# 4 Context-Free Language and Pushdown Automata



## 4.1 CONTEXT-FREE GRAMMAR

- Context-free grammars are a more powerful method of describing languages.
- The productions in a regular grammar are restricted in two ways:
  The left side must be a single variable, while the right-hand side of the production has a constraint as we know for RLG: $A \rightarrow xB|x$ and for LLG: $A \rightarrow Bx|x$ where $A, B \in V$ and $x \in T^*$.
- To create grammars which are more powerful, we must relax some of these restrictions.
- By retaining the restriction on the left side but permitting anything on the right, we get context-free grammar.

**Definition:**

- A grammar G = (V,T,S,P) is said to be context-free if all productions in P have the from:

$$A \rightarrow x$$

Where, $A \in V$ and $x \in (VUT)^*$

Where,

T = Terminals.
V = Non-Terminals or variables.
S = Start symbol.
P = Finite set of rules or productions that represent the recursive definition of a language.

- Each production consists of:
  - **a)** A variable that is being (partially) defined by the production. This variable is often called the head of the production.
  - **b)** The production symbol →.
  - **c)** A string of zero or more terminals and variables. This string is called the body of the production.

**Note:**

A language is said to be context free if and only if there is a context free grammar G such that L = L(G).

**Example:** The Grammar G = ({S}, {a,b}, S,P) with productions:

$S \rightarrow aSa$,
$S \rightarrow bSb$,
$S \rightarrow \in$
is context-free.

A typical derivation in this grammar is:

$S \Rightarrow aSa$
$\Rightarrow aaSaa$
$\Rightarrow aabSbaa$
$\Rightarrow aabbaa$

i.e., $L(G) = \{ww^R \mid w \in \{a,b\}^*\}$ which is context-free language.

**Derivation of string: left most derivation and right most derivation**

- In a grammar that is not linear, a derivation may involve sentential forms with more than one variable.
- In such cases, the variables can be replaced in order of choice

**Example:** The grammar G = ({A,B,S} , {a,b}, S, P) with productions.

$S \rightarrow AB$
$A \rightarrow aaA$
$A \rightarrow \in$
$B \rightarrow Bb$
$B \rightarrow \in$

The above grammar generates the language $L(G) = \{a^{2n}b^m \mid n \geq 0, m \geq 0\}$.

**Left most derivation for the string "aab"**

Step 1 : $S \Rightarrow AB$
Step 2 : $S \Rightarrow aaAB$
Step 3 : $S \Rightarrow aaB$
Step 4 : $S \Rightarrow aaBb$
Step 5 : $S \Rightarrow aab$

**Right most derivation for the string "aab"**

Step 1 : $S \Rightarrow AB$
Step 2 : $S \Rightarrow ABb$
Step 3 : $S \Rightarrow Ab$
Step 4 : $S \Rightarrow aaAb$
Step 5 : $S \Rightarrow aab$

**Definition**

A derivation is said to leftmost if in each step, the leftmost variable in the sentential form is replaced. If in each step, the rightmost variable is replaced, derivation is known as rightmost.

## Previous Years' Question

Identify the language generated by the following grammar, where S is start variable.

**(GATE-2017 (Set-2))**

S→XY

X→zX|a

Y→aYb|∈

**1)** $\{a^m b^n | m \geq n, m > 0\}$ **2)** $\{a^m b^n | m \geq n, n \geq 0\}$

**3)** $\{a^m b^n | m > n, m \geq 0\}$ **4)** $\{a^m b^n | m \geq n, n > 0\}$

**Sol: Option 3)**

## Previous Years' Question

Which of the following languages is generated by the given grammar?

**(GATE-2016 (Set-1))**

S→aS|bS|∈

**1)** $\{a^n b^m\} | n, m \geq 0\}$

**2)** {w∈{a, b}* |w has equal number of a's and b's}

**3)** $\{a^n | n \geq 0\} \cup \{b^n | n \geq 0\} \cup \{a^n b^n | n \geq 0\}$

**4)** {a, b}*

**Sol: Option 4)**

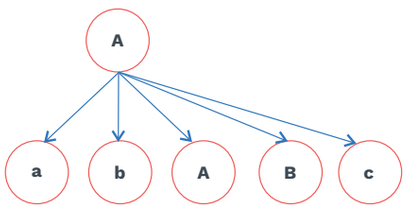
**Derivation trees:**

A second way of showing derivations, independent of the order in which productions are used; is by a derivation tree or parse tree.

### Definition

A derivation tree is an ordered tree in which nodes are labelled with the left sides of production and in which the chileren of a node represent its corresponding right sids

**Example:** A → abABc




**Grey Matter Alert!**

Let G = (V, T, S, P) be a context free grammar. An ordered tree is a derivation tree for G if and only if it has the following propertie:

**a)** The root is labeled S.

**b)** Every leaf has a label from T∪{∈}.

**c)** Every interior vertex (a vertex that is not a leaf) has a label from V.

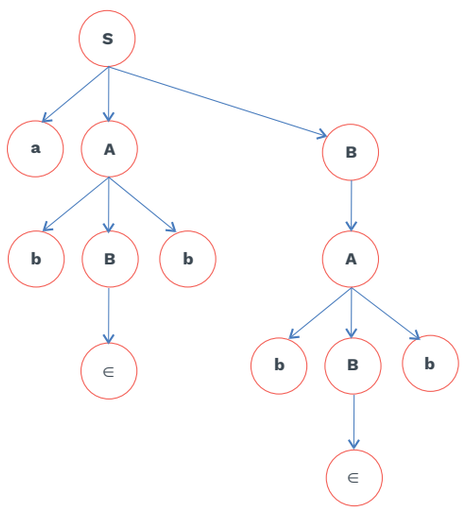
**d)** If a vertex has label A ∈ V, and its children are labelled (from left to right) $a_1$, $a_2$, ..... $a_n$, then P mustcontain a production of the form.
$A \rightarrow a_1, a_2, \ldots\ldots\ldots a_n$

**e)** A leaf labeled ∈ has no siblings, that is a vertex with a child labeled ∈ can have no other children.

**Example:** consider the grammar G, with productions-

$S \rightarrow aAB$,
$A \rightarrow bBb$,
$B \rightarrow A|\in$

**Parse tree for "abbbb":**



**Rack Your Brain**

Which language is generated by grammer S → aSb|SS|∈

## Previous Years' Question

A CFG G is given with the following productiions where S is the start symbol, A is non-terminal and a and b are terminals. **(GATE-IT-2008)**

S→aS|A

A→aAb|bAa|∈

Which of the following strings is generated by the grammar above?

**1)** aabbaba **2)** aabaaba

**3)** ababab **4)** aabbaab

**Sol: Option 4)**

**Ambiguity in grammar:**

When a grammar fails to provide a unique structure, it is sometimes possible to redesign the grammar to make the structure unique for each string in the language; but sometimes we cannot. That is known as ambiguity in grammar.

**Ambiguous grammar:**

**Definition**

A context free grammar G is said to be ambiguous if there exists some w∈L(G) that has atleast two distinct derivation trees (Parse trees).

**Note:**

Ambiguity implies the existence of two or more leftmost or rightmost derivations.

**Example:** E → E+E | E * E

Consider the sentential form E + E * E.

It has two derivations from E:

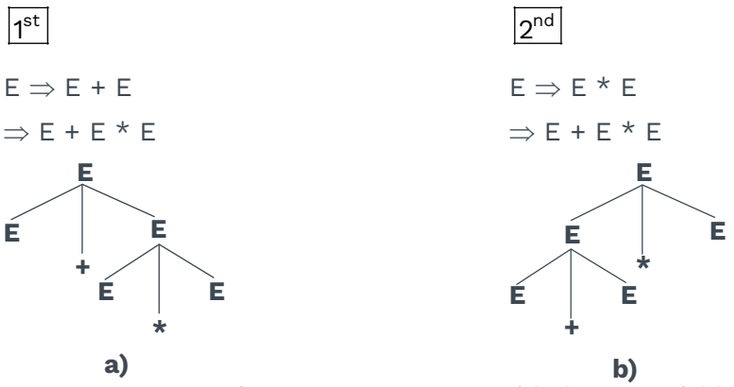



**Fig. 4.1 Two Parse Trees with the Same Yield.**

- Derivation 1st says that the second and third expressions are multiplied and the result is added to the first expression.
- Derivation 2nd adds the first two expressions and multiplies the result by the third.
  **Example:** 1+2*3
  1st derivation gives output = 1+(2*3)=7
  2nd derivation gives output = (1+2)*3 = 9
- Hence, we can say that there is an ambiguity.

**Rack Your Brain**

**1)** Find whether the following grammar is ambiguous or not?
$S \rightarrow aSbS|bSaS|\in$

**2)** Is it possible for a regular grammar to be ambiguous?

**Causes of ambiguity in the grammar:**
There are two causes of ambiguity in the grammar:
**1)** The precedence of operators is not respected.
**Example:** As explained in figure 4.1, both are valid parse trees, and ambiguity comes only because precedence is not followed. We need to force only the structure of figure 4.1 (a) to be legal in unambiguous grammar.
**2)** A sequence of identical operators can group either from the left or from the right.
**Example:** Two different parse tree exists for the string E+E+E. Since addition is associative, doesn't matter whether we group from left or right, but to eliminate ambiguity, we must pick one.

