St. Joseph’s College of Engineering & Technology Palai

Department of Computer Science & Engineering

S4 CS

CS010 406 Theory of Computation

Module 3

Theory of Computation - Module 3

Syllabus

Context Free Grammar –Simplification of CFG-

Normal forms-Chomsky Normal form and Greibach Normal form-

pumping lemma for Context free languages-

Applications of PDA -

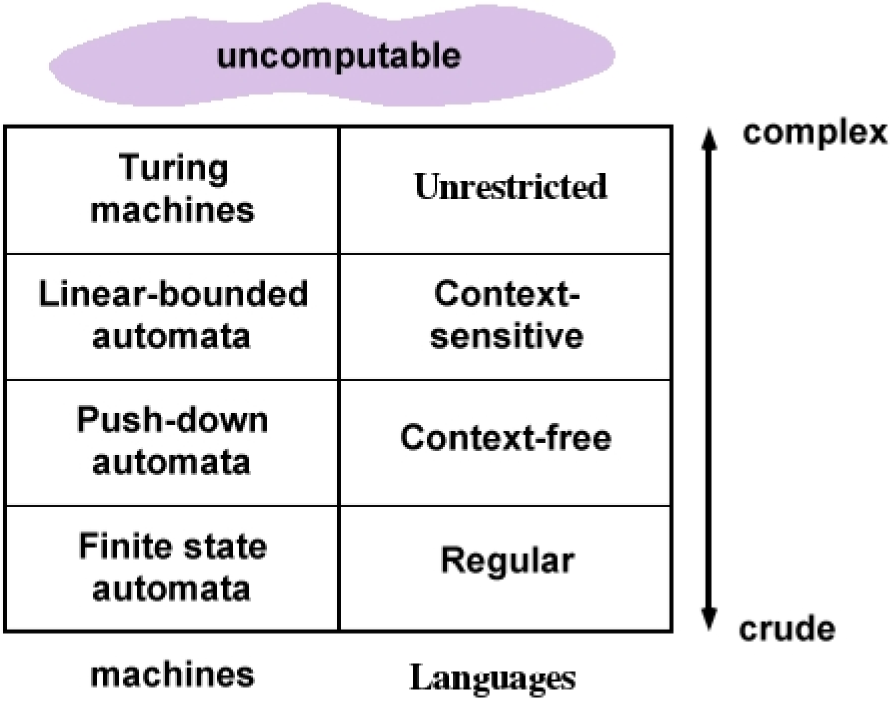
Pushdown Automata – Formal definition – Language acceptability by PDA through empty stack and final state –

Deterministic and nondeterministic PDA – designing of PDA-

Contents

Context Free Grammars and Languages

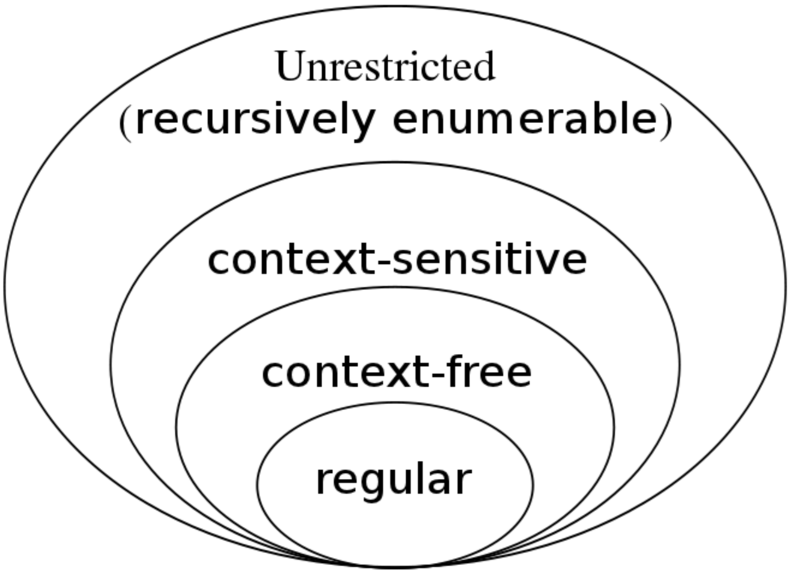
As we learned in first module, according to Chomsky classification, Type- 2 languages are also called context free languages. They are generated using context free grammars. They are recognised using push down automata.



machines	Languages
Turing machines	Unrestricted
Linear-bounded automata	Context-sensitive
Push-down automata	Context-free
Finite state automata	Regular

uncomputable
complex
crude

Also regular languages are a subset of context free languages.



Part I. Context Free Grammars (CFG)

Context free grammar, G is defined as,

$G = (V, \sum, P, S)$

where

V is a set of non-terminals,

$\sum$ is a set of terminals,

S is the start symbol,

P is a set of productions.

A grammar, G is context free, if productions are of the form,

$\alpha \longrightarrow \beta$

where α is a single non terminal.

Every regular grammar is context free.

Example 1:

An example for context free grammar is shown below:

$S \longrightarrow ASA|BSB|a|b$

$A \longrightarrow a$

$B \longrightarrow b$

where S is the start symbol.

From the definition of CFG,

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

$\sum = \{a, b\}$

$P = \{S \longrightarrow ASA|BSB|a|b,\ A \longrightarrow a,\ B \longrightarrow b\}$

$S = S$

The production,

$S \longrightarrow ASA|BSB|a|b$

can also be written as,

$S \longrightarrow ASA$

$S \longrightarrow BSB$

$S \longrightarrow a$

$S \longrightarrow b$

Example 2:

The following is an example for context free grammar.

Stmt ⟶ id = Expr;

Stmt ⟶ if (Expr) Stmt

Stmt ⟶ if (Expr) Stmt else Stmt

Stmt ⟶ while (Expr) Stmt

Stmt ⟶ do Stmt while (Expr);

Stmt ⟶ { Stmts }

Stmts $\longrightarrow$ Stmts Stmt| ε

where $Stmts$ is the start symbol.

From the definition of CFG,

$G = \{Stmts, stmt, Expr\}$

$\sum = \{id, =, ; , (,), if, else, while, do\}$

P is the set of productions given above

$S = Stmts$

Note that above CFG corresponds to some statements in C programming language.

1 Language of Context Free Grammar (Context Free Language)

A language, L of G is a set of strings that can be derived from G.

Example 1:

Consider the following context free grammar,

$S \longrightarrow aA|bB$

$A \longrightarrow x$

$B \longrightarrow y$

The language corresponding to the above CFG is {ax,by}

A string w belongs to a context free language (CFL), L, if it can be derived from the CFG associated with L.

That is,

$L(G) = \{w \in \sum \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$

There are two approaches to check whether a string belongs to a CFL. They are,

Derivations, and

Reductions.

Derivations

It is a mechanism to check whether a string w belongs to a context free language (CFL).

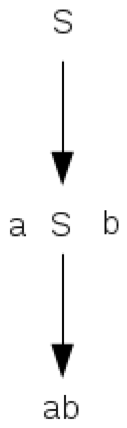
Example 1:

Consider the following CFG,

$S \longrightarrow aSb \mid ab$

where S is the start symbol.

Check whether the string $aabb$ belongs to the CFL corresponding to above CFG.



Above diagram is called a derivation tree (parse tree).

Here, by performing two derivations, we got a^2b^2. So a^2b^2 belongs to the above CFL.

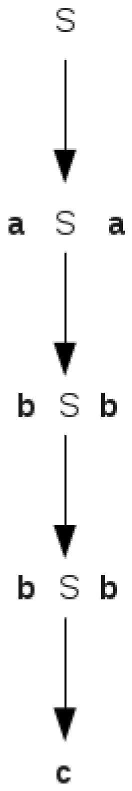
Example 2:

Consider the following CFG,

$S \longrightarrow aSa \,|\, bSb \,|c$

where S is the start symbol.

Check whether the string abbcbba belongs to the language corresponding to the above grammar.



From the above derivation tree, the string abbcbba is derived using the given CFG. So the string abbcbba belongs to the above language.

Example 3:

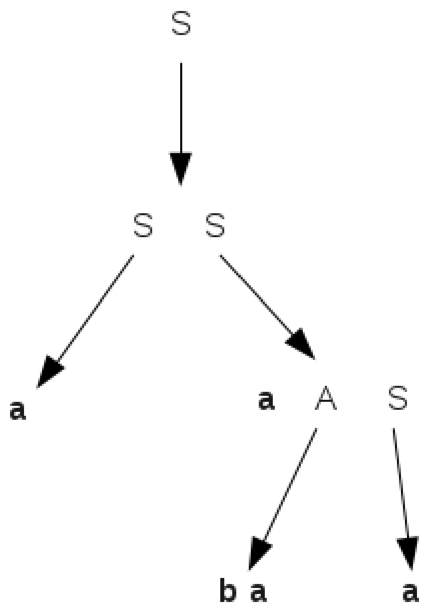
Consider the CFG,

$S \longrightarrow aAS|a|SS$

$A \longrightarrow SbA|ba$

where S is the start symbol.

Check whether the string aabaa can be derived using the above grammar.



Example 4:

Consider the CFG,

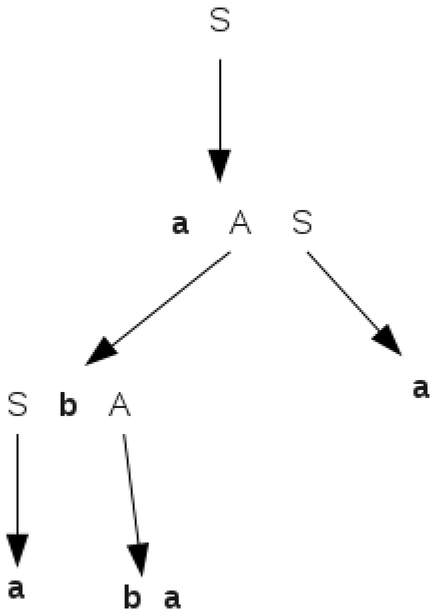
$S \longrightarrow aAS|a$

$A \longrightarrow SbA|SS|ba$

where S is the start symbol.

Check whether the string aabbaa belongs to this language.

Following is the derivation tree:



Thus the above string $aabbaa$ or $a^2b^2a^2$ belongs to the above CFL.

Reductions

This approach can also be used to check whether a string belongs to a context free language. Here, the string w is taken and an inverted tree is made. this is done by performing a number of reductions starting from the given string using the productions of the grammar.

Example 1:

Consider the grammar,

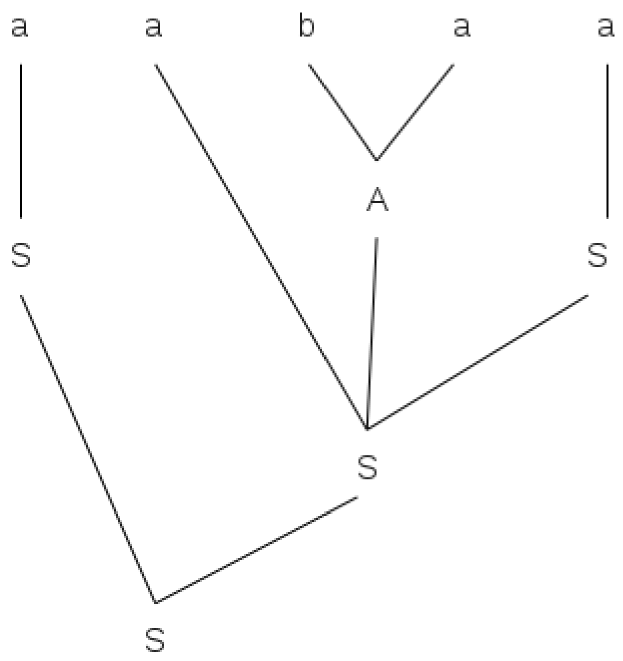
$S \longrightarrow aAS|a|SS$

$A \longrightarrow SbA|ba$

where S is the start symbol.

