YY|ZZ$

$Y \rightarrow a$

$Z \rightarrow b$

$S \rightarrow A\ B\ A$

$S \rightarrow a|b\ B\ A$

$S \rightarrow a|b\ YY|ZZ\ A$

$S \rightarrow a|b\ aa|bb\ A$

$S \rightarrow a|b\ aa|bb\ a|b$

Yes, baab is valid string for given number.

Construct CFG for the language having any no of a's over [a]

$\Sigma = \{a\}$

$L = \{\varepsilon, a, aa, \ldots\}$.

$R.E = a^*$

Production rule —

$S \rightarrow aS$

$S \rightarrow \varepsilon$

Language having any no of a's and b's

R: a+b

L = {ε, a, b, aa, ba, bb, ...}

S → aS | bS | ε

String containing at least 2 a's

RE: (a+b)*a(a+b)*a(a+b)*

{aa, aba, abba, baba, ...}

S → Aa A a A

A → aA | bA | ε

OR

S → bS | aX

X → bX | aY

Y → aY | bY | ε

One occurrence of 000

S → A000A

A → 0A | 1A | ε

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

L = {ab, aabb, ...}

S → aSb

S → ab

for n≥0

S → aSb | ε

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

S → aSbb

S → abb

Content free language for language containing odd no of b's

$S \to XY$
$X \to AbA$
$A \to aA \mid \varepsilon$
$Y \to AbAbA \mid \varepsilon$

$S \to ABA$
$A \to aA \mid \varepsilon$
$B \to b \mid AbAbAb$

$S \to aS \mid bX$
$X \to aX \mid b$
$X \to \varepsilon$

Content free Grammer for palindrom string of odd length.

$S \to aSa$
$S \to bSb$
$S \to a$
$S \to b$

$S \to aX \mid bX$
$X \to aS \mid bS \mid \varepsilon$

Pallindrom string of even length

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

Palindrom for either even or odd length.

$S \to aSa$
$S \to bSb$
$S \to a$
$S \to b$
$S \to \varepsilon$

$L = \{a^n b^{n+2}, n \geq 0\}$
$S \to aSb \mid bb$

$L = \{a^{2n} b^n, n \geq 0\}$
$S \to aaSb \mid \varepsilon$

$L = \{a^{2n} b^n, n \geq 1\}$
$S \to aaSb \mid aab$

$L = \{a^{2n+3} b^n, n \geq 0\}$
$S \to aaSb \mid aaa$

$L = a^m b^n, m > n, n \geq 0$
$S \to AS_1$
$A \to aA \mid a$
$S_1 \to aS_1b \mid \varepsilon$

$L = a^m b^n, m \geq n, n \geq 0$
$S \to AS$
$A \to aA \mid \varepsilon$
$S_1 \to aBb \mid \varepsilon$

$L = \{a^m b^n \mid m < n\}$
$S \to AB$
$A \to aSb \mid \varepsilon$
$B \to bB \mid b$

$a^m, n \geq 0$
$S \to AB$
$A \to aAb \mid \varepsilon$
$B \to aBb \mid \varepsilon$

$L = \{a^n b^m \mid n \neq m\}$

$S \rightarrow aSb \mid X \mid Y$

$X \rightarrow aX \mid a$

$Y \rightarrow bY \mid b$

Derivative Tree

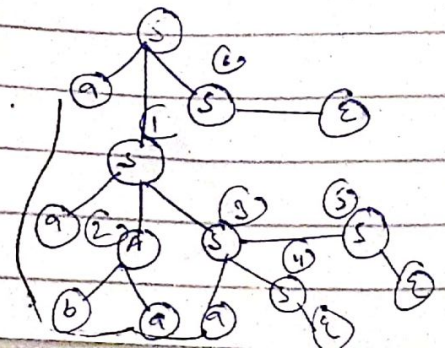
- A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a Context free grammer.

Left derivation tree:-

- A left derivation tree is obtained by applying production to the leftmost variable in each step.

e.g

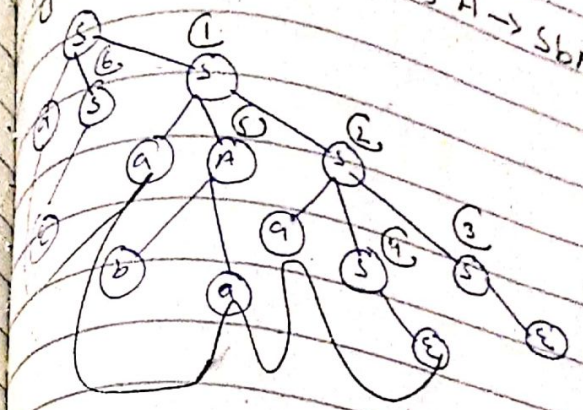
$S \rightarrow aAS \mid ass \mid \varepsilon$, $A \rightarrow SbA \mid ba$



Right derivation tree:

- A right derivation tree is obtained by applying production to the right most variable in each step.

e.g $S \rightarrow aAS \mid ass \mid \varepsilon$, $A \rightarrow SbA \mid ba$



Ambiguous Grammer

A grammer is said to be ambigious if there exists two or more derivation tree for a string w.

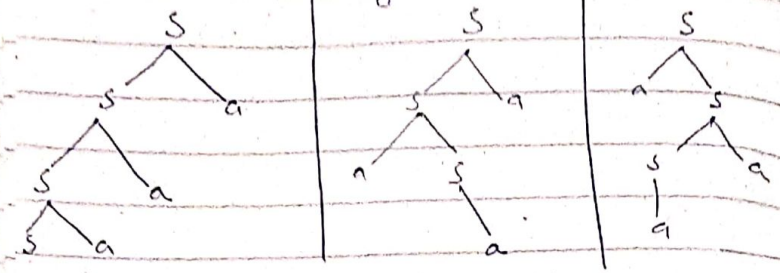
e.g $G = (\{S\}, \{a,b\}, S, S \rightarrow aSb \mid bSa \mid ss \mid a)$

Unambiguous Grammer:

- For unambiguous tree exist exactly one derivation for any word that belong to grammer.