**Example of an unambiguous grammar:**

| **Example 1:** | **Example 2:** | **Example 3:** |
|---|---|---|
| $E \rightarrow E + T$ | $S \rightarrow aAB$ | $X \rightarrow AB$ |
| $T \rightarrow T * F$ | $A \rightarrow bBb$ | $A \rightarrow Aa\|a$ |
| $F \rightarrow id$ | $B \rightarrow A\|\in\Omega$ | $B \rightarrow b$ |

**Rack Your Brain**

Is deterministic context free grammars is always unambiguous?

**Grey Matter Alert!**

Context free languages, generated only by ambiguous grammars, is known as inherently ambiguous.

## Previous Years' Question

A CFG G is given with the following productions where S is the start symbol, A is non-terminal and a and b are terminals. **(GATE-IT-2008)**

S→aS|A

A → aAb|bAa|∈

For the string, "aabbaab" how many steps are required to derive the string and how many parse trees are there?

**1)** 6 and 1 **2)** 6 and 2

**3)** 7 and 2 **4)** 4 and2

**Sol: Option 1)**

## Previous Years' Question

A context free grammar is ambiguous if: **GATE (CS)-1987)**

**a)** The grammar contains useless non-terminals

**b)** It produces more than one parse tree for some sentence.

**c)** Some production has two non-terminals side by side on the right-hand side.

**d)** None of the above

**Sol: Option b)**

**Conversion of CFG into simplified CFG:**

To get the Chomsky Normal Form, we need to make a number of preliminary simplifications, i.e. we need to convert CFG into simplified CFG.

**i)** We must eliminate $\in$-productions, those of the form $A \rightarrow \in$ for some variable A.

**ii)** We must eliminate unit productions, those of them from $A \rightarrow B$ for variables A and B.

**iii)** We must eliminate useless symbols, those variables, or terminals that do not appear in any derivation of a terminal string from the start symbol.

**$\in$ productions:**

**Definition**

Any production of a context free grammar of the form $A \rightarrow \in$ is called as $\in$-production

**Note:**

A variable is nullable if it can derive E directly or indirectly.

**Eliminating $\in$ productions:**

- A grammar may generate a language not containing $\in$, but can have some $\in$-production or nullable variables. In such a case, the $\in$-productions can be removed.

**Example:** $S \rightarrow ABC$
$A \rightarrow BC|a$
$B \rightarrow bAC|\in$
$C \rightarrow cAB|\in$
contains $\in$–productions.

**Step 1:** Find all the nullable variables or null variables. Remove all the null production from the grammar.

**Step 2:** Replace the right-hand side of the production with and without $\in$; wherever nullable variables or null variables are present on the RHS of production.

**Step 3:** After this, whatever grammar we get, that grammar is without $\in$- production.

**Note:**

If start symbol ($S \rightarrow \in$) contain $\in$ - production i.e.
if the language contain $\in$, then we cannot eliminate $\in$ - productions, totally.

**Q1** **S → AbaC**
**A → BC**
**B → b|∈**
**C → D|∈**
**D → d**
**Eliminate ∈- productions from the given grammar**

**Sol:** **Step 1:** Nullable or null Variable = {B, C, A}
B contains ∈ production, so it is a nullable variable, similarly C is also a nullable variable.
A contains production A → BC, where B and C both are null variables. Thus A is also deriving epsilon. Hence A is also a nullable variable.

Remove B → ∈ , C → ∈ productions from the given grammar.

**Step 2:**
S → AbaC
S → Aba [Substitute C → ∈]
S → baC [Substitute A → ∈]
S → ba [Substitute A → ∈, C → ∈]

A → BC
A → B [Substitute C → ∈]
A → C [Substitute B → ∈]
B → b
C → D
D → d

**Step 3:** The final Grammar is:
S → AbaC | Aba | baC | ba
A → BC | B | C
B → b
C → D
D → d

**Q2** **S → ABC**
**A → BC|a**
**B → bAC|∈**
**C → cAB|∈**
**Eliminate ∈-productions from the given grammar.**

**Sol:** **Step 1:** Nullable and null variables = {A,B,C}

B contains $\in$ production, so it is a nullable variable, similarly C is also nullable variable.

A contains production A → BC, where B and C both are null variables. Thus A is also deriving epsilon. Hence A is also a nullable variable.

**Step 2:**

| | |
|---|---|
| S → ABC | |
| S → AB | [Substitute C → $\in$] |
| S → BC | [Substitute A → $\in$] |
| S → AC | [Substitute B → $\in$] |
| S → A | [Substitute B → $\in$, C → $\in$] |
| S → B | [Substitute A → $\in$, C → $\in$] |
| S → C | [Substitute A → $\in$, B → $\in$] |
| A → BC | |
| A → B | [Substitute C → $\in$] |
| A → C | [Substitute B → $\in$] |
| A → a | |
| B → bAC | |
| B → bA | [Substitute C → $\in$] |
| B → bC | [Substitute A → $\in$] |
| B → b | [Substitute A → $\in$, C → $\in$] |
| C → cAB | |
| C → cA | [Substitute B → $\in$] |
| C → cB | [Substitute A → $\in$] |
| C → c | [Substitute A → $\in$, B → $\in$] |

**Step 3:** Final Grammar is:

S → ABC | AB | BC | AC | A | B | C

A → BC | B | C | a

B → bAC | bA | bC | b

C → cAB | cA | cB | c

**Rack Your Brain**

Find a context free grammar without $\in$-production equivalent to the grammar defined by:

S → ABaC

A → BC

B → b|$\in$

C → D|$\in$

D → d

**Eliminating unit productions:**

**Definition**

Unit Production: Any production of a context-free grammar of the form $A \to B$, where $A, B \in V$ (variables), is called a unit production.

To remove unit production, we use the substitution rule:
**Step 1:** Write the given grammar without unit production.
**Step 2:** Write the given grammar's all those production which goes to the unit production and further unit production produce terminals.
**Step 3:** Add all those new productions in the grammar obtained in step 1.

**Note:**
Meaning of Grammar should not be changed while doing elimination of unit production.

## SOLVED EXAMPLES

**Q1** $S \to Aa \mid B$
$B \to A \mid bb$
$A \to a \mid bc \mid B$
Eliminate unit-production from the given grammar.

**Sol:** **Step 1**: Grammar without unit production.
$S \to Aa$
$B \to bb$
$A \to a \mid bc$

**Step 2:** Productions which gives unit production:

$S \to B \to bb$
$S \to B \to A \to a$
$S \to B \to A \to bc$
} So add all these productions ir grammar obtained in step 1.

$B \to A \to a$
$B \to A \to bc$
} add these productions in grammar obtained in step 1.

$A \to B \to bb$ } add these productions in grammar obtained in step 1.

**Step 3:** Final grammar without unit production.

$S \rightarrow Aa \mid bb \mid a \mid bc$
$B \rightarrow bb \mid a \mid bc$
$A \rightarrow a \mid bc \mid bb$

**Q2** $S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C|b$
$C \rightarrow D$
$D \rightarrow E$
$E \rightarrow a$
S, A, B, C, D, E E Variables and T = {a, b}
Eliminate Unit - Productions from the given grammar.

**Sol:**

**Step 1:** Productions without unit production.

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$
$E \rightarrow a$

**Step 2:** Productions which give unit – production.

**i)** $B \rightarrow C \rightarrow D \rightarrow E \rightarrow a$
**ii)** $C \rightarrow D \rightarrow E \rightarrow a$
**iii)** $D \rightarrow E \rightarrow a$

} Add all this production in grammar obtained in step 1

**Step 3:** Grammar without unit productions

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b \mid a$

$C \rightarrow a$
$D \rightarrow a$
$E \rightarrow a$

} Here C, D, E are not reachable from the star So, these are useless symbols which will be removed.

Final Grammar will be:

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b \mid a$

**Eliminating useless symbols:**

- Symbol X is a useless symbol for a grammar G = (V,T,P,S) if that symbol can never take part in any derivation.

**Example:** $S \rightarrow aSb|\in|A$
$A \rightarrow aA$

- Here, the production $S \rightarrow A$ is useless as A cannot be transformed into a terminal string.
- Removing this production $S \rightarrow A$ leaves the language unaffected and is a simplification by any definition.

**Note:**
A variable is useful if it is reachable from the start state and is also able to derive some terminal or some strings of terminals.