Check whether the string aabaa belongs to the CFL associated with the above grammar.

The reduction tree is shown below:



Thus by performing a number of reductions, we finally reached the start symbol. So the string aabaa belongs to the above CFL.

2 CFG corresponding to a CFL

Example 1:

Design a CFG for the language,

$L = \{a^n b^n | \, n \geq 0\}$

At n=0, ε is a valid string for the above language. The production for this is,

$S \longrightarrow \varepsilon$

Every string is of length 2n. The strings are ab, abab, aaabbb, ... etc..

The corresponding production is,

$S \longrightarrow aSb.$

Hence the grammar is,

$S \longrightarrow aSb \,|\, \varepsilon$

where S is the start symbol.

Example 2:

Design a CFG for the language,

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

At n=0, ε is a valid string for the above language. The production for this is,

$S \longrightarrow \varepsilon$

It can be seen that there are double the number of b's as a's. The production is,

$S \longrightarrow aSbb$

So the grammar is,

$S \longrightarrow aSbb \mid \varepsilon$

where S is the start symbol.

Example 3:

Design a CFG for the language, L over {0,1} to generate all possible strings of even length.

Consider number 0 as even. The string of length 0 is ε. The corresponding production is,

$S \longrightarrow \varepsilon$

An even length is of length 2n contains one or more repetitions of 00, 01, 10 or 11. The corresponding production is,

$S \longrightarrow 00S|01S|10S|11S$

Thus the grammar is,

$S \longrightarrow 00S|01S|10S|11S|\varepsilon$

Example 4:

Design a CFG for the language, L over {0,1} to generate all the strings having alternate sequence of 0 and 1.

The minimum string length in L is 2. The strings are 01 or 10.

If a string begins with 01, then it should follow a repetition of 01 and terminate with either 01 or 0.

If a string begins with 10, then it should follow a repetition of 10 and terminate with either 10 or 1.

The grammar will be,

$S \longrightarrow 01|10|01A|10B$

$A \longrightarrow 01A|0|01$

$B \longrightarrow 10B|1|10$

Example 6:

Write a CFG for the regular expression,

$a^*b(a|b)^*$

On analysing this regular expression, we can see that the strings start with any number of a's, followed by a 'b' and may end with a combination of a's and b's.

The CFG can be written as,

$S \longrightarrow AbB$

$A \longrightarrow aA|\varepsilon$

$B \longrightarrow aB|bB|\varepsilon$

For example, using the above productions

$$S \Longrightarrow AbB$$
$$\Longrightarrow aAbB$$
$$\Longrightarrow aAbaB$$
$$\Longrightarrow aaAbaB$$
$$\Longrightarrow aaAbabB$$
$$\Longrightarrow aabab$$

aabab is string belonging to the above CFL.

Example 7:

Write a CFG which generates strings of equal number of a’s and b’s.

The CFG is

$S \longrightarrow aSbS|bSaS|\varepsilon$

For example, using the above productions,

$$S \Longrightarrow bSaS$$
$$\Longrightarrow bbSaSaS$$
$$\Longrightarrow bbaSbSaSaS$$
$$\Longrightarrow bbaSbaSbSaSaS$$
$$\Longrightarrow bbababaa$$

Note that above string contains equal number of a’s and b’s.

Example 8:

Design a CFG for the regular expression,

$(a|b)^*$

The CFG is,

$S \longrightarrow aS$

$S \longrightarrow bS$

$S \longrightarrow \varepsilon$

That is,

$S \longrightarrow aS|bS|\varepsilon$

where S is the start symbol.

Example 9;

Design a CFG for the language,

$L(G) = \{ww^R|w \in \{0,1\}^*\}$

The CFG is,

$S \longrightarrow 0S0|1S1|\varepsilon$

where S is the start symbol.

[In the above, w^R means reverse of string w.]

For example, using the above productions,

$$S \Longrightarrow 0S0$$

$$\Longrightarrow 00S00$$

$$\Longrightarrow 001S100$$

$$\Longrightarrow 001100$$

Example 10:

Design a CFG for the language,

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

Case 1: For n>m,

The language is

$L = \{a^n b^m |\, n > m\}$

The CFG is,

$S_1 \longrightarrow AS_1$

$S_1 \longrightarrow aS_1b|\varepsilon$

$A \longrightarrow aA|a$

Case 2: For n<m,

The language is

$L = \{a^n b^m |\, n < m\}$

The CFG is,

$S_2 \longrightarrow S_2B$

$S_2 \longrightarrow aS_2b|\varepsilon$

$B \longrightarrow bB|b$

Combining Case 1 and Case 2, we get,

$S \longrightarrow S_1|S_2$

Thus the CFG is,

$S \longrightarrow S_1|S_2$

$S_1 \longrightarrow AS_1$

$S_1 \longrightarrow aS_1b|\varepsilon$

$A \longrightarrow aA|a$

$S_2 \longrightarrow S_2B$

$S_2 \longrightarrow aS_2b|\varepsilon$

$B \longrightarrow bB|b$

where S is the start symbol.

Example 11:

Design a CFG for the language,

$L = \{\,(0^n1^n|\,n \geq 0)\;\cup\;(1^n0^n|\,n \geq 0)\,\}$

We can write it as,

$L = L_1 \cup L_2$

Case 1:

Consider L_1,

$L_1 = \{\,0^n1^n|\,n \geq 0\,\}$

The CFG is,

$S_1 \longrightarrow 0S_11|\varepsilon$

Case 2:

Consider L_2,

$L_2 = \{\,1^n0^n|\,n \geq 0\,\}$

The CFG is,

$S_2 \longrightarrow 1S_20|\varepsilon$

Then we have $L = L_1 \cup L_2$

Combining above two cases, we get the CFG as,

$S \longrightarrow S_1|S_2$

Thus the complete CFG is,

$S \longrightarrow S_1|S_2$

$S_1 \longrightarrow 0S_11|\varepsilon$

$S_2 \longrightarrow 1S_20|\varepsilon$

where S is the start symbol.

Example 12:

Design a CFG for the language, L

$L = \{\,a^nb^{2n}|\,n \geq 0\,\}$

The CFG is,

$S \longrightarrow aSbb|\varepsilon$

Example 13;

Write a CFG for the language,

$L = \{\, a^{2n}b^{m} |\, n > 0,\, m \geq 0\,\}$

From the above we can say that L is the set of strings starting with even number of a's followed by any number of b's.

There is no 'a' in the string after first 'b' is encountered. Also there is no 'b' before any 'a' in the string.

The CFG is,

$S \longrightarrow aaAB$

$A \longrightarrow aaA|\varepsilon$

$B \longrightarrow bB|\varepsilon$

where S is the start symbol.

Example 13:

Write a CFG for the language,

$L = \{a^{n}b^{2n}c^{m} |\, n, m \geq 0\}$

This means strings start with 'a' or 'c', but not with a 'b'.

If the string starts with 'a', then number of a's must follow b's, and the number of b's is twice than number of a's.

If the string starts with 'c', it is followed by any number of c's.

There is no 'a' after any 'b' or any 'c'.

There is no 'b' after any 'c'.

There is no 'b' or 'c' after any 'a'.

There is no 'c' after any 'b'.

Thus the CFG is,

$S \longrightarrow AB$

$A \longrightarrow aAbb|\varepsilon$

$B \longrightarrow cB|\varepsilon$

where S is the start symbol.

Example 14:

Design a CFG for the language,

$L = \{a^{n}b^{m}c^{2m} |\, n, m \geq 0\}$

The CFG is

$S \longrightarrow AB$

$A \longrightarrow aA|\varepsilon$

$B \longrightarrow bBcc|\varepsilon$

Example 15:

Design a CFG for the language, $L = \{wcw^{R}|w \in (a, b)^{*}\}$

The CFG is

$$S \longrightarrow aSa$$

$$S \longrightarrow bSb$$

$$S \longrightarrow c$$

where S is the start symbol.

Part II. Simplification of Context Free Grammars

3 Simplification of CFGs

We can simplify a CFG and produce an equivalent reduced CFG. This is done by,

a. Eliminating useless symbols,

b. Eliminating ε productions,

c. Eliminating unit productions.

3.1 Eliminating Useless Symbols

Useless symbols are those non-terminals or terminals that do not appear in any derivation of a string.

A symbol, Y is said to be useful if:

a. $Y \overset{*}{\Longrightarrow} w,$

that is Y should lead to a set of terminals. Here Y is said to be 'generating'.

b. If there is a derivation,

$$S \overset{*}{\Longrightarrow} \alpha Y \beta \overset{*}{\Longrightarrow} w,$$

then Y is said to be 'reachable'.

Thus a symbol is useful, if it is 'generating' and 'useful'.

Thus a symbol is useless, if it is not 'generating' and not 'reachable'.

Example 1:

Consider the CFG,

$$S \longrightarrow AB|a$$

$$A \longrightarrow b$$

where S is the start symbol.

Eliminate uselss symbols from this grammar.

By observing the above CFG, it is clear that B is a non generating symbol. Since A derives 'b', S derives 'a' but B does not derive any string 'w'.

So we can eliminate $S \longrightarrow AB$ from the CFG.

Now CFG becomes,

$S \longrightarrow a$

$A \longrightarrow b$

Here A is a non reachable symbol, since it cannot be reached from the start symbol S.

So we can eliminate the production, $A \longrightarrow b$ from the CFG.

Now the reduced grammar is,

$S \longrightarrow a$

This grammar does not contain any useless symbols.

Example 2:

Consider the CFG,

$S \longrightarrow aB|bX$

$A \longrightarrow BAd|bSX|a$

$B \longrightarrow aSB|bBX$

$X \longrightarrow SBD|aBx|ad$

where S is the start symbol.

Eliminate uselss symbols from this grammar.

First we choose those non terminals which derive to the strings of terminals.

The non terminals,

A derives to the terminal 'a';

X derives to the terminals 'ad'.

So A and X are useful symbols.

Since,

$S \longrightarrow bX$, and X is a useful symbol, S is also a useful symbol.

From the production, $B \longrightarrow aSB|bBX$,

B does not derive any terminal string, B is a non-generating symbol. So eliminate those productions containing B.

Grammar becomes,

$S \longrightarrow bX$

$A \longrightarrow bSX|a$

$X \longrightarrow ad$

From the above CFG, A is a non- reachable symbol, since A cannot be reached from the start symbol, S.

So eliminate the production, $A \longrightarrow bSX|a$.

Now the CFG is,

$S \longrightarrow bX$

$X \longrightarrow ad$

This grammar does not contain any useless symbols.

Example 3:

Consider the CFG,

$A \longrightarrow xyz|Xyzz$

$X \longrightarrow Xz|xYz$

$Y \longrightarrow yYy|Xz$

$Z \longrightarrow Zy|z$

where A is the start symbol.

Eliminate uselss symbols from this grammar.

From the productions,

$A \longrightarrow xyz$, $Z \longrightarrow z$,

A and Z derive to the strings of terminals. So A and Z are useful symbols.

X and Y do not lead to a string of terminals; that means X and Y are useless symbols.