Prove that G: S → Sa | as | a is ambiguous

Grammer:

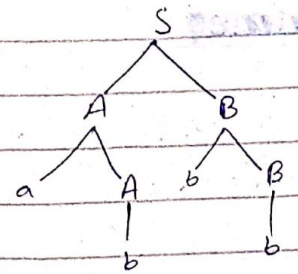


eg aaa

So, it is ambiguous grammer because there are multiple trees.

Prove that G1: S → AB is unambiguous
A → aA | b
B → bB | a
w: abba

grammer:



Types of CFG

- Recursive CFG
- Non-recursive CFG

Recursive CFG:

- generate infinite no of strings.

for example,

S → Sa
S → b

Non-recursive CFG:

- generates finite no of strings.

for example

S → Aa
S → b|c

Simplification of CFG

In CFG, sometimes all the production rules and symbols are not needed for the derivation of strings. Besides, there may also be some null productions and unit productions. Elimination of these productions and symbols is called simplification of CFG. Simplification consists of the following steps:

- Reduction of CFG
- Removal of unit productions
- Removal of NULL productions

Write a Context free Grammar for the following Languages over the alphabet $\Sigma = \{a, b\}$

(a) $L = \{ab^n aa^n \mid n > 0\} \Rightarrow ab^n a^{n+1}$

(b) $L = a^m b^m \mid m \neq n$ and $m, n \geq 0\}$

(c) Language of all words that do not have substring "bbb".

(d) All a's before first b.

(c)

$S \rightarrow bB \mid aS \mid bbB \mid \varepsilon$

$B \rightarrow aS \mid \varepsilon$

(a)

$S \rightarrow abXaa$

$X \rightarrow bXa \mid \varepsilon$

⟶ OR

$S \rightarrow aBaA$

$B \rightarrow bB \mid b$

$A \rightarrow aA \mid a$

$S \rightarrow aS_1a$

$S_1 \rightarrow bSa \mid \varepsilon$

(b)

$S \rightarrow aSb \mid \varepsilon$

(d)

$S \rightarrow aSb \mid A$

$B \rightarrow bB \mid \varepsilon$

$A \rightarrow aA \mid aB \mid \varepsilon$

Define null production.

A production of the form:

$N \rightarrow \lambda$

called null production

Example:

$S \rightarrow AB \mid \lambda$

Define unit production.

A production of the form:

one Nonterminal → one Nonterminal

is called a unit production.

=> it increased the cost of derivation in grammer

Example:

$A \rightarrow B$

In other words, if one variable produce only one other variable is called unit production.

Define useless production.

=> A symbol can be useless if it does not appear on the right hand side of production and it does not take part in the derivation of any string.

=> These unnecessary symbol or non terminal increase the length of the grammer

=> Also called inactive variable

=> A production which contains some inactive variables called useless production.

Examples

$S \rightarrow A$

$A \rightarrow aA / \varepsilon$

$B \rightarrow bA$

Here, B is useless symbol.

Q. Define CNF.

=> if a CFG has only productions of the form:

Nonterminal -> String of two nonterminals

or of the form:

Nonterminal -> one terminal

it is said to be in chomsky Normal form.

=> it is advance form of CFGs. Every Nonterminal produce exactly two Nonterminals or single terminal.

=> In CNF, there should not be:

- Useless production
- Null production
- Unit production.

e.g $S \rightarrow AB$

$B \rightarrow ba$

$A \Rightarrow a$, $A \rightarrow AB$

Find the CFG for the following language over the alphabet $\Sigma = \{a, b\}$

i) All words in which the letter b is never tripled.

$S \to BA$

$B \to bA \mid \varepsilon$

$A \to aA \mid aB \mid aBB \mid \varepsilon$

OR

$S \to aS \mid bB \mid \varepsilon$

$B \to aB \mid bA \mid \varepsilon$

$A \to aA \mid \varepsilon$

OR

$S \to aX \mid bX \mid \varepsilon$

$S \to aY \mid bY \mid \varepsilon$

$Y \to aY \mid \varepsilon$

ii) All words that have exactly two three b's.

Regular expression = $a^*(b+\lambda)a^*ba^*ba^*$

$S \to AB$

$A \to aA \mid \lambda$

$B \to bAbA \mid bAbAbA$

$S \to aS \mid bA$

$A \to aA \mid bB$

$B \to aB \mid bC \mid \varepsilon$

$C \to aC \mid \varepsilon$

iii) All words that have different first and last letters.

R.E = $a(a+b)^*b + b(a+b)^*a$

$S \to aXb \mid bXa$

$X \to aX \mid bX \mid \lambda$

All words that do not have the substring ab.

$S \to XY$

$X \to bX \mid \varepsilon$

$Y \to aY \mid \varepsilon$

All strings without the substring aaa.

$S \to aA \mid bS \mid aaA \mid \varepsilon$

$A \to bS \mid \varepsilon$

All words that do not have the substring baa.

$S \to AB$

$A \to aA \mid \varepsilon$

$B \to bB \mid baB \mid \varepsilon$

CFG for the following regular expression

(1) ab^*

$S \to aB$

$B \to bB \mid \lambda$

(2) a^*b^*

$S \to AB$

$A \to aA \mid \lambda$

$B \to bB \mid \lambda$

(3) $(baa + abb)^*$

$S \to XY$

$X \to baaX \mid \lambda$

$Y \to abbY \mid \lambda$

OR

$S \to baaS \mid abbS \mid \lambda$

$S \to bX \mid aY \mid \varepsilon$

$Y \to aaS$

$Y \to bbS$

All strings with exactly one a or one b.