## SOLVED EXAMPLES

**Q1** **$S \rightarrow A$**
**$A \rightarrow aA \mid \in$**
**$B \rightarrow bA$**
**Eliminate useless symbol from the given grammar.**

**Sol:** Here, we can remove $B \rightarrow bA$, since B is not reachable from the start state.
Final grammar is
$S \rightarrow A$
$A \rightarrow aA|\in$

**Q2** **$S \rightarrow aAa|aBC$**
**$A \rightarrow aS|bD$**
**$B \rightarrow aBa|b$**
**$C \rightarrow abb|DD$**
**$D \rightarrow aDa$**
**Eliminate useless symbol from the given grammar.**

**Sol:**

- Initially, only B and C be the non-terminal symbols that directly generates terminal strings; hence useful symbol.
- S is useful because the rule S → aBC
- A is useful because the rule A → aS
- D does not generate terminal strings; thus, it is a useless symbol.
- Eliminate D and those productions involving D, we get grammar:
  $S \rightarrow aAa|aBC$

A → aS
B → aBa | b
C → abb

**Q3** **S → AB|AC**
**A → aAb|bAa|a**
**B → bbA|aaB|AB**
**C → abCA|aDb**
**D → bD|aC**
**Eliminate useless symbol from the given grammar.**

**Sol:** **Another way**

- Terminals are always useful symbols. Since here, a and b are terminal. So, they are useful.
- Combination of terminal symbols generated by any rules then non-terminals also useful.
- As A → a, so, A is useful, B → bbA which produces B → bba so, B also useful.
- Similarly, S → AB is a combination of two useful symbols A and B. So, S is also useful.
  A useful symbols = {a, b, A, B, S}
- C and D must be removed because they are not producing terminals.
  So, remove S → AC, C → abCA | aDb, D→ bD | aC from the grammar.

**Final grammar:**
S → AB
A → aAb | bAa | a
B → bbA | aaB| AB
All are reachable from the start symbol S.

**Rack Your Brain**

What will be the final grammar after removing the useless symbols from the given grammar:
S → aS|A|C
A → a
B → aa
C → aCb

**Theorem:**
Let G = (V,T,S,P) be a context-free grammar. Then there exists an equivalent grammar $\hat{G} = (\hat{V}, \hat{T}, \hat{S}, \hat{P})$ that does not contain any useless variables or productions.

**Note:**
When we want to convert any CFG into an equivalent CFG that has no useless symbols, $\in$-productions and unit-productions. Some care must be taken in order of application of the consturction.
A safe order is:

**i)** Eliminate E-production.
**ii)** Eliminate unit production.
**iii)** Elminiate useless-symbols.

**Rack Your Brain**

Does removal of $\in$-productions introduces previously non-existent unit-productions?

**Normal forms:**
When working with context-free grammar, it is often convenient to have them in simplified form. There are two types of normal forms of context-free grammar.

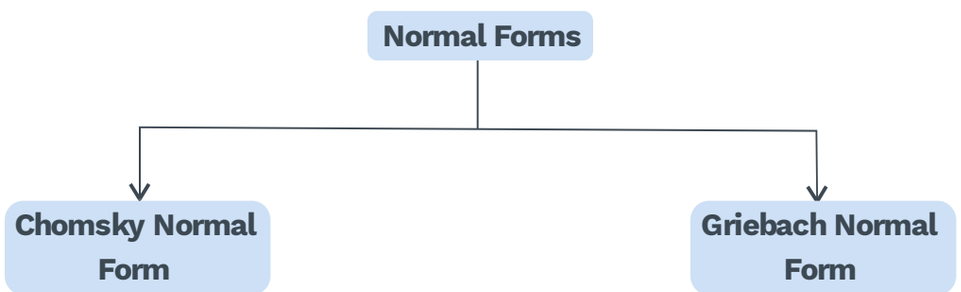



**Fig. 4.2 Types of Normal Forms**

**Chomsky's normal form:**

**Definition**

A context free grammar is in Chomsky Normal Form, if every rule is of the form:

$A \rightarrow BC$

OR

$A \rightarrow a$

where A, B, C are in V (variables) and a is any terminal.

**Note:**

In addition, we permit the rule $S \rightarrow \in$, where S is the start symbol in Chomsky Normal Form.

**Example:** $S \rightarrow AS \mid a$

$A \rightarrow SA \mid b$

is in Chomsky Normal Form.

Whereas the grammar:

$S \rightarrow AS \mid AAS$

$A \rightarrow SA \mid aa$

is not in Chomsky Normal Form; both productions $S \rightarrow AAS$ and $A \rightarrow aa$ violate the condition of the definition of Chomsky Normal Form.

**Theorem:**

Any context-free language is generated by a context-free grammar in Chomsky's normal form.

**Advantages of Chomsky's normal form:**

**i)** Length of each production is restricted.

**ii)** Derivation tree (parse tree) obtained from CNF is always a binary tree.

**iii)** The number of steps required to derive a string of length |w| is (2|w|-1).

**Example:** $S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

Let us have to derive the string w = ab.

So, |w| = length of string = 2.

$S \Rightarrow AB$

$\Rightarrow aB$

$\Rightarrow ab$

3 steps are needed. Thus a number of steps are required to derive a string 'ab' in the given grammar

= 2|w| – 1
= 2×2 – 1 = 3

**iv)** It is easy to apply the CYK membership algorithm, which is of the polynomial type having time complexity = $O(n^3)$

**Note:**

Given any Chomsky Normal Form Grammar, CYK membership algorithm can parse it.

**Previous Years' Question**

If G is a context free grammar and w is a string of length l in L(G), how long is a derivation of w in G, if G is in Chomsky Normal Form? **(GATE - 1992)**

**1)** 1) 2l **2)** 2l+1
**3)** 2l – 1 **4)** l

**Sol: Option 3)**

**CYK algorithm:**

This algorithm is only applicable if the given grammar is in CNF.
Check whether the string "baaba" is a valid member of the following CFG.

S → AB|BC
A → BA|a
B → CC|b
C → AB|a

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | b | a | a | b | a |

| | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|
| 1 | A,S | ϕ | ϕ | A,S | B |
| 2 | S,C,A | B | B | A,C | |
| 3 | B | S,C | A,C | | |
| 4 | A,S | B | | | |
| 5 | A,C | | | | |

Let's see how to fill the above table:

**1)** (1,1) → Starting with '1' ('b') and ending with '1' ('b')
'b' is generated by B.
Similarly we can compute for (2,2), (3,3), (4,4) (5,5)

**2)** (1,2) → Starts with 1('b) and ends with 2('a')
(1,2) = (1,1) (2,2)
= {B} {A,C}
= {BA, BC}

'BA' is generated by 'A'
'BC' is generated by 'S'

**3)** (2,3) = (2,2) (3,3)
= {A,C} {A,C}
= {AA, AC, CA, CC}
| | | |
X X X generated by 'B'

**4)** (3,4) = {3,3} {4,4}
= {A,C} {B}
= {AB, CB}
| |
generated X
by 'S' and 'C'

**5)** (4,5) = {4,4} {5,5}
= {B} {A,S}
= {BA, BC}
/ \
generated generated
by 'A' by 'S'

**6)** (1, 3) = = {1,1} {2,3} or {1,2} {3,3}
= {B} {B} = {A,S} {A,C}
= {BB} = {AA, AC, SA, SC}
| | | | |
X X X X X

$\therefore$ (1,3) = $\phi$
Similarly, we can fill the other entries in the table.
(1,5) can be generated by 'S' (Starting symbol)
$\therefore$ baaba can be generated by grammar.

What is the substring of "baaba" that can be derived from this grammar?

From the table entries, we select the entries that have 'S' (starting symbol)

(1, 2) → 'S' can generate 'ba'
(2, 5) → 'S' can generate 'aaba'
(3, 4) → 'S' can generate 'ab'

**Time complexity:**
We are filling $n^2$ cells.
For each cell, in the worst case, we might need O(n) time.

$$\text{Total complexity} = O(n^2) \times O(n)$$
$$= O(n^3)$$

**Greibach normal form:**
A context-free grammar is in GNF (Greibach Normal Form) if all the productions are of the form:

$$\boxed{A \to a \propto}$$

where $\propto \in v^*$ (non-terminal)
$a \in T$ (Terminals), $A \in V$
**Example:** $A \to aBCDE$
$A \to a$

**Note:**

Every context free grammar can be transformed into an equivalent grammar in Greibach Normal Form.

**Advantages of Greibach normal form:**
**1)** The number of steps required to generate a string of length |w| is |w|.
**Example:** $A \to aB$
$B \to b$

Let we have to derive the strings w = 'ab', |w| = length of strings = 2
**Step 1:** $A \Rightarrow aB$
**Step 2:** $A \Rightarrow ab$

So, the number of steps required to derive the string w = 'ab' in the given form is 2
**2)** Greibach Normal Form (GNF) is useful in order to convert a CFG into PDA.

**Converting context-free grammar to Chomsky's normal form:**

## SOLVED EXAMPLES

**Q1** **Convert the grammar with productions:**
**S → ABa,**
**A → aab,**
**B → Ac**
**to Chomsky Normal Form.**

**Sol:** As required, the grammar does not have any $\in$-productions or any unit productions.

**Step 1:** Introduce new variables $N_a$, $N_b$, $N_c$ and replace a with $N_a$, b with $N_b$, c with $N_c$.

$S \rightarrow ABN_a$
$A \rightarrow N_aN_aN_c$
$B \rightarrow AN_c$
$N_a \rightarrow a$
$N_b \rightarrow b$
$N_c \rightarrow c$

**Step 2:** Introduce additional variables to get the first two productions into Normal form:

$S \rightarrow AD_1$
$D_1 \rightarrow BN_a$
$A \rightarrow N_aD_2$
$D_2 \rightarrow N_aN_b$
$B \rightarrow AN_c$
$N_a \rightarrow a$
$N_b \rightarrow b$
$N_c \rightarrow c$

**Note:**

CNF produces the same language which is generated by CFG (Context Free Grammar).