Eliminating the productions of X and Y, we get,

$A \longrightarrow xyz$

$Z \longrightarrow Zy|z$

From the above, Z is not reachable, since it cannot be reached from the start symbol, A. So eliminate the production corresponding to Z.

The CFG is,

$A \longrightarrow xyz.$

This grammar does not contain any useless symbols.

Example 4:

Consider the CFG,

$S \longrightarrow aC|SB$

$A \longrightarrow bSCa$

$B \longrightarrow aSB|bBC$

$C \longrightarrow aBC|ad$

where S is the start symbol.

Eliminate uselss symbols from this grammar.

Since, $C \longrightarrow ad$,

C is a generating symbol.

Since, $S \longrightarrow aC$,

S is also a useful symbol.

Since, $A \longrightarrow bSCa$,

A is also a useful symbol.

The RHS of $B \longrightarrow aSB|bBC$ contains B. B is not terminating. B is not generating.

So B is a useless symbol. Eliminate those productions containing B.

We get,

$S \longrightarrow aC$

$A \longrightarrow bSCa$

$C \longrightarrow ad$

From the above, A is not reachable, since it cannot be reached from the start symbol, S.

So eliminate the production corresponding to A.

We get the CFG as,

$S \longrightarrow aC$

$C \longrightarrow ad$

where S is the start symbol.

This grammar does not contain any useless symbols.

3.2 Removal of Unit Productions

A unit production is defined as,

$A \longrightarrow B$

where A is a non-terminal, and

B is a non-terminal.

Thus both LHS and RHS contain single non-terminals.

Following examples show how unit productions are removed from a CFG.

Example 1:

Consider the CFG,

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow C|b$

$C \longrightarrow D$

$D \longrightarrow E$

$E \longrightarrow a$

where S is the start symbol.

Eliminate unit productions from this grammar.

From the above CFG, it is seen that,

$B \longrightarrow C$

$C \longrightarrow D$

$D \longrightarrow E$

are unit productions.

To remove the production, $B \longrightarrow C$, check whether there exists a production whose LHS is C and RHS is a terminal. No such production exists.

To remove the production, $C \longrightarrow D$, check whether there exists a production whose LHS is D and RHS is a terminal. No such production exists.

To remove the production, $D \longrightarrow E$, check whether there exists a production whose LHS is E and RHS is a terminal.

There is a production, $E \longrightarrow a$. So remove, $D \longrightarrow E$, and add the production, $D \longrightarrow a$, CFG becomes,

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow C|b$

$C \longrightarrow D$

$D \longrightarrow a$

$E \longrightarrow a$

Now remove the production, $C \longrightarrow D$, and add the production, $C \longrightarrow a$, we get,

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow C|b$

$C \longrightarrow a$

$D \longrightarrow a$

$E \longrightarrow a$

Now remove the production, $B \longrightarrow C$, and add the production, $B \longrightarrow a$, we get,

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow a|b$

$C \longrightarrow a$

$D \longrightarrow a$

$E \longrightarrow a$

where S is the start symbol.

Now the grammar contains no unit productions.

From the above CFG, it is seen that the productions, $C \longrightarrow a$, $D \longrightarrow a$, $E \longrightarrow a$ are useless becuse the symbols C, D and E cannot be reached from the start symbol, S.

By eliminating these productions, we get the CFG as,

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow a|b$

Above is a completely reduced grammar.

Example 2:

Consider the CFG,

$S \longrightarrow a|b|Sa|Sb|S0|S1$

$F \longrightarrow S|(E)$

$T \longrightarrow F|T * F$

$E \longrightarrow T|E + T$

where E is the start symbol.

Eliminate unit productions from this grammar.

From the above CFG,

$F \longrightarrow S$

$T \longrightarrow F$

$E \longrightarrow T$

are unit productions.

The unit production $F \longrightarrow S$ can be removed by rewriting it as, $F \longrightarrow a|b|Sa|Sb|S0|S1$

Now the CFG is,

$S \longrightarrow a|b|Sa|Sb|S0|S1$

$F \longrightarrow a|b|Sa|Sb|S0|S1|(E)$

$T \longrightarrow F|T * F$

$E \longrightarrow T|E + T$

The unit production $T \longrightarrow F$ can be removed by rewriting it as, $T \longrightarrow a|b|Sa|Sb|S0|S1|(E)$

Now the CFG is,

$$S \longrightarrow a|b|Sa|Sb|S0|S1$$

$$F \longrightarrow a|b|Sa|Sb|S0|S1|(E)$$

$$T \longrightarrow a|b|Sa|Sb|S0|S1|(E)|T * F$$

$$E \longrightarrow T|E + T$$

The unit production $E \longrightarrow T$ can be removed by rewriting it as, $E \longrightarrow a|b|Sa|Sb|S0|S1|(E)|T * F$

Now the CFG is,

$$S \longrightarrow a|b|Sa|Sb|S0|S1$$

$$F \longrightarrow a|b|Sa|Sb|S0|S1|(E)$$

$$T \longrightarrow a|b|Sa|Sb|S0|S1|(E)|T * F$$

$$E \longrightarrow a|b|Sa|Sb|S0|S1|(E)|T * F|E + T$$

This is the CFG that does not contain any unit productions.

Example 1:

Consider the CFG,

$$S \longrightarrow A|bb$$

$$A \longrightarrow B|b$$

$$B \longrightarrow S|a$$

where S is the start symbol.

Eliminate unit productions from this grammar.

Here unit productions are,

$$S \longrightarrow A$$

$$A \longrightarrow B$$

$$B \longrightarrow S$$

Here,

$S \longrightarrow A$ generates $S \longrightarrow b$

$S \longrightarrow A \longrightarrow B$ generates $S \longrightarrow a$

$A \longrightarrow B$ generates $A \longrightarrow a$

$A \longrightarrow B \longrightarrow S$ generates $A \longrightarrow bb$

$B \longrightarrow S$ generates $B \longrightarrow bb$

$B \longrightarrow S \longrightarrow A$ generates $B \longrightarrow b$

The new CFG is,

$S \longrightarrow b|a|bb$

$A \longrightarrow a|bb|b$

$B \longrightarrow bb|b|a$

which contains no unit productions.

3.3 Removal of ε- Productions

Productions of the form, $A \longrightarrow \varepsilon$ are called ε -productions.

We can eliminate ε -productions from a grammar in the following way:

If $A \longrightarrow \varepsilon$ is a production to be eliminated, then we look for all productions, whose RHS contains A, and replace every occurrence of A in each of these productions to obtain non ε -productions. The resultant non ε -productions are added to the grammar.

Example 1:

Consider the CFG,

$S \longrightarrow aA$

$A \longrightarrow b|\varepsilon$

where S is the start symbol.

Eliminate epsilon productions from this grammar.

Here, $A \longrightarrow \varepsilon$ is an epsilon production.

By following the above procedure, put ε in place of A, at the RHS of productions, we get,

The production, $S \longrightarrow aA$ becomes, $S \longrightarrow a$.

Then the CFG is,

$S \longrightarrow aA$

$S \longrightarrow a$

$A \longrightarrow b$

OR

$S \longrightarrow aA|a$

$A \longrightarrow b$

Thus this CFG does not contain any epsilon productions.

Example 2:

Consider the CFG,

$S \longrightarrow ABAC$

$A \longrightarrow aA|\varepsilon$

$B \longrightarrow bB|\varepsilon$

$C \longrightarrow c$

where S is the start symbol.

Eliminate epsilon productions from this grammar.

This CFG contains the epsilon productions, $A \longrightarrow \varepsilon$, $B \longrightarrow \varepsilon$.

To eliminate $A \longrightarrow \varepsilon$, replace A with epsion in the RHS of the productions, $S \longrightarrow ABAC$, $A \longrightarrow aA$,

For the production, $S \longrightarrow ABAC$, replace A with epsilon one by one as,

we get,

$S \longrightarrow BAC$

$S \longrightarrow ABC$

$S \longrightarrow BC$

For the production, $A \longrightarrow aA$, we get,

$A \longrightarrow a$,

Now the grammar becomes,

$S \longrightarrow ABAC|ABC|BAC|BC$

$A \longrightarrow aA|a$

$B \longrightarrow bB|\varepsilon$

$C \longrightarrow c$

To eliminate $B \longrightarrow \varepsilon$, replace A with epsion in the RHS of the productions, $S \longrightarrow ABAC$, $B \longrightarrow bB$,

For the production, $S \longrightarrow ABAC|ABC|BAC|BC$, replace B with epsilon as,

we get,

$S \longrightarrow AAC|AC|C$

For the production, $B \longrightarrow bB$, we get,

$B \longrightarrow b$,

Now the grammar becomes,

$S \longrightarrow ABAC|ABC|BAC|BC|AAC|AC|C$

$A \longrightarrow aA|a$

$B \longrightarrow bB|b$

$C \longrightarrow c$

which does not contain any epsilon production.

Example 3:

Consider the CFG,

$S \longrightarrow aSa$

$S \longrightarrow bSb|\varepsilon$

where S is the start symbol.

Eliminate epsilon productions from this grammar.

Here, $S \longrightarrow \varepsilon$ is an epsilon production. Replace the occurrence of S by epsilon, we get,

$S \longrightarrow aa,$

$S \longrightarrow bb$

Now the CFG becomes,

$S \longrightarrow aSa|bSb|aa|bb$

This does not contain epsilon productions.

Example 4:

Consider the CFG,

$S \longrightarrow a|Xb|aYa$

$X \longrightarrow Y|\varepsilon$

$Y \longrightarrow b|X$

where S is the start symbol.

Eliminate epsilon productions from this grammar.

Here, $X \longrightarrow \varepsilon$ is an epsilon production. Replace the occurrence of X by epsilon, we get,

$S \longrightarrow a|b$

$X \longrightarrow Y$

$Y \longrightarrow \varepsilon$

Now the grammar becomes,

$S \longrightarrow a|Xb|aYa|a|b$

$X \longrightarrow Y$

$Y \longrightarrow b|X|\varepsilon$

In the above CFG, $Y \longrightarrow \varepsilon$ is an epsilon production. Replace the ocurrence of Y by epsilon, we get,

$S \longrightarrow aa$

$X \longrightarrow \varepsilon$

Now the grammar becomes,

$S \longrightarrow a|Xb|aYa|a|b|aa$

$X \longrightarrow Y$

$Y \longrightarrow b|X$

This grammar does not contain epsilon productions.

Exercises:

Simplify the following context free grammars:

1.

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow b$

$B \longrightarrow C$

$E \longrightarrow c|\varepsilon$

where S is the start symbol.

2.

$S \longrightarrow aAa$

$A \longrightarrow Sb|bCC|DaA$

$C \longrightarrow abb|DD$

$E \longrightarrow aC$

$D \longrightarrow aDA$

where S is the start symbol.

3.

$S \longrightarrow bS|bA|\varepsilon$

$A \longrightarrow \varepsilon$

where S is the start symbol.

4.

$S \longrightarrow bS|AB$

$A \longrightarrow \varepsilon$

$B \longrightarrow \varepsilon$

$D \longrightarrow a$

where S is the start symbol.

5.

$S \longrightarrow A$

$A \longrightarrow B$

$B \longrightarrow C$

$C \longrightarrow a$

where S is the start symbol.

6.

$S \longrightarrow AB$

$A \longrightarrow a$

$B \longrightarrow C|b$

$C \longrightarrow D$

$D \longrightarrow E$

$E \longrightarrow a$

where S is the start symbol.

7.