$S \to XaX \mid YbY$

$X \to BaB$

$Y \to AbA$

$B \to bB \mid \varepsilon$

$A \to aA \mid \varepsilon$

$S \to XaX \mid YbY$

$X \to bX \mid \varepsilon$

$Y \to aY \mid \varepsilon$

The set of all strings of odd length.

$S \to aX \mid bX$

$X \to aS \mid bS \mid \varepsilon$

$S \to A \mid B$

$A \to CAC \mid a$

$B \to CBC \mid b$

$C \to a \mid b$

All strings with an odd number of a's & even number of b's.

$S \to aX \mid Y$

$X \to aaX \mid \varepsilon$

$Y \to bbY \mid \varepsilon$

$S \to aX \mid bX$

$X \to aS \mid bS \mid \varepsilon$

All string that end in b and have an even number of b's in total.

$S \to AbAbS \mid \varepsilon$

$A \to aA \mid \varepsilon$

OR

$S \to SS$

$S \to XbXb \mid \varepsilon$

$X \to aX \mid \varepsilon$

OR (~)

$S \to TbTb$

$T \to aT \mid bTb \mid \varepsilon$

The language defined by (aaa+b)*

S → aaaS | bS | ε

The language defined by (a+b)*(bbb+aaa)(a+b)*

S → XaaaX | XbbbX or | S → XYX
X → aX | bX | ε | Y → aaa | bbb
| X → aX | bX | ε

A CFG for all strings that end in b and have odd no of b's.

~~S → YYb~~

S → Tb

T → aT | bTb | ε

for String length at least 2

S → AA

A → a | b

B → aB | bB | ε

for String length at most 2 (a+b+λ)(a+b+λ)

S → AA

A → a | b | λ

All strings start with a and end with b

S → aAb

A → aA | bA | ε

Having at least one occurrence of substring aa

S → AaaA

A → aA | bA | ε

CFG to generate the language $0^n 1^m 0^{n+m}$

S → 0S0 | ε

A → 0A | 1

B → 1B | ε

Context free grammar to generate the language (0+1)0*(11)*

S → ABC

A → 0 | 1

B → 0B | ε

C → 11C | ε

L = {$a^n b^n$: n is even}

S → λ | aaSbb

L = {$a^n b^n$: n is odd}

S → ab | aaSbb

L = {$a^n b^n$: n is multiple of 3}

S → ε | aaaSbbb

$L = \{ a^n b^n : n \text{ is not multiple of } 3 \}$

$S \rightarrow aa \mid aabb \mid aaaSbbb$

$L = \{ a^n b^n c^n \}$

$S \rightarrow aBC \mid aSBC$

$CB \rightarrow BC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

$C \rightarrow XY$

$X \rightarrow aXb \mid ab$

$Y \rightarrow cY \mid c$

$L = \{ a^m b^n c^p d^q : m+n = p+q \}$

$R = \{ S \rightarrow aSd, S \rightarrow T, S \rightarrow U, T \rightarrow aTC, T \rightarrow V, U \rightarrow bUd, U \rightarrow V, V \rightarrow bVc, V \rightarrow \varepsilon \}$

$L = \{ a^i b^j c^k : i \neq j+k \}$

$= \{ a^i b^j c^k : i < j+k \} \cup \{ a^i b^j c^k : i > j+k \}$

$S \rightarrow aSc \mid Sc \mid X$

$X \rightarrow aXb \mid Xb \mid \varepsilon$

$L = \{ a^n b^{2n} \mid n \geq 0 \}$

$S \rightarrow aSbb \mid \varepsilon$

$L = \{ a^n b^{4n} \mid n \geq 1 \}$

$S \rightarrow aSbb \mid abb$

$L = \{ a^i b^j c^k : i \neq j \}$

$S \rightarrow A_{i<j} C \mid A_{i>j}$

$A_{i<j} \rightarrow aA_{i<j}b \mid aBb$

$B \rightarrow bB \mid \varepsilon$

$C \rightarrow cC \mid \varepsilon$

$A_{i>j} \rightarrow aA_{i>j} \mid aR$

Simplification of CFGs

=> The reason to make it more efficient and compiler friendly

Removal of Null production:

=> No change will be in grammer, after removal it will generate that grammer

Steps:

=> Find all Nullable variables.

=> Create two versions of all production containing nullable variables on their R.H.S one without and one with Nullable variable. e.g $A \to aA$ [$A \to aA$, $A \to a$]

=> Remove ε productions.

=> production whose R.H.S does not have any nullable variables are included directly

Note:

=> it is not possible to remove all ε, when it is part of the language of grammer.

Example:

$S \to a \mid Xb \mid aXa$

$X \to Y \mid \Lambda$

$Y \to b \mid X$

=> $S \to a \mid Xb \mid aXa \mid b \mid aa$

$X \to Y$

$Y \to b \mid X$

Ex #

$S \to XY$

$X \to Zb$

$Y \to bW$

$Z \to AB$

$W \to Z$

$A \to aA \mid bA \mid \varepsilon$

$B \to Ba \mid Bb \mid \varepsilon$

Step 1:

Nullable = { A, B, Z, W }

$S \to XY$

$X \to Zb$

$Y \to bW$

$Z \to AB \mid A \mid B$

$W \to Z$

$A \to aA \mid bA \mid a \mid b$

$B \to Ba \mid Bb \mid a \mid b$

Ex #

$S \to ABAC$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

$C \to c$

Ans:

$S \to ABAC \mid ABC \mid AAC \mid BAC \mid AC \mid BC \mid c$

$A \to aA \mid a$

$B \to bB \mid b$

$C \to c$

Ex #

$S \to AbB$

$A \to a \mid \varepsilon$

$B \to b \mid \varepsilon$

Ans:

$S \to AbB \mid Ab \mid bB \mid b$

$A \to a$

$B \to b$

Algorithm:

while (there exist a unit production $A \to B$)

{

* Select a unit Production $A \to B$ such that there exist a production $B \to \alpha$, where α is terminal.