**Q2** **Convert the following grammar with production:**
**$S \rightarrow bA \mid aB$**
**$A \rightarrow bAA \mid aS \mid a$**
**$B \rightarrow aBB \mid bs \mid b$**
**to Chomsky Normal Form.**

**Sol:** As required, the grammar does not have any $\in$-productions or any unit productions.

**Step 1:** Introduce new variables $N_a$, $N_b$.

$S \rightarrow N_bA \mid N_aB$
$A \rightarrow N_bAA \mid N_aS \mid a$
$B \rightarrow N_aBB \mid N_bS \mid b$

$N_a \rightarrow a$
$N_b \rightarrow b$

**Step 2:** Introduce additional variables $C_1$ and $C_2$.

$S \rightarrow N_bA \mid N_aB$
$A \rightarrow N_bC_1 \mid N_aS \mid a$
$B \rightarrow N_aC_2 \mid N_bS \mid b$
$N_a \rightarrow a$
$N_b \rightarrow b$
$C_1 \rightarrow AA$
$C_2 \rightarrow BB$

**Note:**

For a given grammar, more than one CNF can be possible.

**Grey Matter Alert!**

Given a grammar G of length n, we can find an equivalent Chomsky Normal Form Grammar for G in time $O(n^2)$, the resulting grammar has length $O(n^2)$.

**Converting context-free grammar to Greibach normal form conversion (CFG → GNF)**

Variable → Terminal
Variable → Terminal variable
Variable → Terminal variable variable variable..

## SOLVED EXAMPLES

**Q1** **Convert the following context free grammar**
**S → aSb | aB**
**B → b**
**to Griebach Normal Form**

**Sol:** **In GNF:**

S → aSB | aB
B → b

**Q2** **Convert the following context free grammar**

**S → bAc | cB**

**A → b|c**

**B → c**

**where A, B, S ∈ variables and b, c ∈ terminal**

**to Griebach Normal Form**

**Sol:**

**In GNF:**

S → bAB | cB

A → b | c

B → c

**Q3** **Convert the following context free grammar**

**S → mSn | mn**

**to Griebach Normal Form**

**Sol:**

**In GNF:**

S → mSA | mA

A → n

**Q4** **Convert the following context free grammar**

**S → aBb | bCa**

**B → bBa | b**

**C → bCa | a**

**Sol:**

**In GNF:**

S → aBX | bCY

B → bBY | b

C → bCY | a

X → b

Y → a

## 4.2 PUSHDOWN AUTOMATA

**Definition**

A pushdown Automata is a non-deterministic finite automaton coupled with a stack that can be used to store a string of arbitrary length. The stack can be read and modified only at its top.

**(OR)**

The automaton which defines context-free languages is known as pushdown Automata.

A Non-deterministic pushdown Acceptor (NPDA) is defined by the set tuples:

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

Where,

Q : A finite set of states,

$\Sigma$ : A finite set of input symbols.

$\tau$ : A finite stack alphabet.

(set of symbols that we are allowed to push into the stack)

$\delta : Q \times (\Sigma \cup \{\in\}) \times \tau \rightarrow$ set of finite subsets of $Q \times \tau^*$ is the transition function.

$q_0 \in Q$ is the initial state of the control unit.

$Z_0 : Z_0 \in \tau$ is the stack start symbol.

$F : F \subseteq Q$ is the set of final states.

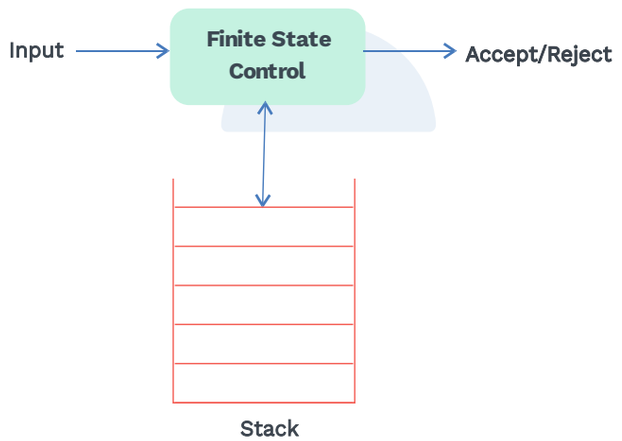


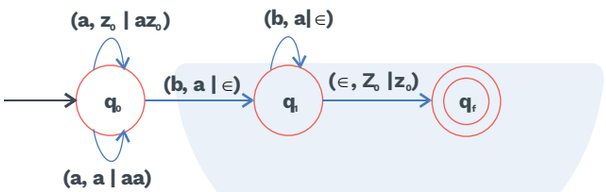

**Fig. 4.3 Pushdown Automata**

- Pushdown Automata is basically a finite Automaton with a stack.
- A pushdown Automata can write symbols on the stack and read them back later.
- Writing a symbol on the stack is referred to as pushing the symbol.
- Removing a symbol is referred to as popping it.

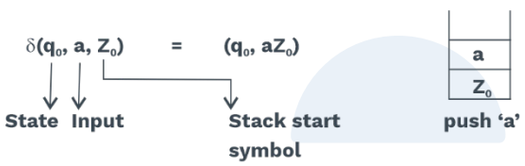
**Grey Matter Alert!**

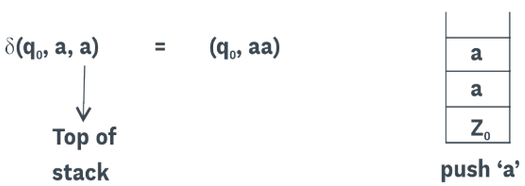
A stack is valuable part of Pushdown Automata because it can hold an unlimited amount of information.

**Example:** L = $\{a^n b^n \mid n \geq 1\}$
We cannot construct the finite Automata corresponding to this language.
L is context-free language.
We can construct a push down Automata for Language L.

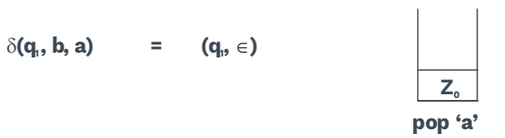



Consider a string w = aabb







$\delta(q_1, \in, Z_0)$ = $(q_2, \in)$ or $(q_2, Z_0)$

$(q_2, \in)$ ↓ Acceptance by empty stack

$(q_2, Z_0)$ ↓ Acceptance by final state

**Pushdown automata acceptance can be done in 2 ways:**

**1) acceptance by final state:**

Let P = (Q, ∑, τ, δ, $q_0$, $z_0$, F) be a PDA.

Then language accepted by P is the set

$$L(P) = \{W \mid (q_0, w, z_0) \vdash_P^* (q_f, \in, s)\}$$

Where $q_f$ is a final state and s is any stack string.

i.e. language accepted by P is the set of all strings that can put P into a final state at the end of the string.

**2) Acceptance by empty stack:**

An NPDA P = (Q, ∑, τ, δ, $q_0$, $z_0$, F) is said to accept the language N(P) by empty stack if,

$N(P) = \{W \mid (q_0, w, z_0) \vdash_P^* (q, \in, \in)\}$ for any state q.

That is, N(P) is a set of inputs W that P can consume, and when the end of the input string is reached, PDA has emptied the stack.

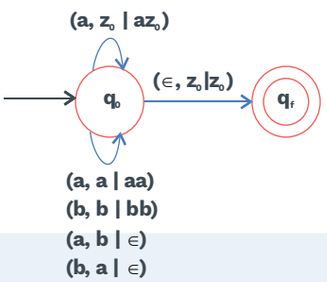
**Deterministic and non deterministic pushdown automata:**

- Deterministic pushdown Automata and Non-Deterministic pushdown differ in terms of their expressive power.
- Non-Deterministic pushdown automata are more powerful than Deterministic pushdown Automata.
- Transition function (δ) for Deterministic Pushdown Automata:
  $\delta: Q \times \{\sum \cup \in\} \times \tau \rightarrow Q \times \tau^*$
- Transition function (δ) for Non-Deterministic Pushdown Automata:
  $\delta : Q \times \{\sum \cup \in\} \times \tau \rightarrow$ set of finite subsets of $Q \times \tau^*$ is the transition function.