$S \longrightarrow ABCD$

$A \longrightarrow a$

$B \longrightarrow C|b$

$C \longrightarrow D$

$D \longrightarrow c$

where S is the start symbol.

8.

$S \longrightarrow aS|AC|AB$

$A \longrightarrow \varepsilon$

$C \longrightarrow \varepsilon$

$B \longrightarrow bB|bb$

$D \longrightarrow c$

where S is the start symbol.

9.

$S \longrightarrow ABC|A0A$

$A \longrightarrow 0A|B0C|000|B$

$B \longrightarrow 1B1|0|D$

$C \longrightarrow CA|AC$

$D \longrightarrow \varepsilon$

where S is the start symbol.

10.

$S \longrightarrow AAA|B$

$A \longrightarrow 0A|B$

$B \longrightarrow \varepsilon$ where S is the start symbol.

Part III. Normal Forms for CFGs

Standard form or normal forms have been developed for context free grammars. Any CFG can be converted to these forms.

The advantages of representing a CFG in this form are,

1. Complexity of the CFG will be less, and

2. It will be easier to implement the CFG.

Two commonly used normal forms for CFGs are,

1. Chomsky normal form (CNF), and

2. Greibach normal form (GNF).

4 Chomsky Normal Form (CNF)

A context free grammar, G is in Chomsky Normal Form, if every production is of the form,

$A \longrightarrow a$,

$A \longrightarrow BC$,

$S \longrightarrow \varepsilon$; for this production, S is the start symbol.

Thus the RHS of a production can be a single terminal, or two non-terminals. Also it can be ε, if LHS is the start symbol.

Example 1:

Following CFG is in Chomsky normal form (CNF):

$S \longrightarrow AB|b|\varepsilon$,

$A \longrightarrow BC|a$,

$B \longrightarrow b$,

$C \longrightarrow c$

where S is the start symbol.

Example 2:

Following CFG is not in Chomsky normal form (CNF):

$S \longrightarrow ASB|AB|b$,

$A \longrightarrow BC|a$,

$A \longrightarrow a$,

$B \longrightarrow bb$

where S is the start symbol.

The above is not in CNF because the productions,

$S \longrightarrow ASB$,

$B \longrightarrow bb$

do not satisfy the conditions of CNF.

4.1 Conversion to CNF

Every CFG can be reduced to CNF. Following examples show how this is done.

Example 1:

Consider the following CFG,

$S \longrightarrow bA|aB,$

$A \longrightarrow bAA|aS|a,$

$B \longrightarrow aBB|bS|b$

where S is the start symbol.

Convert this CFG to CNF.

Consider the production,

$S \longrightarrow bA|aB$

This is not in CNF. So replace 'b' by non-terminal, P and 'a' by non-terminal, Q. Thsi is by adding the productions, $P \longrightarrow b$, $Q \longrightarrow a$.

,Then we get,

$S \longrightarrow PA|QB$

$P \longrightarrow b$

$Q \longrightarrow a$

Now the CFG becomes,

$S \longrightarrow PA|QB$

$P \longrightarrow b$

$Q \longrightarrow a$

$A \longrightarrow PAA|QS|a,$

$B \longrightarrow QBB|PS|b$

In the above the productions,

$A \longrightarrow PAA,$

$B \longrightarrow QBB$

are not in CNF. So replace AA by R and BB by T. this is done by adding the productions, $R \longrightarrow AA$, $T \longrightarrow BB$.

Then we get,

$A \longrightarrow PR,$

$B \longrightarrow QT$

$R \longrightarrow AA,$

$T \longrightarrow BB$

Now the CFG becomes,

$S \longrightarrow PA|QB$

$P \longrightarrow b$

$Q \longrightarrow a$

$A \longrightarrow PR|QS|a,$

$B \longrightarrow QT|PS|b$

$R \longrightarrow AA,$

$T \longrightarrow BB$

where S is the start symbol.

Note that above CFG is in CNF.

Example 2:

Consider the following CFG,

$S \longrightarrow 1A|0B,$

$A \longrightarrow 1AA|0S|0,$

$B \longrightarrow 0BB|1$

where S is the start symbol.

Convert this CFG to CNF.

Consider the production, $S \longrightarrow 1A|0B$,

This is not in CNF. So replace 1 by P and 0 by Q, by adding the productions, $P \longrightarrow 1, Q \longrightarrow 0$,

we get,

$S \longrightarrow PA|QB$

$P \longrightarrow 1$

$Q \longrightarrow 0$

Now the CFG is,

$S \longrightarrow PA|QB$

$P \longrightarrow 1$

$Q \longrightarrow 0$

$A \longrightarrow PAA|QS|0,$

$B \longrightarrow QBB|1$

In the above CFG, the productions,

$A \longrightarrow PAA$

$B \longrightarrow QBB$

are not in CNF.

So replace AA by R and BB by T. Then we get,

$A \longrightarrow PR$

$B \longrightarrow QT$

$R \longrightarrow AA$

$T \longrightarrow BB$

Now the CFG is,

$S \longrightarrow PA|QB$

$P \longrightarrow 1$

$Q \longrightarrow 0$

$A \longrightarrow PR|QS|0,$

$B \longrightarrow QT|1$

$R \longrightarrow AA$

$T \longrightarrow BB$

where S is the start symbol.

Above CFG is in CNF.

Example 3:

Consider the following CFG,

$S \longrightarrow a|b|CSS,$

where S is the start symbol.

Convert this CFG to CNF.

In the above, the production, $S \longrightarrow CSS$,

is not in CNF. So replace SS by P, we get,

$S \longrightarrow CP$

$P \longrightarrow SS$

Now the CFG becomes,

$S \longrightarrow a|b|CP$

$P \longrightarrow SS$

where S is the start symbol.

Above CFG is in CNF.

Example 4:

Consider the following CFG,

$S \longrightarrow abSb|a|aAb,$

$A \longrightarrow bS|aAAb$,

where S is the start symbol.

Convert this CFG to CNF.

Consider the production, $S \longrightarrow abSb|aAb$ is not in CNF.

So replace Ab by P , we get,

$S \longrightarrow abSb|aP$

$P \longrightarrow Ab$

Now the grammar becomes,

$S \longrightarrow abSb|a|aP$

$P \longrightarrow Ab$

$A \longrightarrow bS|aAP$

Now replace Sb by R, we get,

$S \longrightarrow abR|a|aP$

$P \longrightarrow Ab$

$A \longrightarrow bS|aAP$

$R \longrightarrow Sb$

Now replace bR with T, we get

$S \longrightarrow aT|a|aP$

$P \longrightarrow Ab$

$A \longrightarrow bS|aAP$

$R \longrightarrow Sb$

$T \longrightarrow bR$

Now replace aA with U, we get,

$S \longrightarrow aT|a|aP$

$P \longrightarrow Ab$

$A \longrightarrow bS|UP$

$R \longrightarrow Sb$

$T \longrightarrow bR$

$U \longrightarrow aA$

Now replace a with V and b with W, we get,

$S \longrightarrow VT|a|VP$

$P \longrightarrow AW$

$A \longrightarrow WS|UP$

$R \longrightarrow SW$

$T \longrightarrow WR$

$U \longrightarrow VA$

$V \longrightarrow a$

$W \longrightarrow b$

where S is the start symbol.

The above CFG is in CNF.

Exercises:

Convert the following CFGs to CNF.

1.

$S \longrightarrow bA|aB$

$A \longrightarrow bAA|aS|a$

$B \longrightarrow aBB|bS|b$

where S is the start symbol.

2.

$S \longrightarrow aAD$

$A \longrightarrow aB|bAB$

$B \longrightarrow b$

$D \longrightarrow d$

where S is the start symbol.

3.

$S \longrightarrow aAbB$

$A \longrightarrow aA|a$

$B \longrightarrow bB|b$

where S is the start symbol.

4.

$S \longrightarrow aAbB$

$A \longrightarrow Ab|b$

$B \longrightarrow Ba|a$

where S is the start symbol.

5.

$S \longrightarrow aA|bB$

$A \longrightarrow bAA|a$

$B \longrightarrow BBa|b$

where S is the start symbol.

5.

$S \longrightarrow abSb|ab$

where S is the start symbol.

5 Greibach Normal Form (GNF)

A context free grammar, G is in Greibach nromal form, if every production is of the form,

$A \longrightarrow aX$

$S \longrightarrow \varepsilon$, for this production, S should be start symbol.

where a is a single terminal,

X is a set of 0 or more non-terminals.

Example 1:

The following CFG is in Greibach normal form (GNF):

$S \longrightarrow aAB|\varepsilon$

$A \longrightarrow bC$

$B \longrightarrow b$

$C \longrightarrow c$

where S is the start symbol.

Example 2:

The following CFG is not in Greibach normal form (GNF):

$S \longrightarrow aAB$

$A \longrightarrow BbC$

$B \longrightarrow b|\varepsilon$

$C \longrightarrow c$

where S is the start symbol.

The above is not in GNF becuause the productions, $A \longrightarrow BbC$, $B \longrightarrow \varepsilon$ do not satisfy the conditions of GNF.

5.1 Conversion to GNF

Every CFG can be converted to Greibach normal form (GNF).

Following are the steps:

Step 1: Convert the given CFG to CNF.

Rename all non-terminals by $A_1, A_2, ...A_n$, where A_1 is the start symbol.

Step 2: Modify the grammar so that every production is of the form,

$A_i \longrightarrow a\gamma$, or

$A_i \longrightarrow A_j\gamma$, where j>i, and

γ is a set of 0 or more non-terminals.

This can be done by performing a number of substitutions.

Now if a production is of the form,

$A \longrightarrow A\gamma$, it is in left recursive form.

This left recursion can be eliminated by using a new variable, Z.

This is done as follows:

Let there be a production, $A \longrightarrow ADF|Aa|a|b|c$

This can be written as,

$$A \longrightarrow a|b|c|aZ|bZ|cZ$$

$$Z \longrightarrow DF|a|DFZ|aZ$$

In a production leftmost symbol of RHS must be a terminal as,

$A_n \longrightarrow \alpha\gamma,$

otherwise if it is of the form,

$A_n \longrightarrow A_m\gamma$

then replace A_m on RHS of production of A_{m-1} by replacement rule.

Thus in short, to convert a grammar to GNF, first start with grammar in CNF and then apply substitution rule and eliminate left recursion.

Example 1:

Consider the CFG,

$S \longrightarrow AB|BC$

$A \longrightarrow aB|bA|a$

$B \longrightarrow bB|cC|b$

$C \longrightarrow c$

Convert this grammar to Greibach normal form (GNF).

The production, $S \longrightarrow AB|BC$ is not in GNF.

On applying the substitution rule we get,

$S \longrightarrow aBB|bAB|aB|bBC|cCC|bC$

Now the CFG becomes,

$S \longrightarrow aBB|bAB|aB|bBC|cCC|bC$

$A \longrightarrow aB|bA|a$

$B \longrightarrow bB|cC|b$

$C \longrightarrow c$

Example 2:

Consider the CFG,

$S \longrightarrow abaSa|aba$

Convert this grammar to Greibach normal form (GNF).

Replace a by A using the production, $A \longrightarrow a$,

Replace b by B using the production, $B \longrightarrow b$,

we get,

$S \longrightarrow aBASA|aBA$

$A \longrightarrow a$

$B \longrightarrow b$

Now the above CFG is in GNF.

Example 3:

Consider the CFG,

$S \longrightarrow AB$

$A \longrightarrow aA|bB|b$

$B \longrightarrow b$

Convert this grammar to Greibach normal form (GNF).