* { for (every non-unit production $B \to \alpha$)

Add production $A \to \alpha$ to the grammar

* Eliminate $A \to B$ from the grammar

}

Removal of Unit Production:

eg.

S → Aa | B | c
B → A | bb
A → a | bc | B

S → Aa | bb | bc | a | c
B → a | bb | bc
A → a | bc | bb

e.g

S → A | bb
A → B | b
B → S | a

UP

S → A
A → B
B → S

Decent folks

S → bb
A → b
B → a

S → A
S → A → B
A → B
A → B → S
B → S → A

S → b
S → a
A → a
A → bb
B | bb

S → bb | b | a
A → a | b | bb
B → bb | b | a

e.g S → aX | Yb
X → S
Y → bY | b

S → aX | bbb
X → bbb | abbb
Y → bY | b

S → Yb
S → bbb
Y → bb
X → bbb
S → abbb

e.g

S → AA
A → B | BB
B → abB | b | bb

S → AA
A → b | bb | abb | abbb | BB
B → abB | b | bb

B → b | bb | abb | abbb

(8) S → AB
A → B
B → aB | Bb | Λ

Firstly remove Λ Production

S → AB | A
A → λ | B
B → aB | Bb | a | b

Now, remove unit production

S → AB | aa | bb | ab | ε B → aa | ab | ab | bb
A → aa | bb | ab | ε
B → aB | Bb | a | b

Q. What is CNF?

Definition:-

⇒ CNF is a method used to convert CFG into normal form.

⇒ In CNF: every production rule has two forms:

(1) Every Non-terminal represents exactly two non-terminal variables. e.g. A → XY

(2) Every Non-terminal represents exactly one terminal.

e.g. X → a

⇒ CNF used to check the occurrance of specific string in CFG.

⇒ CNF is advanced form of CNF and used to generate formal languages.

⇒ CNF has some important properties that make it useful for algorithms and simplified processing and processing of CFL, such as efficient parsing algorithms and simplified grammatical structure.

⇒ However, not all CFGs can be converted to CNF, as some may generate language that cannot be generated by a CNF grammar.

Example:

(i) S → aSa | SSa | a

S → aSa

S → SSa

S → a

⇒ S → TaTa

Ta → a

Ts → Sa

S → STs

S → a

(ii) S → aXX

X → aS | bS | a

S → aXX

X → aS

Now, in CNF

$S \to TaX$

$Ta \to aX$

$S \to Tx X$

$Tx \to TaX$

$X \to TaS$

$X \to TbS$

$X \to a$

$Tb \to b$

$Ta \to a$

(iii)

$S \to X$

$X \to Y$

$Y \to Z$

$Z \to aa$

=> firstly remove unit production

=> $S \to aa$

$X \to aa$

$Y \to aa$

$Z \to aa$

=> Now remove useless production

$S \to aa$

=> Now convert in CNF

$S \to TaTa$

$Ta \to a$

(iv) $S \to SS \mid A$

$A \to SS \mid AS \mid a$

$S \to SS$

$S \to A$

$A \to SS$

$A \to AS$

$A \to a$

firstly remove unit production

$S \to SS$

$S \to a$

$A \to SS$

$A \to AS$

Now, convert in CNF,

it is already in CNF.

(v) $S \to aA \mid bB \mid ab$

$A \to B \mid aA$

$B \to bB \mid \varepsilon$

=> Firstly remove null production.

$S \to aA \mid bB \mid ab \mid a$

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

$B \to bB \mid b$

=> Now, remove unit production

$S \to aA \mid bB \mid ab \mid a$

$A \to \varepsilon \mid aA \mid a$

$A \to B$

$B \to bB \mid b$

$A \to a$

$S \to SS \mid a \mid AS$

$A \to a \mid AS \mid SS \mid A$

=> S → aA | bB | ab | a
A → ε | aA | a
A → b | bb | a or bB
B → bB | b

Now Convert CFG to CNF:-

Introducing $T_a \to a$, $T_b \to b$

So,

S → T_aA | T_bB | $T_a T_b$ | b | a
A → b | T_bB | T_aA | ε
B → T_bB | b
T_a → a
T_b → b

Example : Convert the given CFG to CNF.

S → a | aA | B
A → aBB | ε
B → Aa | b

=> Firstly remove null production.

S → a | aA | B
A → aBB
B → Aa | a | b

=> Now, remove unit production:-

S → a | aA | b |
B → Aa | a, b
A → aBB
S → a | aA | b | Aa

aBBA
aa ab
aba | aab

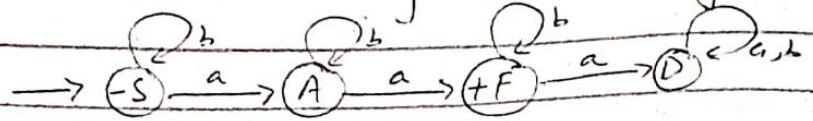
Now, eliminate useless production

=> Now, Convert in CNF:-

S → a
S → b
S → T_aA
B → AT_a | a | b
A → $T_a T_b$
S → a | T_aA | b | AT_a
T_a → a
T_b → BB

Q. How to generate CFG from FSA.

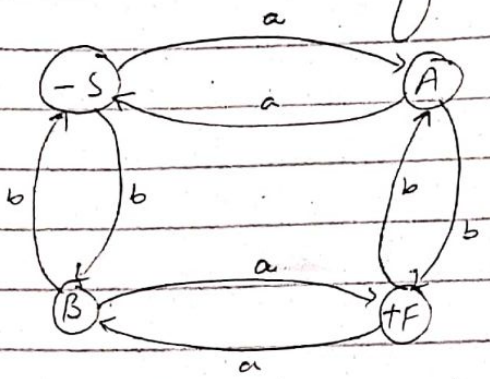
Ex: write CFG for the FSA given below:



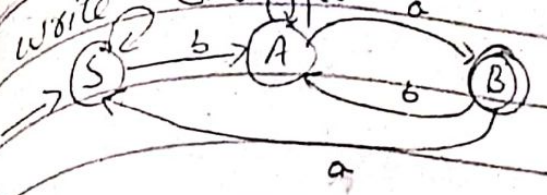
S → bS
S → aA
A → bA
A → aF
F → bF | ε

=> Note, no need to generate dead state.