## SOLVED EXAMPLES

**Q1** **Construct Pushdown Automata for the language L = {w|$n_a(w) = n_b(w)$, w ∈ {a, b}*}.**

**Sol:**




Consider a string w = abbbaa
Here, number of a's in w = number of b's in w

$\delta(q_0, a, z_0) = (q_0, az_0)$


**push 'a'**

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


**pop 'a'**

$\delta(q_0, b, z_0) = (q_0, bz_0)$


**push 'b'**

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


**push 'b'**

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


**pop 'b'**

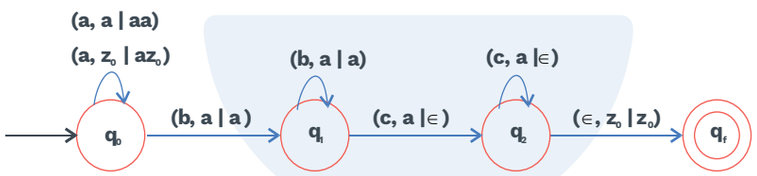
$\delta(q_0, a, b) = (q_0, \in)$


pop 'b'

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

**Q2** **Construct a PDA for the language : L = {$a^n b^m c^n$ | n, m ≥ 1}**

## Sol:

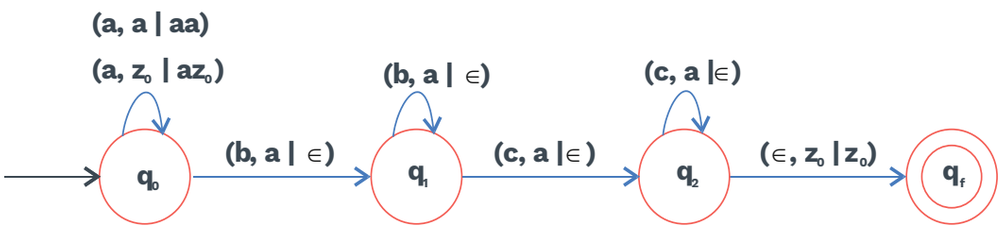



Consider a string w = aabcc

| | | | |
|---|---|---|---|
| $\delta(q_0, a, z_0)$ | = | $(q_0, az_0)$ | push 'a' onto the stack |
| $\delta(q_0, a, a)$ | = | $(q_0, aa)$ | push 'a' onto the stack |
| $\delta(q_0, b, a)$ | = | $(q_1, a)$ | Do Nothing (skip) |
| $\delta(q_1, c, a)$ | = | $(q_2, \in)$ | pop 'a' |
| $\delta(q_2, c, a)$ | = | $(q_2, \in)$ | pop 'a' |
| $\delta(q_2, \in, z_0)$ | = | $(q_f, z_0)$ | |

**Q3** **Construct a PDA for the language : L = {$a^{m+n} b^m c^n$ | m, n ≥ 1}**

## Sol:



Consider a string w = aaabbc

We can write L = $\{a^{m+n}b^mc^n \mid m, n \geq 1\}$

as L = $\{a^na^mb^mc^n \mid m, n \geq 1\}$

| | | | |
|---|---|---|---|
| $\delta(q_0, a, z_0)$ | = | $(q_0, az_0)$ | push 'a' onto the stack |
| $\delta(q_0, a, a)$ | = | $(q_0, aa)$ | push 'a' onto the stack |
| $\delta(q_0, a, a)$ | = | $(q_0, aa)$ | push 'a' onto the stack |
| $\delta(q_0, b, a)$ | = | $(q_1, \in)$ | pop 'a' |
| $\delta(q_1, b, a)$ | = | $(q_1, \in)$ | pop 'a' |
| $\delta(q_1, c, a)$ | = | $(q_2, \in)$ | pop 'a' |
| $\delta(q_2, \in, z_0)$ | = | $(q_f, z_0)$ | |

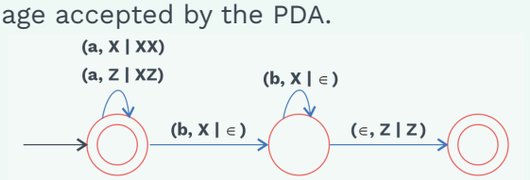
**Rack Your Brain**

Construct a PDA for the language: L = $\{a^{n+1}\ b^{2n} \mid n \geq O\}$

**Previous Years' Question**

Consider the transition diagram of a PDA given below with input alphabet ∑ = {a, b} and stack alphabet τ = {X, Z}, Z is the initial stack symbol.

Let L denote the language accepted by the PDA.




Which one of the following is TRUE? **(GATE 2016 (Set-1))**

**1)** L = $\{a^nb^n \mid n \geq O\}$ and is not accepted by any Finite Automata.

**2)** L = $\{a^n \mid n \geq O\} \cup \{a^nb^n \mid n \geq O\}$ and is not accepted by any Deterministic PDA.

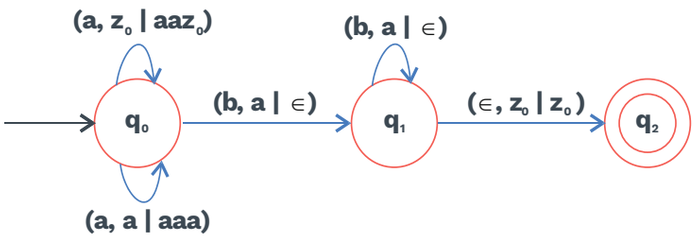
**3)** L is not accepted by any Turing Machine that halts on every input.

**4)** L = $\{a^n \mid n \geq O\} \cup \{a^nb^n \mid n \geq O\}$ and is deterministic context free.

**Sol: Option 4)**

**Q4** **Construct a PDA for the language : $L = \{a^n b^{2n} \mid n \geq 1\}$**

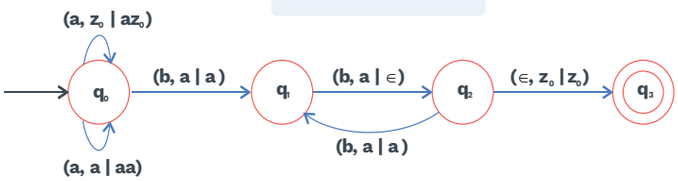
**Sol:** Push 2 a's onto stack on seeing 1 a in input




Consider a string w = aabbbb

| | | | |
|---|---|---|---|
| $\delta(q_0, a, z_0)$ | = | $(q_0, aaz_0)$ | Push 2 a's on seeing 1 a in input |
| $\delta(q_0, a, a)$ | = | $(q_0, aaa)$ | Push 2 a's onto stack |
| $\delta(q_0, b, a)$ | = | $(q_1, \in)$ | Pop One a on input b |
| $\delta(q_1, b, a)$ | = | $(q_1, \in)$ | Pop One a on input b |
| $\delta(q_1, b, a)$ | = | $(q_1, \in)$ | Pop One a on input b |
| $\delta(q_1, b, a)$ | = | $(q_1, \in)$ | Pop One a on input b |
| $\delta(q_1, \in, z_0)$ | = | $(q_2, z_0)$ | |

**Method: 2**




Consider a string w = aabbbb

| | | | | |
|---|---|---|---|---|
| $\delta(q_0, a, z_0)$ | = | $(q_0, az_0)$ | a<br>$z_0$ | Push 'a' |
| $\delta(q_0, a, a)$ | = | $(q_0, aa)$ | a<br>a<br>$z_0$ | Push 'a' |
| $\delta(q_0, b, a)$ | = | $(q_1, a)$ | a<br>a<br>$z_0$ | Do Nothing (Skip) |

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

Pop 'a'

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

Do Nothing (Skip)

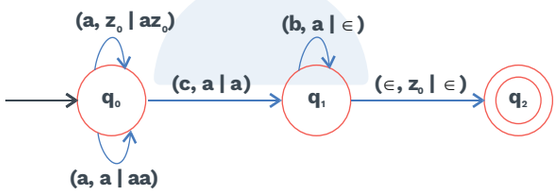
$\delta(q_1, b, a) = (q_2, \in)$ 

Pop 'a'

$\delta(q_2, \in, z_0) = (q_3, z_0)$

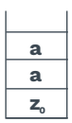
**Q5** **Construct a PDA for the language : L = {$a^n cb^n$ | n ≥ 1}**

**Sol:** On input 'a' perform push operation onto the stack. On input 'c', skip (i.e., do nothing). On input 'b' and 'a' on top of stack, perform pop operations.




Consider a string w = aacbb.