In the above CFG, the production,

$S \longrightarrow AB$

is not in GNF.

So replace this by,

$S \longrightarrow aAB|bBB|bB$

Now the CFG becomes,

$S \longrightarrow aAB|bBB|bB$

$A \longrightarrow aA|bB|b$

$B \longrightarrow b$

The above CFG is in Greibach normal form.

Example 4:

Consider the CFG,

$S \longrightarrow abSb|aa$

Convert this grammar to Greibach normal form (GNF).

The above production is not in GNF.

So introduce two productions,

$A \longrightarrow a$

$B \longrightarrow b$

Now the CFG becomes,

$S \longrightarrow aBSB|aA$

$A \longrightarrow a$

$B \longrightarrow b$

Above CFG is in GNF.

Example 5:

Consider the CFG,

$S \longrightarrow aSa|bSb|aa|bb$

Convert this grammar to Greibach normal form (GNF).

Introduce two productions,

$A \longrightarrow a$

$B \longrightarrow b$

Then the grammar becomes,

$S \longrightarrow aSP|bSQ|aP|bQ$

$A \longrightarrow a$

$B \longrightarrow b$

Above CFG is in GNF.

Example 6:

Consider the CFG,

$S \longrightarrow AA|a$

$A \longrightarrow SS|b$

Convert this grammar to Greibach normal form (GNF).

Step 1:

Above CFG is in CNF.

Rename the variables, that is,

Rename S, A by A_1, A_2.

Then the grammar becomes,

$A_1 \longrightarrow A_2A_2|a$

$A_2 \longrightarrow A_1A_1|b$

Step 2: In this, productions must be of the form that

the RHS of a production must start with a terminal followed by nonterminals, or

the RHS of a production starts with a higher numbered variable.

The productions,

$A_1 \longrightarrow A_2A_2|a$

$A_2 \longrightarrow b$

satisfy this criterion.

Consider the production,

$A_2 \longrightarrow A_1A_1$

Replace the first A_1 on the RHS by its production.

Then,

$A_2 \longrightarrow A_2A_2A_1|aA_1$

Now the CFG is,

$A_1 \longrightarrow A_2A_2|a$

$A_2 \longrightarrow b$

$A_2 \longrightarrow A_2A_2A_1|aA_1$

That is,

$A_1 \longrightarrow A_2A_2|a$

$A_2 \longrightarrow A_2A_2A_1|aA_1|b$

The production,$A_2 \longrightarrow A_2A_2A_1|aA_1|b$ contains left recursion.

To eliminate left recursion, this is written as,

$A_2 \longrightarrow aA_1|b|aA_1Z|bZ$

$Z \longrightarrow A_2A_1|A_2A_1Z$

Now the CFG is,

$A_1 \longrightarrow A_2A_2|a$

$A_2 \longrightarrow aA_1|b|aA_1Z|bZ$

$Z \longrightarrow A_2A_1|A_2A_1Z$

Consider the production, $A_1 \longrightarrow A_2A_2$

Replace first A_2 on the RHS with its production, we get,

$A_1 \longrightarrow aA_1A_2|bA_2|aA_1ZA_2|bZA_2$

Now the CFG is

$A_1 \longrightarrow aA_1A_2|bA_2|aA_1ZA_2|bZA_2$

$A_2 \longrightarrow aA_1|b|aA_1Z|bZ$

$Z \longrightarrow A_2A_1|A_2A_1Z$

Consider the production, $Z \longrightarrow A_2A_1|A_2A_1Z$

Replace first A_2 on the RHS with its production,

$Z \longrightarrow aA_1A_1|aA_1A_1Z|bA_1|bA_1Z|aA_1ZA_1|aA_1ZA_1Z|bZA_1|bZA_1Z$

Now the CFG is,

$A_1 \longrightarrow aA_1A_2|bA_2|aA_1ZA_2|bZA_2$

$A_2 \longrightarrow aA_1|b|aA_1Z|bZ$

$Z \longrightarrow aA_1A_1|aA_1A_1Z|bA_1|bA_1Z|aA_1ZA_1|aA_1ZA_1Z|bZA_1|bZA_1Z$

Above CFG is in GNF.

Example 8:

Consider the CFG,

$S \longrightarrow AB$

$A \longrightarrow BS|a$

$B \longrightarrow SA|b$

Convert this grammar to Greibach normal form (GNF).

Step 1: The given grammar is in Chomsky normal form.

Rename the variables, that is,

Rename S, A, B by A_1, A_2, A_3.

Then the grammar becomes,

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow A_3A_1|a$

$A_3 \longrightarrow A_1A_2|b$

Step 2: In this, productions must be of the form that

the RHS of a production must start with a terminal followed by nonterminals, or

the RHS of a production starts with a higher numbered variable.

The productions,

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow A_3A_1|a$

$A_3 \longrightarrow b$

satisfy this criterion.

But the production,

$A_3 \longrightarrow A_1A_2$ does not satisfy this.

By applying substitution rule, substitute for A_1 as

$A_3 \longrightarrow A_2A_3A_2$

substitute for A_2 as

$A_3 \longrightarrow A_3A_1A_3A_2|aA_3A_2$

Now the CFG is,

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow A_3A_1|a$

$A_3 \longrightarrow A_3A_1A_3A_2|aA_3A_2|b$

Consider the production,

$A_3 \longrightarrow A_3A_1A_3A_2|aA_3A_2|b$

Eliminating left recursion from the above production, we get,

$A_3 \longrightarrow aA_3A_2B_3|bB_3|aA_3A_2|b$

$B_3 \longrightarrow A_1A_3A_2B_3|A_1A_3A_2$

Now we have the productions,

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow A_3A_1|a$

$A_3 \longrightarrow aA_3A_2B_3|bB_3|aA_3A_2|b$

$B_3 \longrightarrow A_1A_3A_2B_3|A_1A_3A_2$

Now all the A_3 productions start with terminals.

Using substitution rule, replace A_3 in the RHS of the production for A_2.

$A_2 \longrightarrow A_3A_1|a$ becomes,

$A_2 \longrightarrow aA_3A_2B_3A_1|bB_3A_1|aA_3A_2A_1|bA_1|a$

Now the CFG is,

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow aA_3A_2B_3A_1|bB_3A_1|aA_3A_2A_1|bA_1|a$

$A_3 \longrightarrow aA_3A_2B_3|bB_3|aA_3A_2|b$

$B_3 \longrightarrow A_1A_3A_2B_3|A_1A_3A_2$

Using substitution rule, replace A_2 in the RHS of the production for A_1.

$A_1 \longrightarrow aA_3A_2B_3A_1A_3|bB_3A_1A_3|aA_3A_2A_1A_3|bA_1A_3|aA_3$

Now the CFG is,

$A_1 \longrightarrow aA_3A_2B_3A_1A_3|bB_3A_1A_3|aA_3A_2A_1A_3|bA_1A_3|aA_3$

$A_2 \longrightarrow aA_3A_2B_3A_1|bB_3A_1|aA_3A_2A_1|bA_1|a$

$A_3 \longrightarrow aA_3A_2B_3|bB_3|aA_3A_2|b$

$B_3 \longrightarrow A_1A_3A_2B_3|A_1A_3A_2$

Using substitution rule, replace A_1 in the RHS of the production for B_3.

$B_3 \longrightarrow aA_3A_2B_3A_1A_3A_3A_2B_3|aA_3A_2B_3A_1A_3A_3A_2|bB_3A_1A_3A_3A_2B_3|bB_3A_1A_3A_3A_2|aA_3A_2A_1A_3A_3A_2B_3|aA_3A_2$

Now the CFG is,

$A_1 \longrightarrow aA_3A_2B_3A_1A_3|bB_3A_1A_3|aA_3A_2A_1A_3|bA_1A_3|aA_3$

$A_2 \longrightarrow aA_3A_2B_3A_1|bB_3A_1|aA_3A_2A_1|bA_1|a$

$A_3 \longrightarrow aA_3A_2B_3|bB_3|aA_3A_2|b$

$B_3 \longrightarrow aA_3A_2B_3A_1A_3A_3A_2B_3|aA_3A_2B_3A_1A_3A_3A_2|bB_3A_1A_3A_3A_2B_3|bB_3A_1A_3A_3A_2|aA_3A_2A_1A_3A_3A_2B_3|aA_3A_2$

Above CFG is in Greibach normal form.

Exercises:

1.

Consider the CFG,

$A \longrightarrow aBD|bDB|c|AB|AD$

Convert this grammar to Greibach normal form (GNF).

2.

Consider the CFG,

$S \longrightarrow AB$

$A \longrightarrow BS|b$

$B \longrightarrow SA|a$

Convert this grammar to Greibach normal form (GNF).

3.

Consider the CFG,

$E \longrightarrow E + T|T$

$T \longrightarrow T * F|F$

$F \longrightarrow (E)|a$

Convert this grammar to Greibach normal form (GNF).

4.

Consider the CFG,

$S \longrightarrow YY|0$

$Y \longrightarrow SS|1$

Convert this grammar to Greibach normal form (GNF).

5.

Consider the CFG,

$S \longrightarrow XY$

$X \longrightarrow YS|1$

$Y \longrightarrow SX|0$

Convert this grammar to Greibach normal form (GNF).

6.

Consider the CFG,

$S \longrightarrow XY$

$X \longrightarrow 0X|1Y|1$

$Y \longrightarrow 1$

Convert this grammar to Greibach normal form (GNF).

7.

Consider the CFG,

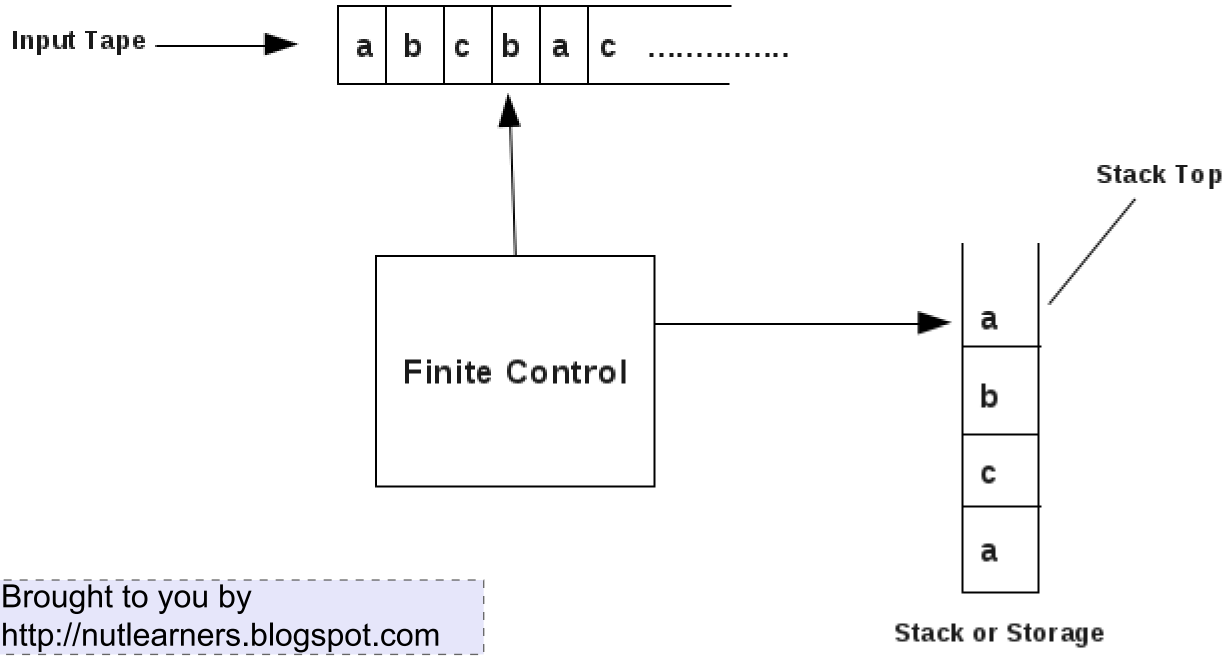
$S \longrightarrow 01S1|11$

Convert this grammar to Greibach normal form (GNF).