Q. write CFG for the FSA given:

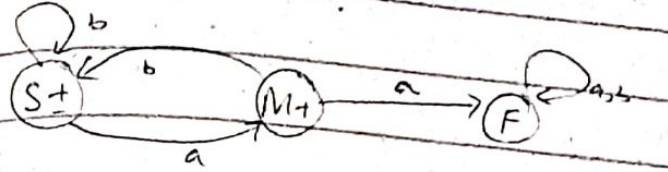


S → aA | bB
A → aS | bF
B → bS | aF
F → aB | bA | ε



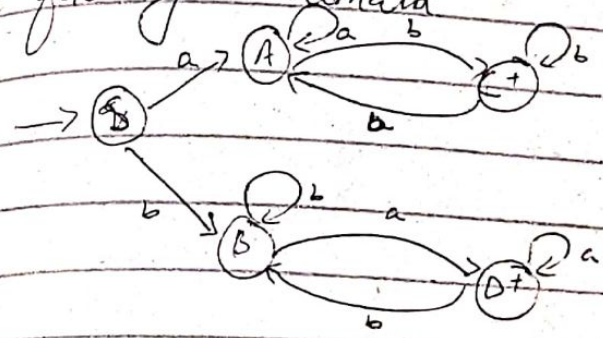
S → aS | bA
A → bA | aB
B → bA | aS | ε

Q.



S → aM | bS | ε
M → bS | aF | ε
F → aF | bF

Q. Determine the CFG, corresponding to the following automata



S → aA | bB
A → aA | bC | ε
C → bC | aA | ε

B → bB | aD
D → aD | bB | ε

Q: what is the concept of the union of FAs?

=> it is a fundamental concept in ToA involves combining two or more finite automata into a single automaton that recognize the union of the languages recognized by each individual automaton

=> it involves creating a new start state and connecting it to the start states of the two automata being combined using ε transitions. The accepting states of the resulting automata are the union of the accepting states of the original automata

=> it is denoted by $L_1 \cup L_2$.

Suppose we have two finite automata A_1 and A_2, recognizing languages L_1 and L_2 respectively, then $A_1 \cup A_2$ is another automata that recognize $L_1 \cup L_2$

=> Union operation is the basic building block for constructing more complex automata and plays imp role in many area of CS, including PL, Compilers & SE

Q How one can create a RE of a particular language?

=> To create RE for a particular language there are several steps to be followed; it requires a clear understanding of the language that needs to be expressed as an RE. It means defining the set of strings that belong to the language, as well as the basic building blocks and operators used in RE like union, Kleene star and constructing the RE and verify the language expression by testing the RE against a set of strings that belong to the language as well as the set of strings that do not belong to the language.

Q. What is equal RE?

=> Two regular expressions are said to be equal if they represent the same language means they accept the same set of strings.

=> Determining if two regular expressions are equal can be challenging, especially for complex expressions, and may require converting the expression to a different form, such as finite automaton or regular grammar to compare them.

for example:- $r_1 = (a+b)^*$ $r_2 = (a+b)^*a(a+b)^*$

The regular expressions $(a|b)^*$ and $(a|b)^*$ are equal because they both represent the language of all strings composed of 0 or more occurrences of the letters "a" and "b".

Q. What are the basic rules to build FA?

o The basic rules for building a finite automaton including; identifying the states, defining the transition states or functions, drawing the FA diagram, and testing the FA.

o The transition function specifies how the automaton moves from one state to another based on the input symbol it receives.

o The FA can be tested using a set of input strings to determine if it accepts the desired language.

o These rules apply to different types of FA, including DFA & NFA

Q. what is difference b/w palindrom & Reverse function?

Pallindrom:

⇒ A pallindrome is a string that reads the same forwards and backwards.

⇒ Palindrom is a property of a specific string.

⇒ e.g mom and madam, or aia, babab.

Reverse function:

⇒ A reverse function is an operation that takes a sequence of characters and returns the sequence with order of the characters reversed.

⇒ A reverse function is a general operation that can be applied to any string to produce a new string with reversed order.

eg:- "hello" reverse of this string is "olleh".

⇒ The reverse of a palindrome is also a palindrom.

why we need TGs when we have FAs?

TGs is another representation of FA: both are equivalent in terms of their computational power. TGs are just another of FAs TGs can provide a more intuitive and visual representation of the automaton and analyzing the behavior of automata ~~it is used to~~ it can read more than one character at a time.

Q: if a language can be expressed in the form of FA then why it is needed to use NFA?

⇒ while a language can be expressed using a FA, sometimes it is easier or more efficient to use NFA instead of FA because

- its simplicity, transition to multiple states for a given i/p symbols
- flexibility allows ε-transition
- Efficiency used in algorithms that require searching through a large number of potential solutions.

Q. what is regular language? give example.

- A regular language is a formal language that can be expressed using a regular expression or can be recognized by a finite automaton.
- A RL is a subset of all possible strings over a given alphabet, and it can be characterized by a finite set of rules or patterns that describe the structure of the language.

e.g.