$\delta(q_0, a, z_0) = (q_0, az_0)$ 


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


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

Do Nothing

$\delta(q_1, b, a) = (q_1, \in)$ 

Pop 'a'

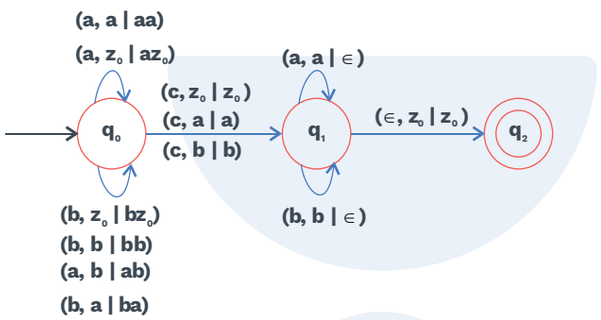
$\delta(q_1, b, a) =$ $(q_1, \in)$  Pop 'a'



$\delta(q_1, \in, z_0) = (q_2, z_0)$

**Q6** **Construct PDA which accepts the languages $L = \{wcw^R \mid w \in \{a, b\}^*\}$.**

## Sol:





Consider a string = $\frac{ab}{w}\frac{c}{c}\frac{ba}{w^R}$

$\delta(q_0, a, z_0) = (q_0, az_0)$  Push 'a'



$\delta(q_0, b, a) = (q_0, ba)$  Push 'b'



$\delta(q_0, c, b) = (q_1, b)$ Do Nothing

$\delta(q_1, b, b) = (q_1, \in)$  Pop 'b'



$\delta(q_1, a, a) = (q_1, \in)$  Pop 'a'



$\delta(q_1, \in, z_0) = (q_2, z_0)$

**Previous Years' Question**

Consider the Pushdown Automaton (PDA) below which runs over the input alphabet (a, b, c). It has stack alphabet $\{z_0, x\}$ where $z_0$ is the bottom-of-stack marker. The set of states of the PDA is (s, t, u, f) where s is the start state and f is the final state. The PDA accepts by final states. The transitions of the PDA given below are depicted in a standard manner. For example, the transition $(s, b, x) \rightarrow (t, xz_0)$ means that if the PDA is in state s and the symbol on the top of the stack is x, then it can read b from the input and move to state t after popping top of stack and pushing the symbols $z_0$ and x (in that order) on the stack. **(GATE IT 2006)**

$(s, a, z_0) \rightarrow (s, xxz_0)$

$(s, \in, z_0) \rightarrow (f, \in)$

$(s, a, x) \rightarrow (s, xxx)$

$(s, b, x) \rightarrow (t, \in)$

$(t, b, x) \rightarrow (t, \in)$

$(t, c, x) \rightarrow (u, \in)$

$(u, c, x) \rightarrow (u, \in)$

$(u, \in, z_0) \rightarrow (f, \in)$

The language accepted by the PDA is:

**1)** $\{a^l b^m c^n \mid l = m = n\}$

**2)** $\{a^l b^m c^n \mid l = m\}$

**3)** $\{a^l b^m c^n \mid 2l = m + n\}$

**4)** $\{a^l b^m c^n \mid m = n\}$

**Sol: Option 3)**

**Rack Your Brain**

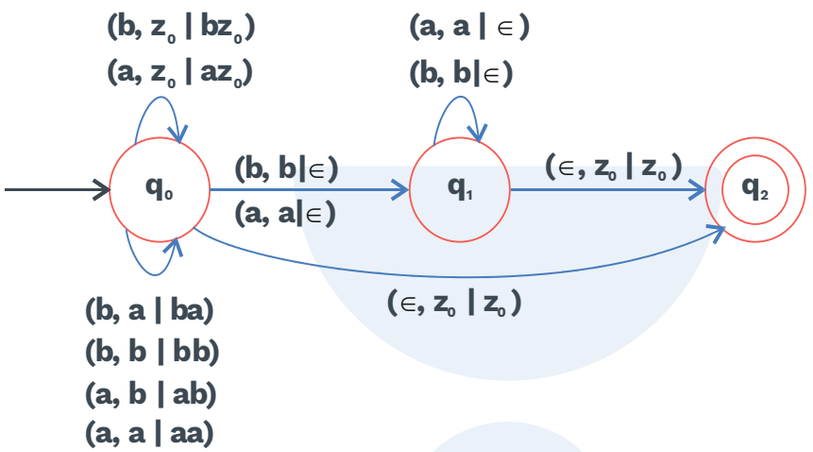
Construct PDA that accepts the following language on $\Sigma = \{a, b, c\}$

$L = \{a^n b^m c^{n+m} \mid n \geq 0, m \geq 0\}$.

**Q7** **Construct NPDA's that accepts the languages L = {$ww^R | w \in \{a, b\}^*$}.**

**Sol:** We cannot construct deterministic pushdown Automata for the given language L = {$ww^R | w \in \{a, b\}^*$}.

Here Non-determinism comes into picture since on same state, on same input more than one transition will be defined.




$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

$\delta(q_0, a, a) = (q_0, aa)$ or $(q_1, \in)$

$\delta(q_0, b, b) = (q_0, bb)$ or $(q_1, \in)$

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

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

$\delta(q_1, \in, z_0) = (q_2, z_0)$

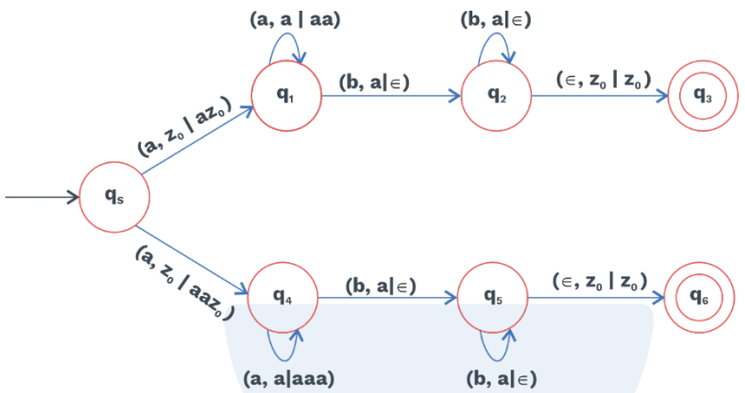
$\delta(q_0, \in, z_0) = (q_2, z_0)$

$\delta(q_1, a, a) = (q_1, \in)$

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

**Q.8** **Construct NPDA for the language : $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$**

## Sol:




**Difference between DPDA and NPDA:**

| Deterministic pushdown automata | Non-deterministic pushdown automata |
|---|---|
| **1)** Transition function for deterministic pushdown automata: $\delta : Q \times \{\Sigma \cup \in\} \times \tau \rightarrow Q \times \tau^*$ | **1)** Transition function for non-deterministic pushdown Automata: $\delta : Q \times \{\Sigma \cup \in\} \times \tau \rightarrow$ set of finite subsets of $Q \times \tau^*$ |
| **2)** Only one transition is defined from a state on an input symbol and stack symbol. | **2)** More than one transition can be defined from a state on an input symbol and stack symbol. |
| **3)** DPDA is less powerful than NPDA. | **3)** NPDA is more powerful than DPDA. |

**Table 4.1 Differences**

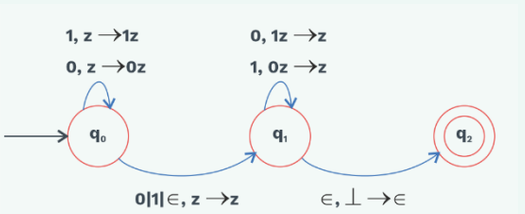
### Rack Your Brain

Construct NPDA's that accepts the language $L = \{a^n b^m \mid n \geq 0, n \neq m\}$.

**Previous Years' Question**

Consider the NPDA < Q = $\{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $\tau = \{0, 1, \perp\}$, $\delta$, $q_0$, $\perp$, F = $\{q_2\}$ >
Where (as per usual convention) Q is the set of states, $\Sigma$ is the input alphabet, $\tau$ is the stack alphabet, $\delta$ is the state transition function, $q_0$ is the initial state, $\perp$ is the initial stack symbol, and F is the set of accepting states. The state transition is as follow:

**( GATE 2015 (Set-1))**




Which one of the following sequences must follow the string 101100 so that the overall string is accepted by the automaton?

**1)** 10110 **2)** 10010

**3)** 01010 **4)** 01001

**Sol: Option 2)**

**Rack Your Brain**

Construct NPDA on $\Sigma = \{a, b, c\}$ that accepts the language
$L = \{w_1 c w_2 \mid w_1, w_2 \in \{a, b\}^*, w_1 \neq w_2\}$.

**Equivalence between CFL and CFG**

**1)** $L = \{a^m b^n \mid m = n\}$
Grammar corresponding to given CFL:
$S \rightarrow aSb|\in$

**2)** $L = \{a^m b^n \mid m = 2n\}$
Grammar Corresponding to given CFL:
$S \rightarrow aaSb|\in$

**3)** $L = \{a^m b^n c^n \mid m, n \geq 1\}$
Grammar Corresponding to given CFL:
$S \rightarrow AB$
$A \rightarrow a|aA$
$B \rightarrow bBc|bc$

**4)** $L = \{a^n b^p c^n \mid p, n \geq 1\}$
Grammar Corresponding to given CFL:
$S \rightarrow aSc|aAc$
$A \rightarrow bB|b$

**5)** $L = \{a^m b^n \mid m > n;\ m,n \geq 1\}$
Grammar Corresponding to given CFL:
$S \rightarrow aSb|aAb$
$A \rightarrow aA|a$

**6)** $L = \{a^m b^n \mid m \neq n;\ m,n \geq 1\}$
**Cases:** $m > n$ or $m < n$
$S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow aS_1b|aAb$
$A \rightarrow aA \mid a$
$S_2 \rightarrow aS_2b|aBb$
$B \rightarrow bB|b$

**7)** $L = \{w| \ |w_a| = |w_b|, w \in \{a, b\}^*\}$ (number of a's and number of b's should be equal)
Grammar Corresponding to given CFL:
$S \rightarrow SaSbS \mid SbSaS \mid \in$

**Closure properties of CFL's**

The context-free languages are closed under:

- **Union:**
  Let $L_1$ and $L_2$ be CFL's, then $L_1 \cup L_2$ is also context-free language.
- **Concatenation:**
  Let $L_1$ and $L_2$ be CFL's, then $L_1.L_2$ is also context-free language.
- **Kleen-closure:**
  If $L_1$ is a CFL, then $L_1^*$ is also context-free language.
- **Reversal:**
  The CFL's are also closed under reversal.
  If L is a CFL, then so is $L^R$.
- The family of context-free languages is not closed under intersection, set difference, and complementation.
- If CFL is not closed under intersection means there exist two CFL languages for which $CFL \cap CFL_2 \neq CFL$.