Part IV. Pushdown Automata (PDA)

As we learned, context free languages are generated using context free grammars. Context free languages are recognised using pushdown automata.

Following diagram shows a pushdown automation.



The PDA has three components:

An input tape,

A control mechanism, and

A stack.

From the input tape, the finite control reads the input, and also

the finite control reads a symbol from the stack top.

6 Definition of PDA

A pushdown automation, P is a system,

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

where

Q is a set of states,

$\sum$ is a set of input symbols,

Γ is a set of stack symbols (pushdown symbols),

q_0 is the start state,

F is a set of final states,

δ is a transition function which maps,

$$(Q \times \sum^* \times \Gamma^*) \longrightarrow (Q \times \Gamma^*)$$

The meaning of δ is explained below:

Consider a transition function, δ in a PDA defined by,

$$\delta = (p, a, \beta) \longrightarrow (q, \gamma)$$

This means that

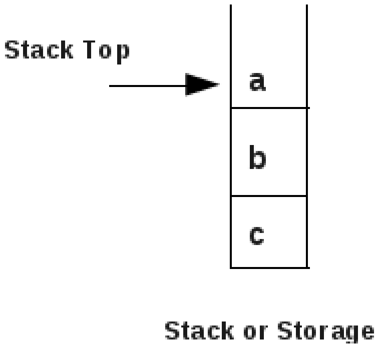
read a from the input tape,

pop the string β from stack top,

move from state p to state q,

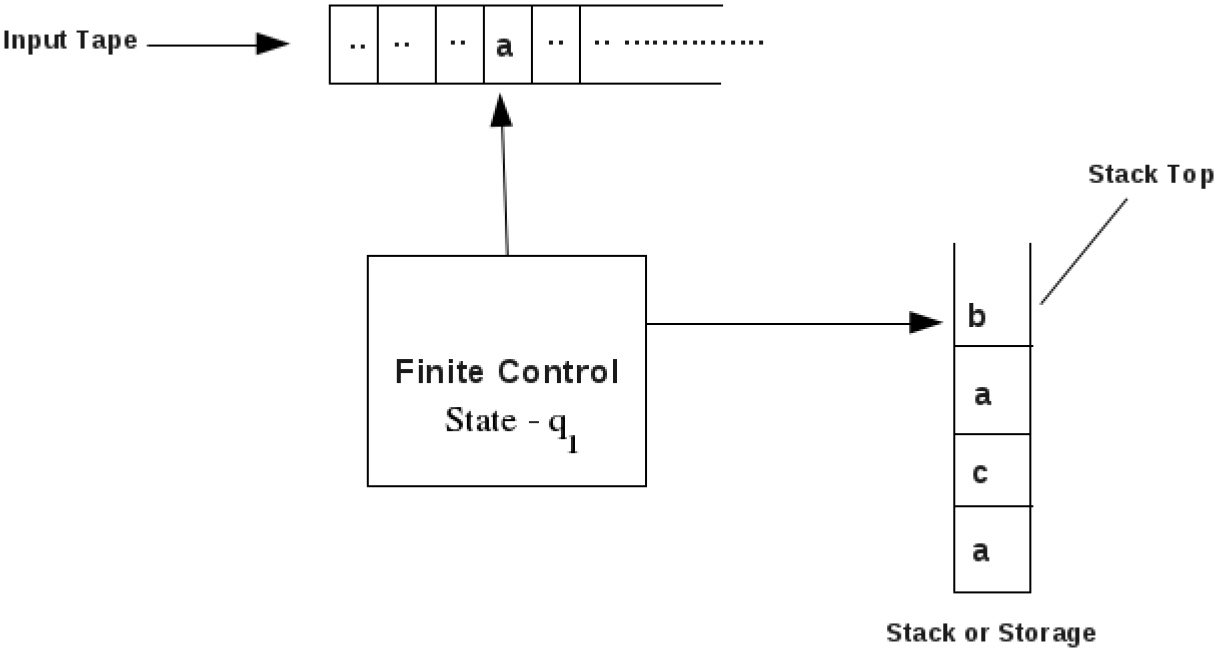
push string γ on to the stack top.

Pushing the string abc on to the stack means,



Example:

Consider the following PDA,



Suppose PDA is currently in state q_1 and the finite control points to state a and the stack contents are as shown above.

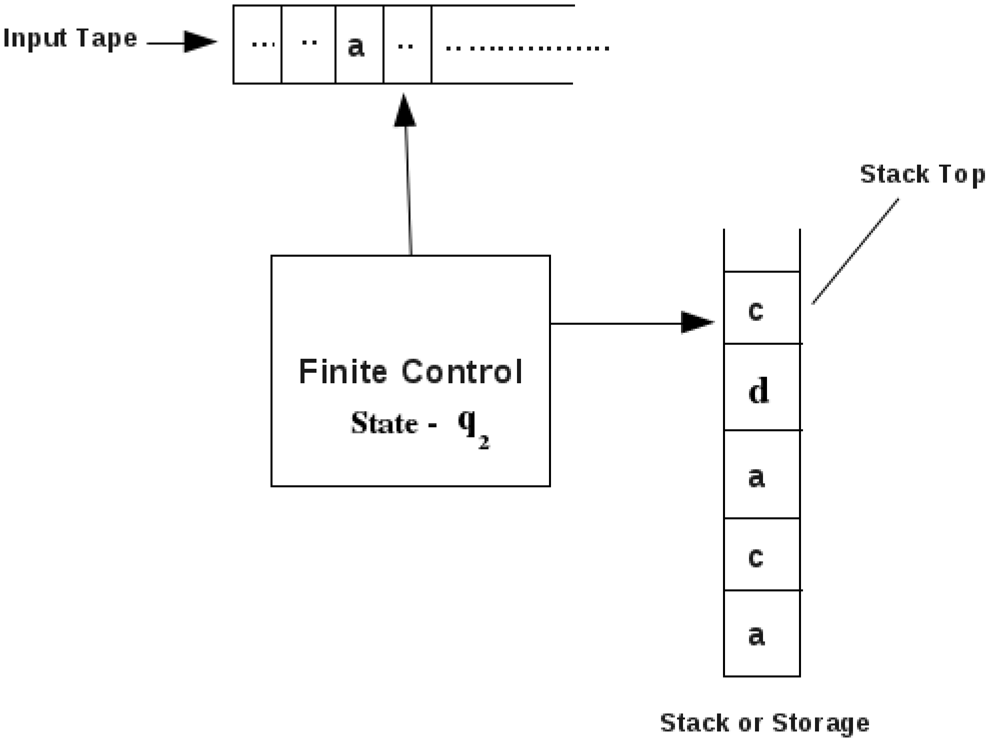
Let a transition is given as,

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

This means PDA currently in state q_1, on reading the input symbol, a and popping b from the stack top changes to state

q_2 and pushes cd on to the stack top.

The new PDA is shown below:



7 Language Acceptability by PDA

Example 1:

Consider a PDA,

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

where

$Q = \{s, q, f\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b, c\}$

$q_0 = \{s\}$

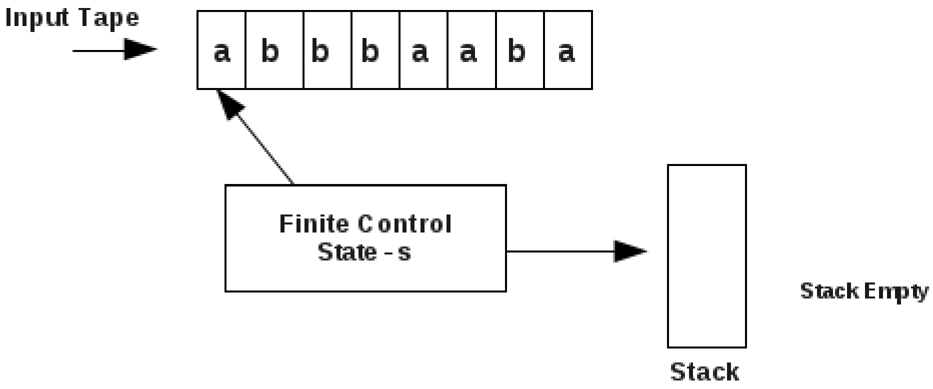
$F = \{f\}$

δ is given as follows:

1. $(s, a, \varepsilon) \longrightarrow (q, a)$
2. $(s, b, \varepsilon) \longrightarrow (q, b)$
3. $(q, a, a) \longrightarrow (q, aa)$
4. $(q, b, b) \longrightarrow (q, bb)$
5. $(q, a, b) \longrightarrow (q, \varepsilon)$
6. $(q, b, a) \longrightarrow (q, \varepsilon)$
7. $(q, b, \varepsilon) \longrightarrow (q, b)$
8. $(q, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

Check whether the string abbbaaba is accepted by the above pushdown automation.

From the above, stack is initially empty, the start state of the PDA is s and PDA points to symbol, a as shown below:



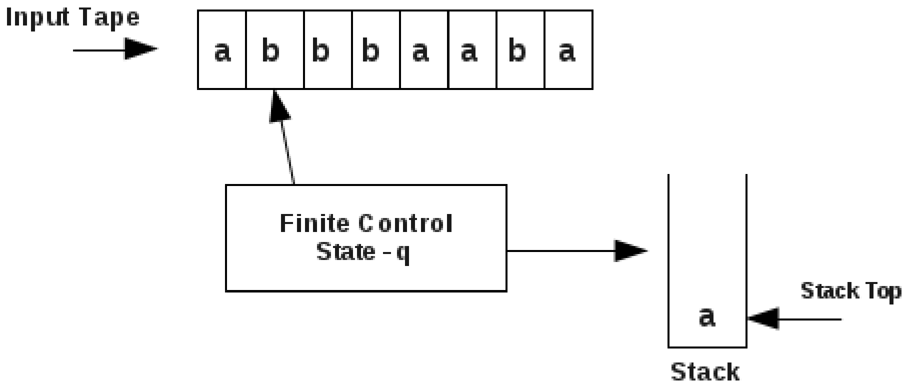
1.

The PDA is in state s, stack top contains symbol ε. Consider the transition,

1. $(s, a, \varepsilon) \longrightarrow (q, a)$

This means PDA in state s, reads a from the tape, pops nothing from stack top, moves to state q and pushes a onto the stack.

PDA now is,



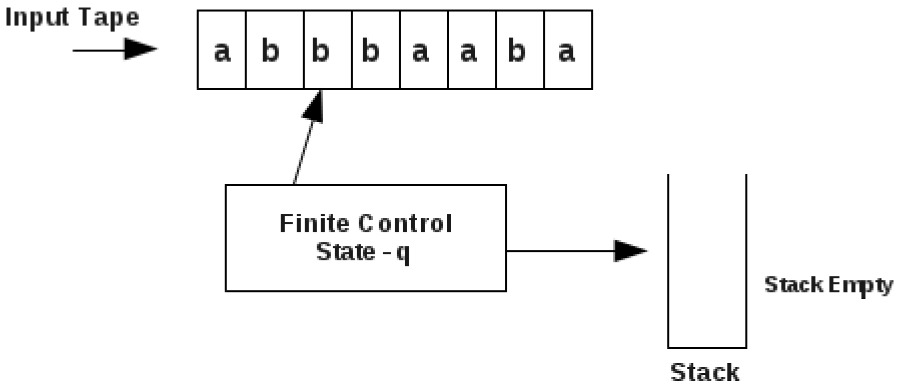
2.