The set of all strings that contains the substring "101"

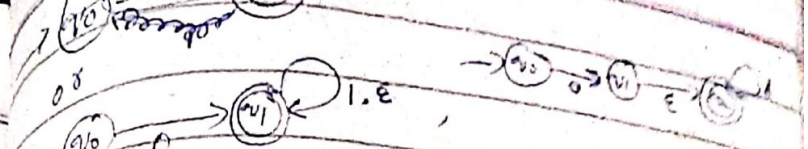
Q. which language {00, 01, 10, 11} presents?

⇒ The language {00, 01, 10, 11} consists of all possible strings of length 2 over the binary alphabet {0, 1}. Each string

⇒ Therefore, the language can be defined using the RE

$$(00)|(01)|(10)|(11)$$

Q. Give an example of an NFA except NFA $\{0\}\{1\}^*$



⇒ it generate language that starts with 1 and followed by any 1's

Q. what is non regular language?

- non-regular language is a formal language that cannot be described using a RE or recognized by a FA.
- They demonstrate that there are some language that are too complex to be recognized by FA or RE.

e.g.

- The language of all strings of the form $a^n b^n$, where n is the integer.
- The language of all palindromes over the binary alphabet

Q. what is minimal FA?

- A minimal FA is a finite automata that has the smallest number of states among all finite automata that accept the same language.
- In other words, FA that recognizes a given language using fewest possible no of states

=> The process of finding a minimal FA by further eliminating any redundant or unreachable states from the original FA.

=> This is done by merging states that are equivalent w.r.t their ability to recognize the language.

=> It can be useful for optimizing the performance of algorithms that use the FA to process the strings in the language.

Q. what does mean the LANGUAE is CLOSED?

=> In the context of formal language, a language is said to be closed under a particular operation if applying that operation to any string in the language results in a new string that is also in the language.

example:

if we have two languages L1 and L2 that are closed under concatenation, then the the language L = L1·L2. means if we concatenate any string in L1 with any string in L2, the resulting string L is also closed under concatenation.

Q. what is non-determinism & Determinism and what is the difference btw them?

Determinism and Non-Determinism are different modes of operation of automata. Determinism & non-determinism refer to how different types of automata

2. what is difference btw (a,b) & (a|b)? AND

=> (a,b) is an ordered pair of two symbols, where a and b are two distinct symbols from a given alphabet.

=> it can be used in formal language to describe a string containing a followed by b such as the string ab. different from concatenation.

=> (a+b) represents the language that contains either a or b or both

Q. Complement of a regular language is always regular. state either either true or false along with reason.

The statement :... is true.

Since, it can be recognized by a DFA constructed from the DFA that recognizes the original language.

⇒ This can be showing that if a language L is regular, then its complement L' is also regular.

⇒ As we know that a language is regular iff it is recognized by a DFA.

So let L be recognized by some DFA $M = (Q, \Sigma, \delta, q_0, F)$.

⇒ we construct a new DFA, $M' = (Q, \Sigma, \delta, q_0, F)$, where δ' is the same as δ except that all the accepting states in M become non-accepting states in M', and all the non-accepting states in M become accepting states in M'. Formally, $F' = Q \setminus F$.

⇒ Consider the language recognized by M'. if a string is accepted by M', it means it is not accepted by M.

Q. Every subset of a regular language is always regular. State either this statement is true or false. explain with example.

The statement : ... is false

⇒ A counter example is provided by the regular language $L = \{a^n b^n \mid n \geq 0\}$ and its subset $S = \{a^n b^n \mid n$ is a Prime number$\}$.

it can be shown that S is not a regular language using the pumping lemma or regular languages.

Therefore, the statement is false

while drawing RE for $0^2 1^2$

$RE = (00)(11)$

⇒ the set of all strings consisting of 0 repeated twice followed by 1's repeated twice

Q. what is the difference between Nullable and Null production?

⇒ A nullable is a non-terminal symbol that can derive the empty string, either directly or indirectly.

e.g

$S \to AB$

$A \to \varepsilon$

⇒ A is nullable symbol because it can derive ε directly.

e.g

$B \to CD$ and C and D are both nullable symbols, then B is also nullable because it can derive ε indirectly through C and D.

Null production:

A null production is a production that allows a non-terminal to derive the empty string directly.

e.g

$S \rightarrow AB$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

then both A and B are nullable symbols, but only A is associated with null production
=> it is production that has the empty string on its right-hand side.

Eg:iii

CNF

Q#: S → ABA
A → aA | ε
B → bB | ε

Step 1: Remove A → ε production

S → ABA | BA | AB
A → aA | a
B → bB | ε

Now, remove B → ε production

S → ABA | BA | AB | A | B | AA
A → aA | a
B → bB | b

Step 2:

Remove unit production

S → A	⇒	S → aA | a
S → B	⇒	S → bB | b
S → ABA | BA | AB | AA		S → ABA | BA | AB | AA
A → aA | a		A → aA | a
B → bB | b		B → bB | b

Step3:

S → aA
S → a
S → bB
S → b
S → ABA
S → BA
S → AB
S → AA
A → aA
A → a
B → bB
B → b.

New grammer in CNF

S → TaA
Ta → a
S → a
S → TbB
Tb → b
S → b
R1 → AB
S → R1A
S → BA | AB | AA
A → TaA | a
B → TbB | b

Simplify the grammer.