**Example: Consider the two languages:**

$L_1 = \{a^n b^n c^m \mid n \geq 0,\ m \geq 0\}$ and $L_2 = \{a^n b^m c^m \mid n \geq 0,\ m \geq 0\}$, then
$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ which is not context-free.

Thus, context-free language is not closed under intersections.

Again, $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

If the family of context-free languages was closed under complementation, then the right side of the above expression would be a context-free language for any context-free $L_1$ and $L_2$.

But this contradicts since we have shown, the intersection of two context-free languages is not necessarily context-free. Thus, the family of context-free languages is not closed under complementation.

**Note:**

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

**Rack Your Brain**

Is the language $L = \{a^n b^n | \ n \geq 0, n \neq 100\}$ is context free?

**Previous Years' Question**

Let $L_1$ be a regular language and $L_2$ be a context free language. Which of the following languages is/are context free? **(GATE 2021 (Set-2))**

**1)** $L_1 \cap \overline{L}_2$

**2)** $\overline{\overline{L}_1 \cup \overline{L}_2}$

**3)** $L_1 \cup (L_2 \cup \overline{L}_2)$

**4)** $(L_1 \cap L_2) \cup (\overline{L}_1 \cap L_2)$

**Sol: Option 2) 3), 4)**

**Previous Years' Question**

Let $L_1$, $L_2$ be any two context free languages and R be any regular language, then which of the following is/are **CORRECT**? **(GATE 2017 (Set-2))**

**I)** $L_1 \cup L_2$ is context free.

**II)** $\overline{L}_1$ is context free.

**III)** $L_1 - R$ is context free.

**IV)** $L_1 \cap L_2$ is context free.

**1)** I, II and IV only **2)** I and III only

**3)** II and IV only **4)** I only

**Sol: Option 2)**

- CFLs are closed with the following operations:
  - **1)** Union
  - **2)** Concatenation
  - **3)** Kleene closure
  - **4)** Substitution
  - **5)** Homomorphism
  - **6)** Inverse Homomorphism
  - **7)** Reversal
  - **8)** Intersection with a regular set
  - **9)** INIT (L)
- CFL are not closed under the following operations:
  - **i)** Intersection
  - **ii)** Complement
  - **iii)** Set difference
  - **iv)** Quotient
  - **v)** Inverse substitution

**Deterministic Context-Free Languages (DCFLs)**

**Definition**

The language that are recognized by deterministic pushdown Automata (DPDAs) are called as Deterministic Context free languages (DCFLs).

- This subclass of context-free languages is relevant to practical applications, such as the design of parsers in compilers for programming languages.

- In DPDAs, at each step of its computation, the DPDA has at most one way to proceed according to its transition function.

**Example:** $L = \{0^n1^n \mid n \geq 0\}$ is DCFL.

**Note:**

Every language accepted by DPDA is also accepted by NPDA but converse need not to be true.

**Closure properties of DCFL**

- The class of DCFLs is closed under complementation.
- The class of DCFLs is closed under Intersection with Regular set.
  If $L_1$ = DCFL and $L_2$ = Regular language

  Then $L_1 \cap L_2$ is DCFL
- The class of DCFLs is closed under Inverse Homomorphism.
- DCFLs are not closed under the following operations:
  1) Union
  2) Concatenation
  3) Kleene closure
  4) Homomorphism
  5) Intersection
  6) Substitution
  7) Reversal
  8) $\in$-free Homomorphism

**Rack Your Brain**

Is the language $L = \{a^nb^m | m > n + 2\}$ is deterministic?

**Grey Matter Alert!**

- If a Grammar G is an LL(K) grammar, then L(G) is a deterministic context free language.
- A grammar is an LL(K) grammar if we can uniquely identify the correct production, given the currently scanned symbol and a 'look-ahead' of the next K-1 symbols.

**Previous Years' Question**

If $L_1$ and $L_2$ are context free languages and R a regular set, one of the languages below is not necessarily a context free language? **(GATE -1996)**

**Which one?**

1) $L_1.L_2$ 2) $L_1 \cap L_2$ 3) $L_1 \cap R$ 4) $L_1 \cup L_2$

**Sol: Option 2)**

**Previous Years' Question**

Let L be a context free language and M a regular language. Then the language $L \cap M$ is: **(GATE-2006)**

**1)** always regular

**2)** never regular

**3)** always a deterministic context free language

**4)** always a context free language

**Sol: Option 4)**

**Decision properties of CFL's**

The following problems are decidable under CFL's:

**1)** Emptiness problem

**2)** Finiteness problem

**3)** Membership problem

**Emptiness problem:**

Given a context-free grammar G = (V, T, S, P), there exists an algorithm for deciding whether L (G) is empty or not; hence this is a decidable problem.

$E_{CFG}$ = { <G> | G is a CFG and L(G) = $\phi$ }

**Finiteness problem:**

Given a context-free grammar G = (V, T, S, P), there exists an algorithm for determining whether L(G) is infinite or not.

**Membership problem:**

Given a context-free grammar G = (V, T, S, P), there exists an algorithm, for deciding whether a given word w is in the language defined by a context-free grammar.

**Proof:**

- Start by converting whatever representation of L is given into a CNF grammar for L.
- As the parse trees of a Chomsky-normal form grammar are binary trees, if w is of length n then there will be exactly 2n-1 nodes labelled by variables in the tree.
- The number of possible trees and node-labellings is thus only exponential in n. List them all and check if any of them yields w.

**Note:**

There is much more efficient technique based on the idea of dynamic programming, known as table filling algorithm or tabulation.
This algorithm is CYK algorithm; used to test membership in a context free language.

**Grey Matter Alert!**

The Cocke-Younger-Kasami (CYK) algorithm tells whether a given string is in a given context free language. For a fixed CFL, this test takes time $O(n^3)$, if n is the length of the string being tested.

**Undecidable CFL problems**

**1)** Is a given CFG G ambiguous?
**2)** Is a given CFL inherently ambiguous?
**3)** Is the intersection of two CFL's empty?
**4)** Are two CFL's the same?
Is a given CFL equal to $\Sigma^*$, where $\Sigma$ is the alphabet of this language?

**Rack Your Brain**

Does L(G) contain atleast 100 strings, for a given CFG G?
Is the given problem is decidable?

**Previous Years' Question**

Which of the following problems is undecidable? **(GATE 2014 (Set 3))**

**1)** Deciding if a given context free grammar is ambiguous.

**2)** Deciding if a given string is generated by a given context free grammar.

**3)** Deciding if the language generated by a given context free grammar is empty.

**4)** Deciding if the language generated by a given context free grammar is finite.

**Sol: Option 1)**

**Rack Your Brain**

Which of the following problem is undecidable?

**a)** Completeness problem for CFGs

**b)** Equivalence problem for CFGs

**Pumping lemma for context-free language:**

Let L be an infinite context-free language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as

$w = uvxyz,$

with $|vxy| \leq m,$

and $|vy| \geq 1$

Such that

$uv^i xy^i z \in L$

for all i = 0, 1, 2, ...

This is known as the pumping lemma for context-free languages.

**Note:**

Pumping lemma for context free language are used negatively to show that a given language does not belong to context free language family.

Show that the language:

**Example:** $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free language.

**Sol.:** Let's assume L is context-free language

$\exists\, m \geq 1$

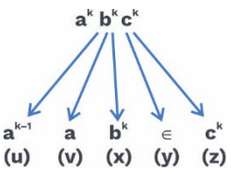
such that

$w \in L$ with $|w| \geq m$

$w = a^k b^k c^k$

Here $3k \geq m$

Now, split $a^k b^k c^k$ in uvxyz.




$|vxy| = k + 1 \leq m$

Hence, $uv^ixy^iz \in L \ \forall \ i = 0, 1, 2, \ldots$

for $i = 2$, $a^{k-1} a^2 b^k \in c^k$

$\Rightarrow$ $a^{k+1} b^k c^k$ which does not belong to given language

$\therefore$ $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free language.

**Note:**

If $\Sigma = \{a\}$ i.e. singleton set and language L is defined over L then Lis context free language if and only if the length of strings in L are in arithmetic progression.