The PDA is in state q, finite control points to symbol b in the input tape, stack top contains symbol a. Consider the transition,

6. $(q, b, a) \longrightarrow (q, \varepsilon)$

This means PDA is in state q, reads b from the input tape, pops a from stack top, moves to state q and pushes nothing onto the stack.

PDA now is,



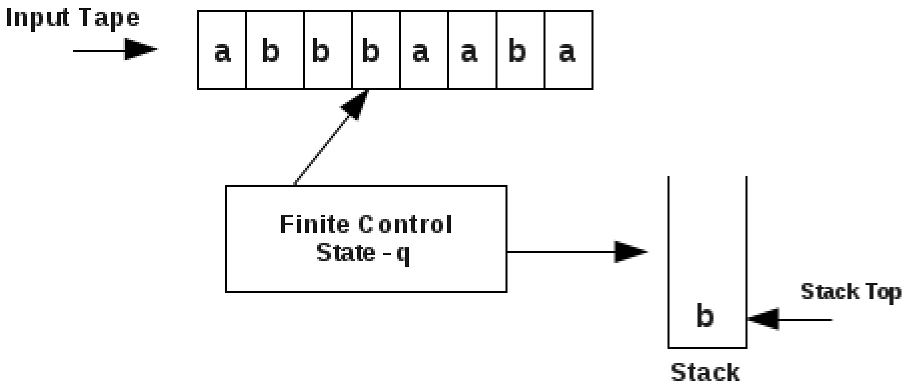
3.

The PDA is in state q, finite control points to symbol b in the input tape, stack top contains symbol ε. Consider the transition,

7. $(q, b, \varepsilon) \longrightarrow (q, b)$

This means PDA is in state q, reads b from the input tape, pops ε from stack top, moves to state q and pushes b onto the stack.

PDA now is,



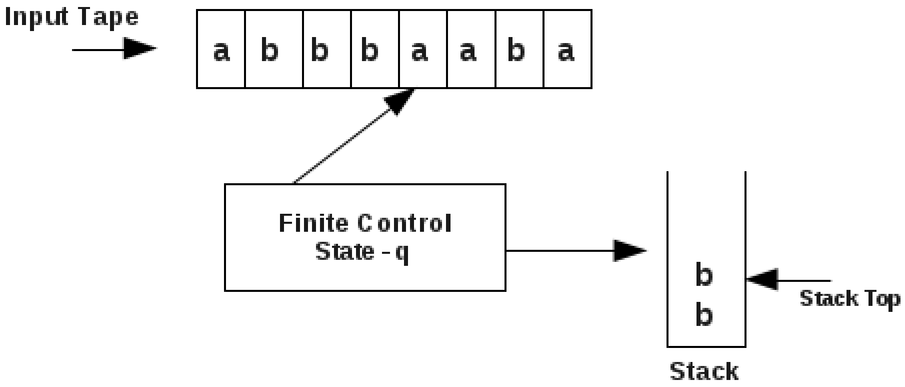
4.

The PDA is in state q, finite control points to symbol b in the input tape, stack top contains symbol b. Consider the transition,

4. $(q, b, b) \longrightarrow (q, bb)$

This means PDA is in state q, reads b from the input tape, pops b from stack top, moves to state q and pushes bb onto the stack.

PDA now is,



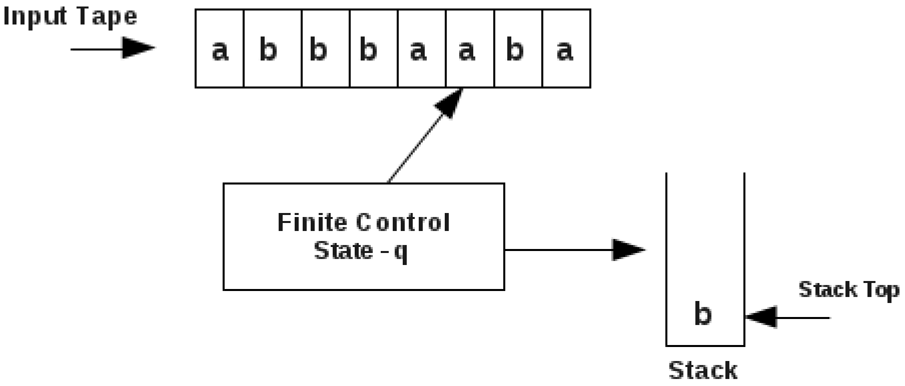
5.

The PDA is in state q, finite control points to symbol a in the input tape, stack top contains symbol b. Consider the transition,

5. $(q, a, b) \longrightarrow (q, \varepsilon)$

This means PDA is in state q, reads a from the input tape, pops b from stack top, moves to state q and pushes ε onto the stack.

PDA now is,



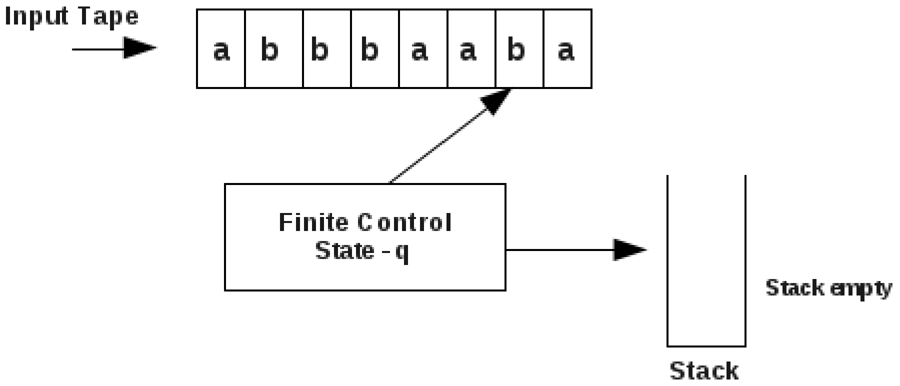
6.

The PDA is in state q, finite control points to symbol a in the input tape, stack top contains symbol b. Consider the transition,

5. $(q, a, b) \longrightarrow (q, \varepsilon)$

This means PDA is in state q, reads a from the input tape, pops b from stack top, moves to state q and pushes nothing onto the stack.

PDA now is,



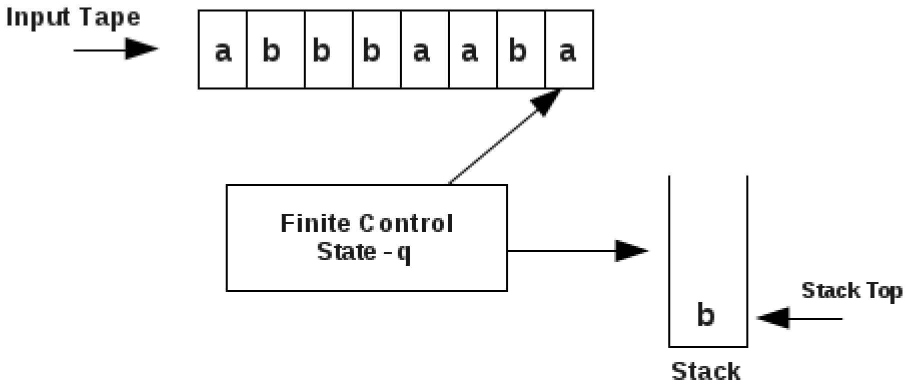
7.

The PDA is in state q, finite control points to symbol b in the input tape, stack top contains symbol ε. Consider the transition,

7. $(q, b, \varepsilon) \longrightarrow (q, b)$

This means PDA is in state q, reads b from the input tape, pops ε from stack top, moves to state q and pushes b onto the stack.

PDA now is,



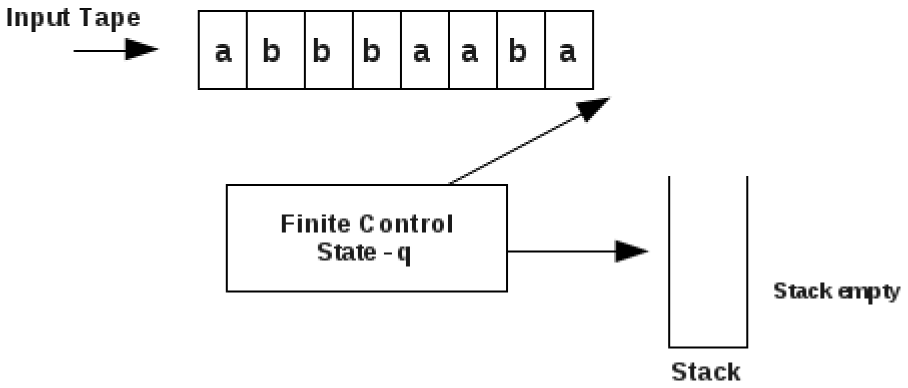
8.

The PDA is in state q, finite control points to symbol a in the input tape, stack top contains symbol b. Consider the transition,

5. $(q, a, b) \longrightarrow (q, \varepsilon)$

This means PDA is in state q, reads a from the input tape, pops b from stack top, moves to state q and pushes ε onto the stack.

PDA now is,



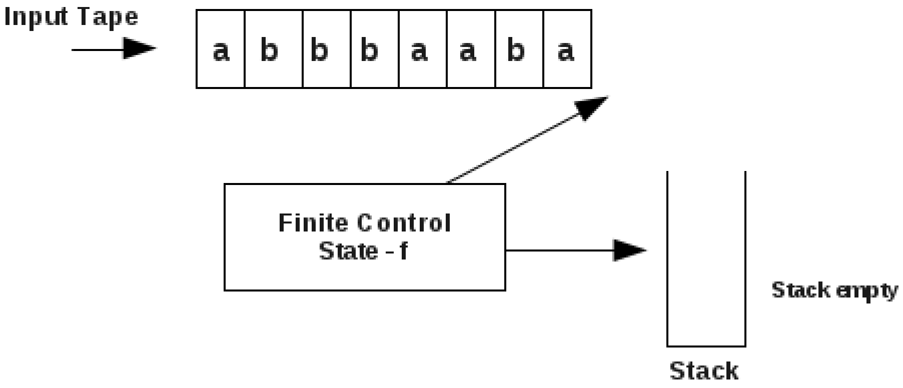
8.

The PDA is in state q, finite control points to symbol ε in the input tape, stack top contains symbol ε. Consider the transition,

8. $(q, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

This means PDA is in state q, reads ε from the input tape, pops ε from stack top, moves to state f and pushes nothing onto the stack.

PDA now is,



Now PDA is in final state, f, and stack is empty. There are no more symbols in the input tape.

So the string $abbbaaba$ is accepted by the above pushdown automation.

Example 2:

1.