S → aAbB
A → aA | a
B → bB | b

Step 1

⇒ It is already simplified because no:

- ε - Production
- Unit - production
- Useless production

Step 2:

	Simplified new grammer
⇒ S → CaA CbB	⇒ S → Ca D1
A → CaA | a	D1 → AD2
B → CbB | b	D2 → CbB
Ca → a	A → CaA | a
Cb → b	B → CbB | b
	Ca → a
	Cb → b

Q. Convert given CFG in CNF.

$S \rightarrow 1B \mid 0A$

$B \rightarrow 1BB \mid 0S \mid 1$

$A \rightarrow 0AA \mid 1$

$S \rightarrow T_1B \mid T_0A$

$T_1 \rightarrow 1$

$T_0 \rightarrow 0$

$B \rightarrow T_1X \mid T_0S \mid 1$

$X \rightarrow BB$

$A \rightarrow T_0Y \mid 1$

$Y \rightarrow AA$

Q. Convert given CFG in CNF.

$S \rightarrow asa \mid bsb \mid a \mid b \mid aa \mid bb$

Step:

$S \rightarrow T_aST_a \mid T_bST_b \mid a \mid b \mid T_aT_a \mid T_bT_b$

$T_a \rightarrow a$

$T_b \rightarrow b$

Still it is not in CNF.

Step:

$S \rightarrow T_aR_1 \mid T_bR_2 \mid a \mid b \mid T_aT_a \mid T_bT_b$

$R_1 \rightarrow ST_a$

$R_2 \rightarrow ST_b$

Convert CFG to CNF.

$S \rightarrow aSa \mid SSa \mid a$

$S \rightarrow T_aST_a \mid SST_a \mid a$

$T_a \rightarrow a$

Here, Still Condition not satisfied of CNF.

$S \rightarrow T_aR \mid SR \mid a$

$T_a \rightarrow a$

$T_b \rightarrow b$

$R \rightarrow ST_a$

Q. Convert CFG to CNF.

$S \rightarrow aXX$

$X \rightarrow aS \mid bS \mid a$

$S \rightarrow RX$

$R \rightarrow T_aX$

$T_a \rightarrow a$

$X \rightarrow T_aS \mid T_bS \mid a$

$T_b \rightarrow b$

Q. Convert CFG to CNF.

S → ABABAB

A → a | ε

B → ε

Step 1:

Removing A → ε production.

S → ABABAB | BABAB | ABBAB | ABABB | BBAB | BABB | BBB | ABBB |

A → a |

B → ε | b

Removing B → ε production.

S → ABABAB | AABAB | ABAAB | ABABA | AAAB | AABA | AAAB | ABAA | AAA | ABAB | ABBA | AA | ABA | AAB | BAB | AB | A | ABB | BAB | BA | BB | B | BBB | BABB | ABBB | BBAB | ABABB | ABBAB | BABAB | ε

Now,

Step 2:

S → $R_1R_1R_1$ | AR_1R_1 | R_1AR_1 | R_1R_1A | R_2R_1 | R_1R_2 | AA | R_1A | AR_1 | R_3B | AB | a | R_1B | R_3B | BA | b | R_2B | R_3R_4 | R_1R_4 | R_4R_1 | R_1R_1B | R_1R_3B | R_2R_3B

R_1 → AB

R_2 → AA

R_3 → BA

A → a

B → b

R_4 → BB

⇒ Still it is not in CNF

Step:

S → DR_1 | AD_1 | D_2R_1 | D_1A | R_2R_1 | R_2R_1 | R_2R_1 | R_1R_2 | R_2A | R_1R_1 | R_1R_3 | AA | R_1A | AR_1 | R_3B | AB | R_1B | R_3B | BA | R_4B | R_3R_4 | R_1R_4 | R_4R_1 | D_1B | D_3B | D_4B | a | b

D_1 → R_1R_1

D_2 → R_1A

D_3 → R_1R_3

D_4 → R_3R_3

Now, it is in CNF form.

R_1 → AB

R_2 → AA

R_3 → BA

R_4 → BB

A → a

B → b

#14) $S \to SaS \mid SaSbS \mid$

Step 1: Remove Null productions

$S \to SaS \mid aS \mid Sa \mid SasbS \mid aSbS \mid abS \mid SabS \mid abS \mid Sab \mid ab \mid a \mid SaSb \mid aSaS \mid SaS \mid Sa \mid SbS \mid b \mid Sb \mid bS \mid bSaS \mid SbaS \mid SbSa \mid baS \mid bSa \mid ba \mid Sba \mid$

$\mid Sba$

Step 2:

Still it is not in CNF.

$S \to STaS \mid TaS \mid STa \mid STaSTbS \mid TaSTbS \mid TaTbS \mid STaTbS \mid TaTbS \mid STaTb \mid TaTb \mid a \mid STaSTb \mid TaSTb \mid STaS \mid STa \mid STbS \mid b \mid STb \mid TbS \mid TbSTaS \mid STbTaS \mid STbSTa \mid TbTaS \mid TbSTa \mid TbTa \mid STbTa \mid STbSTa \mid STbTa$

Still it is not in CNF.

$S \to SR_1 \mid TaS \mid STa \mid SR_3 R_2 \mid R_1 R_2 \mid TaR_2 \mid D_1 R_2 \mid TaR_2 \mid D_1 Tb \mid TaTb \mid a \mid D_1 D_2 \mid R_1 Tb \mid D_1 S \mid STa \mid SD_2 \mid b \mid STb \mid TbS \mid R_2 R_1 \mid R_2 R_1 \mid STa \mid D_2 S \mid b \mid STb \mid TbS \mid R_2 R_1 \mid D_2 D_1 \mid D_2 D_1 \mid$

$R_1 \to TaS$, $D_1 \to STa$, $Ta \to a$ $TbR_1 \mid TbD_1$

$R_2 \to TbS$, $D_2 \to STb$, $Tb \to b$ $TbTa \mid TbT$

Q. $S \to bA \mid aB$

$A \to bAA \mid aS \mid a$

$B \to aBB \mid bS \mid b$

$S \to TaA \mid TaB$

$Ta \to a$

$A \to TbR_1 \mid TaS \mid a$

$R_1 \to AA$

$B \to TaR_2 \mid TbS \mid b$

$R_2 \to BB$

$Tb \to b$

Q. $S \to AAAAS$

$S \to AAAA$

$A \to a$

$S \to AR_1$

$R_1 \to AR_2$

$R_2 \to AR_3$

$R_3 \to AS$

$S \to AR_4$

$R_4 \to AR_5$

$R_5 \to AA$

$A \to a$

Q. Eliminate the unit produ

S → ASA | aB

A → B | S

B → b | ε

Step 1:

Remove Epsilon production:

S → ASA | aB | a

B → B | ε

B → b

Step 2:

Remove unit production:

S → ASA | aB

A → b | ASA | aB | ε

B → b

Step 3:

S → AR | TaB

R → SA

Ta → a

A → b | AR | TaB | ε

B → b

Q. Eliminate unit productions from the CFG:

S → Aa | B

B → A | bb

A → a | bc | B

S → B

B → A

A → B

S → Aa | a | bc | bb

B → a | bc | bb

A → a | bc | bb

Q E → E + T | T

T → T * F | F

F → (E) | a

Note: Everything other than capital letter is always terminal.

E → E + T | T * F | (E) | a

T → T * F | (E) | a

F → (E) | a

Q. Convert the following CFG to CNF

S → ASB

A → aAS | a | null

B → BbS | A | bb

Step 1:

Remove S → ε productions

S → ASB | AB

A → aAS | a | aA

B → BbS | A | bb

Step 2:

Remove unit production

S → ASB | AB

S → aAS | a | aA

B → BbS | a | aA | aAS | bb

Step 3:

S → RB | AB

R → AS

S → TaR | TaA

Ta → a

B → DS | a | TaA

Ta → a

D → TaA

Q # CFG to CNF.

S → abAB

A → bAB | ε

B → BAa | A | ε

Step 1:

Remove null production.

S → abAB | abB	S → abAB | abB | abA
A → bAB | bB	A → bAB | bB | bA | b
B → BAa | Ba | ε	B → BAa | Aa | Ba | a

Step 2:

Remove unit production OR Remove A → a

S → abAB | abB | abA | ab

A → bAB | bB | bA | b

B → BAa | Aa | Ba | a | bAB | bB | bA | b

Step 3:

S → R1R2 | R1B | R1A | TaTb

R1 → TaTb

R2 → AB

A → TbR2 | TbB | TbA | b

Tb → b

Ta → a

B → BR3 | ATa | BTa | a | TbR2 | TbB | TbA | b

R3 → ATa

Q. Convert into CNF.

S → AACD

A → aAb | ε

C → ac | a

D → aDa | bDb | ε

Step 1:

Remove null production.

Here A and D are nullable variable.

S → AACD | ACD

A → aAb | ab

C → aC | a

D → aDa | bDb | ε

Now, D → ε eliminate

S → AACD | ACD | AAC | AC | AD | C

A → aAb | ab

C → ac | a

D → aDa | aa | bDb | bb

Step 2:

Remove unit production

S → AACD | ACD | AAC | AC | CD

S → aC | a

C → ac | a

D → aDa | aa | bDb | bb

Step 3:-

S → R_1R_2 | AB | BC | AC | CD | T_aC | a

R_1 → AA

R_2 → CD

T_a → a

C → T_aC | a

D → R_3T_a | T_aT_a | R_4T_b | T_bT_b

R_3 → T_aD

R_4 → T_bD

T_b → b

Q. Remove the null production from the following grammer.

S → ABAC

A → aA | ε

S → ABAC | BAC | ABC | BC |

A → aA

Q. Convert CFG to CNF.

S → ABA

A → aAa | aB | ε

B → bBb | bA | ε

Step 1:

Remove null production.

Here, A and B are nullable variables

Now, remove $A \to \varepsilon$

$S \to ABA \mid AB \mid BA \mid B$

$A \to aAa \mid aB \mid aa$

$B \to bBb \mid bA \mid \varepsilon \mid b$

Now, remove $B \to \varepsilon$

$S \to ABA \mid AB \mid BA \mid B \mid AA \mid A \mid B \mid \varepsilon$

$A \to aAa \mid aB \mid aa \mid a$

$B \to bBb \mid bA \mid bb \mid b$

Step 2:-

Now remove unit production.

$S \to ABA \mid AB \mid BA \mid B \mid AA \mid A \mid aAa \mid aB \mid aa \mid a \mid bBb \mid bA \mid bb \mid b$

$A \to aAa \mid aB \mid aa \mid a$

$B \to bBb \mid bA \mid bb \mid b$

Step 3:-

$S \to R_1A \mid AB \mid BA \mid b \mid AA \mid a \mid R_2A \mid XB \mid XX \mid a \mid R_3X \mid YA \mid YY \mid b$

$R_1 \to AB$

$X \to b$

$Y \to a$

$R_2 \to XA$

$R_3 \to YB$

$A \to R_2X \mid XB \mid XX \mid a$

$B \to R_3Y \mid YA \mid YY \mid b$

Q. Convert given CFG to CNF.

$S \to aAbB$

$A \to aA \mid a$

$B \to bB \mid b$

$S \to R_1R_2$

$R_1 \to C_aA$

$R_2 \to C_bB$

$C_a \to a$

$C_b \to b$

$A \to C_aA \mid a$

$B \to C_bB \mid b$

Note:

- if S appears in RHS, then add a new production $S' \to S$
- S is start symbol, $S \to aS$

$S' \to S$

$S \to aS$

Example:

$S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \varepsilon$

Step 1:

$S' \to S$

$S \to ASA \mid aB$, $A \to B \mid S$, $B \to b \mid \varepsilon$

Q. What is type-2 Grammer?

⇒ Also known as Context free grammer, which will generate CFL that will be accepted by P.D.A (NPL default case).

⇒ if there is a production, form

$$\alpha \rightarrow \beta$$

$$\alpha \in V_n$$

$$\beta \in \{\Sigma \cup V_n\}^*$$

⇒ Examplee

ALGOL, PASCAL