**eg. 1:** $L = \{a^{3n-1} \mid n \geq 1\}$
$=\{aaaa, aaaaaaa, \ldots\ldots\}$
context free language. **(Following arithmetic progression)**

**eg. 2:** $L = \{a^p \mid p$ is a prime$\}$
$=\{aa, aaa, aaaaa, \ldots\ldots\}$
Not context free language. **(Not following arithmetic progression)**

**eg. 3:** $L = \{a^{n^2} \mid n \geq 1\}$
$=\{a, aaaa, \ldots\ldots\}$
Not context free languages **(Not following arithmetic progression)**

**eg.4:** $L = \{a^{n!} \mid n \geq 1\}$
$=\{a, aa, aaaaaa, \ldots\ldots\}$
Not context free langauge **(Not following arithmetic progression)**

## 4.3 CSL AND GRAMMAR

**Context-sensitive grammar:**

**Definition**

A grammar G = (V, T, S, P) is said to be context sensitive if all productions are of the form,

$$x \rightarrow y,$$

where $x, y \in (VUT)^*$ and $|x| \geq |y|$

**Grey Matter Alert!**

The definition shows clearly that this grammar is non contracting in the sense that the length of successive sentential forms can never decrease.
All such grammars can be rewritten in a normal form in which all productions are of the form:

$xAy \rightarrow xvy$

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 stringy on the right.

eg. : $S \rightarrow abc/aAbc$

$Ab \rightarrow bA$

$Ac \rightarrow Bbcc$

$bB \rightarrow Bb$

$aB \rightarrow aa/aaA$ is an example of context sensitive grammar.

**Context-sensitive languages:**

The language generated by the context-sensitive grammar is known as context-sensitive languages.

**Definition**

A language L is said to be context sensitive if there exists a context sensitive grammar G such that

$L = L(G)$ or, $L = L(G) \cup \{\in\}$

Here, in this definition, we reintroduce the empty string:

- Definition of context-sensitive grammar implies that $x \rightarrow \in$ is not

allowed so that a context-sensitive grammar can never generate a language containing the empty string.

- Every context-free language without $\in$ can be generated by a special case of a context-sensitive grammar, say by one in Chomsky or Greibach's normal form. Both satisfy the condition of context-free grammar defined earlier.

**Note:**

By including the empty string in the definition of a context sensitive language. We can claim that the family of context free language is a subset of the family of context sensitive languages.

**Example 1:** The language $L = \{a^n b^n c^n \mid n \geq 1\}$ is a Context-Sensitive language.

Context-Sensitive grammar exists for this language is:

$S \rightarrow abc|aAbc$
$Ab \rightarrow bA$
$Ac \rightarrow Bbcc$
$bB \rightarrow Bb$
$aB \rightarrow aa|aaA$

Look at a derivation of $a^3b^3c^3$.

$S \Rightarrow aAbc$
$\Rightarrow abAc$
$\Rightarrow abBbcc$
$\Rightarrow aBbbcc$
$\Rightarrow aaAbbcc$
$\Rightarrow aabAbcc$
$\Rightarrow aabbAcc$
$\Rightarrow aabbBbccc$
$\Rightarrow aabBbbccc$
$\Rightarrow aaBbbbccc$
$\Rightarrow aaabbbccc$

**Note:**

The language in the previous example is not context free, we can say that the family of context free languages is a proper subset of the family of context sensitive languages.

**Example 2 :** The language $L=\{a^n b^m c^n d^m \mid n,m \geq 1\}$ is a Context-Sensitive language. Context-Sensitive grammar exists for this language is:

S → aAB | aB
A → aAX | aX
B → bBd | bYd
Xb → bX
XY → Yc
Y → c

Derivation of string aabbbccddd as follows:

S ⇒ aAB
⇒ aaXB
⇒ aaXbBd
⇒ aaXbbBdd
⇒ aaXbbbYddd
⇒ aabXbbYddd
⇒ aabbXbYddd
⇒ aabbbXYddd
⇒ aabbbYcddd
⇒ aabbbccddd

Example of Context-Sensitive languages

$L_1 = \{a^n b^n c^n \mid n \geq 1\}$
$L_2 = \{a^{n!} \mid n \geq 0\}$
$L_3 = \{a^n \mid n = m^2, m \geq 1\}$
$L_4 = \{a^n \mid n \text{ is a prime number}\}$
$L_5 = \{a^n \mid n \text{ is not a prime number}\}$
$L_6 = \{ww \mid w \in (a,b)^+\}$
$L_7 = \{w^n \mid w \in (a,b)^+, n \geq 1\}$
$L_8 = \{www^R \mid w \in \{a,b\}^+\}$

**Rack Your Brain**

Find context sensitive grammars for the languages:

**a)** $L = \{w \mid n_a(w) = n_b(w) = n_c(w)\}$

**b)** $L = \{w \mid n_a(w) = n_b(w) < n_c(w)\}$

**Theorem:**
For every Context-Sensitive language L not including $\in$, there exists some linear bounded automaton M such that L = L(M).

**Theorem:**
If a language L is accepted by some linear bounded automaton M, then there exists a Context-Sensitive grammar that generates L.

**Relation between recursively enumerable and context-sensitive languages:**
Every Context-Sensitive language is accepted by some Turing machine and is therefore recursively enumerable.

**Theorem:**
Every Context-Sensitive language is recursive.

**Theorem:**
There exists a recursive language that is not Context-Sensitive.

**Note:**

- There exists a recursive language that is not context-sensitive. This theorem indicates that linear bounded automata are indeed less powerful than Turing Machines. Since, they accept only a proper subset of the recursive languages.
- It follows from the same result that linear bounded automata are more powerful than Pushdown Automata.

**Note:**

- Context free languages, being generated by context free grammars, are a subset of the context-sensitive languages.
- Any language accepted by a Pushdown Automata is also accepted by some linear bounded automaton but there are languages accepted by some linear bounded automata for which there are no Pushdown Automata.

**Previous Years' Question**

Which one of the following language over $\Sigma = \{a, b\}$ is **NOT** context free?

**(GATE (CSE) -2019)**

**1)** $\{ww^R \mid W \in \{a, b\}^*\}$

**2)** $\{wa^n b^n w^R \mid w \in \{a, b\}^*, n \geq 0\}$

**3)** $\{wa^n w^R b^n \mid w \in \{a, b\}^*, n \geq 0\}$

**4)** $\{a^n b^i \mid i \in \{n, 3n, 5n\}, n \geq 0\}$

**Sol: Option 3)**

**Closure properties of context-sensitive language:**

The family of Context-Sensitive languages is closed under

**1)** Union.
**2)** Intersection.
**3)** Complementation.
**4)** Concatenation
**5)** Kleene star.
**6)** Reversal.
**7)** Inverse Homomorphism.

**Note:**

The family of context sensitive language is not closed under Homomorphism.

**Closure properties of DCFL, CFL and CSL:**

| Property | DCFL | CFL | CSL |
|---|---|---|---|
| Union | No | Yes | Yes |
| Intersection | No | No | Yes |
| Set-difference | No | No | Yes |
| Complement | Yes | No | Yes |
| Concatenation | No | Yes | Yes |

| | | | |
|---|---|---|---|
| Kleene closure | No | Yes | Yes |
| Homomorphism | No | Yes | No |
| Reversal | No | Yes | Yes |
| Substitution (∈-free) | No | Yes | Yes |
| Inverse homomorphism | Yes | Yes | Yes |
| Intersection with a regular language | Yes | Yes | Yes |

**Table 4.2 Closure Properties**

- **Context-free grammar:** A grammar G = (V, T, S, P) is said to be context-free if all productions in P have the form:

$$A \rightarrow x$$

where $A \in V$ and $x \in (VUT)^*$

- **Derivation of string:** Left Most derivation and Right Most Derivation.
- A derivation is said to be left most if in each step, the leftmost variable in the sentential form is replaced.
- A derivation is said to be right most if in each step, the rightmost variable in the sentential form is replaced.
- **Ambiguity in grammar:** A Context-Free Grammar G is said to be ambiguous if there exists some $w \in L(G)$ that has at least two distinct derivative trees (parse tree).

**Conversion of CFG into simplified context-free grammar:**

A safe order is:

**i)** Eliminate ∈ - productions
**ii)** Eliminate Unit - productions
**iii)** Eliminate useless symbol

- **Chomsky normal form:** A Context-Free Grammar is in Chomsky Normal Form if every rule is of the form: $A \rightarrow BC$ (OR) $A \rightarrow a$
where A, B, C are in variables and a is any terminal.
- We can convert Context-Free Grammar to Chomsky Normal Form as well as to Greibach Normal Form.
- **Pushdown automata:** The automaton which defines the Context-Free languages is known as Pushdown Automata.

## Chapter Summary

- **Deterministic and non-deterministic pushdown automata:** Non-Deterministic Pushdown Automata is more powerful than Deterministic Pushdown Automata.
- **Closure properties of CFLs:** The Context-Free Languages are closed under: **1)** Union **2)** Concatenation **3)** Kleene-Closure **4)** Reversal.
- **Decision properties of CFLs:** Following problems are decidable under CFLs: Emptiness problem, Finiteness problem, Membership problem
- **Context-sensitive languages:** Language generated by Context-Sensitive Grammar is known as Context-Sensitive Languages.