Consider a PDA,

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

where

$Q = \{s, f\}$

$\sum = \{a, b, c\}$

$\Gamma = \{a, b\}$

$q_0 = \{s\}$

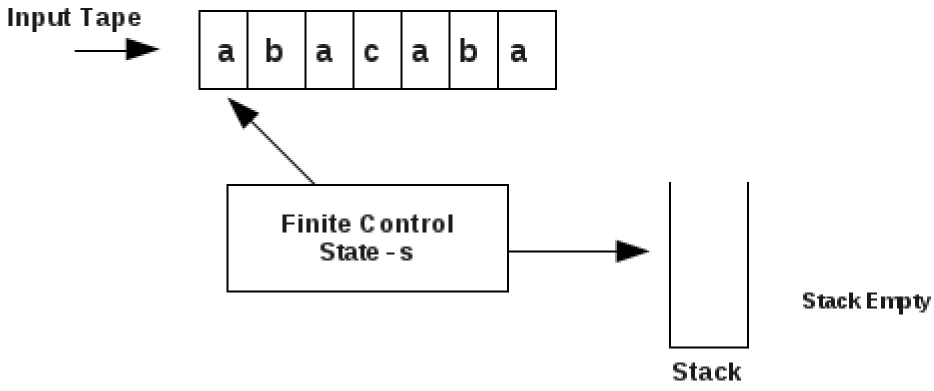
$F = \{f\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (s, a)$
2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$
4. $(f, a, a) \longrightarrow (f, \varepsilon)$
5. $(f, b, b) \longrightarrow (f, \varepsilon)$

Check whether the string $abacaba$ is accpeted by the above pushdown automation.

From the above, stack initially is empty, the start state of the PDA is s as shown below:



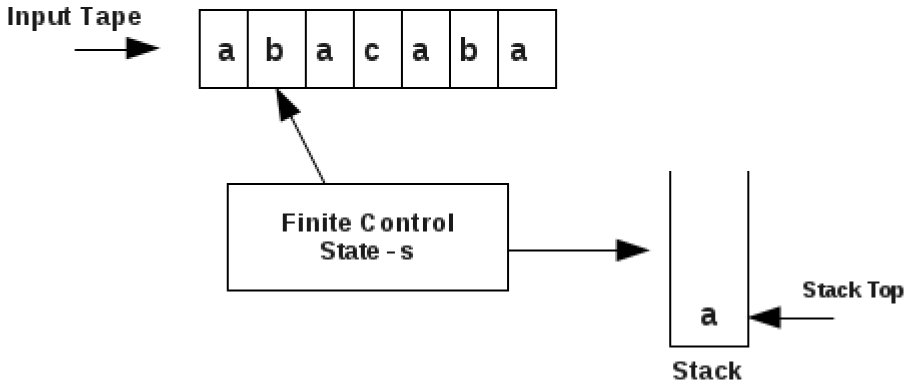
1.

The PDA is in state s, finite control points to symbol a in the input tape, stack top contains symbol ε. Consider the transition,

1. $(s, a, \varepsilon) \longrightarrow (s, a)$

This means PDA is in state s, reads a from the input tape, pops nothing from stack top, moves to state s and pushes a onto the stack.

PDA now is,



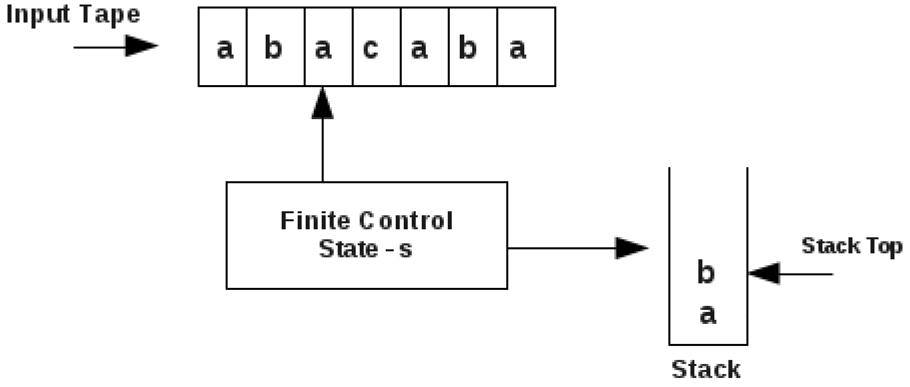
2.

The PDA is in state s, finite control points to symbol b in the input tape, stack top contains symbol a. Consider the transition,

2. $(s, b, \varepsilon) \longrightarrow (s, b)$

This means PDA is in state s, reads b from the input tape, pops nothing from stack top, moves to state s and pushes b onto the stack.

PDA now is,



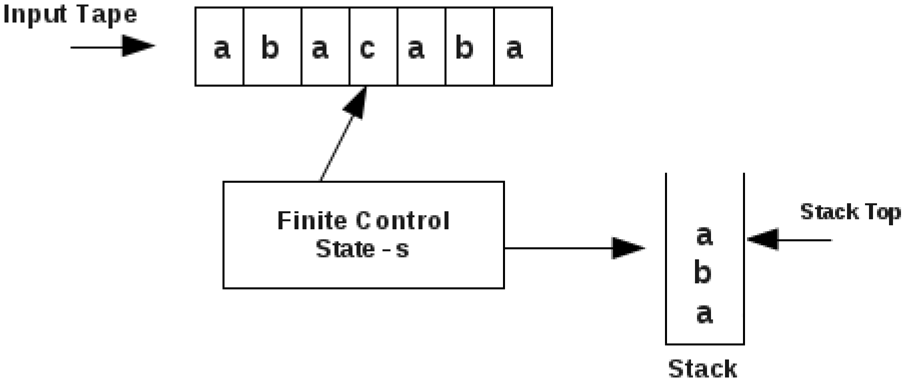
3.

The PDA is in state s, finite control points to symbol a in the input tape, stack top contains symbol b. Consider the transition,

1. $(s, a, \varepsilon) \longrightarrow (s, a)$

This means PDA is in state s, reads a from the input tape, pops nothing from stack top, moves to state s and pushes a onto the stack.

PDA now is,



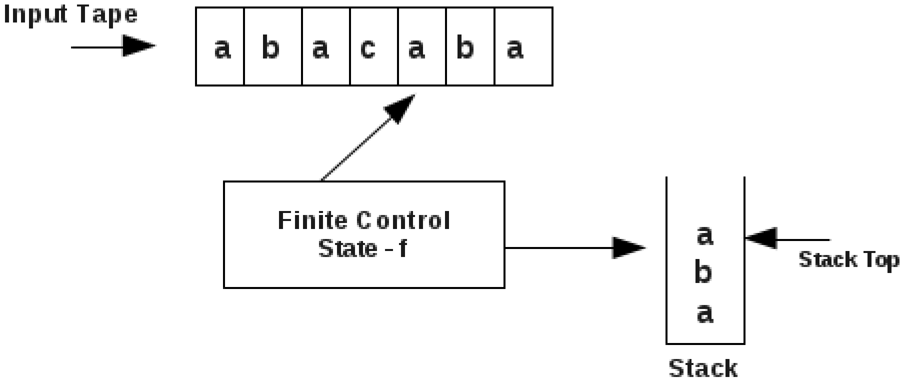
4.

The PDA is in state s, finite control points to symbol c in the input tape, stack top contains symbol a. Consider the transition,

3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$

This means PDA is in state s, reads c from the input tape, pops nothing from stack top, moves to state f and pushes nothing onto the stack.

PDA now is,



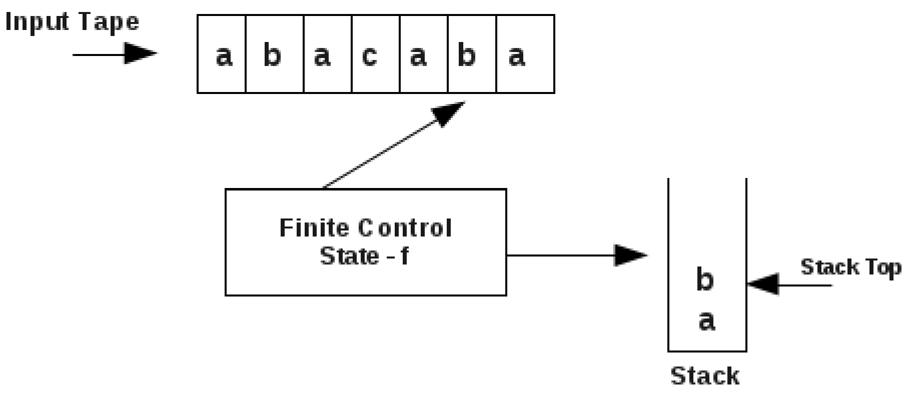
5.

The PDA is in state f, finite control points to symbol a in the input tape, stack top contains symbol a. Consider the transition,

4. $(f, a, a) \longrightarrow (f, \varepsilon)$

This means PDA is in state f, reads a from the input tape, pops a from stack top, moves to state f and pushes nothing onto the stack.

PDA now is,



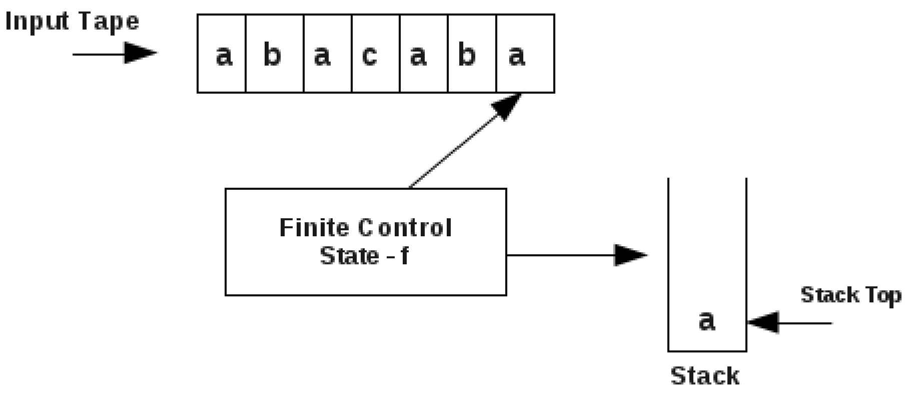
6.

The PDA is in state f, finite control points to symbol b in the input tape, stack top contains symbol b. Consider the transition,

5. $(f, b, b) \longrightarrow (f, \varepsilon)$

This means PDA is in state f, reads b from the input tape, pops b from stack top, moves to state f and pushes nothing onto the stack.

PDA now is,



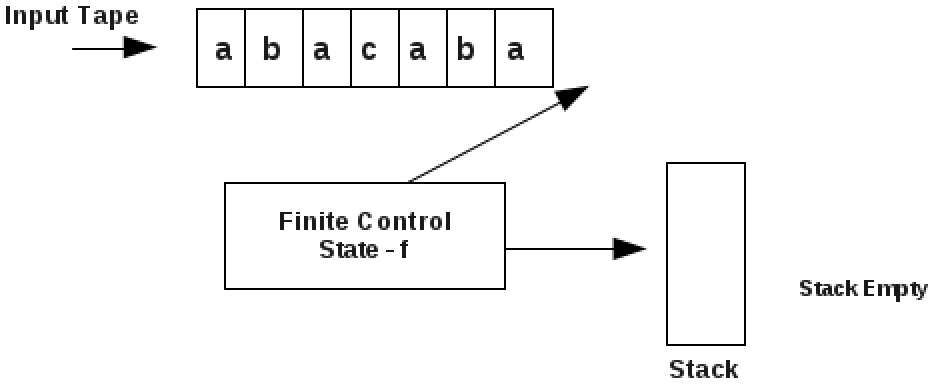
7.

The PDA is in state f, finite control points to symbol a in the input tape, stack top contains symbol a. Consider the transition,

4. $(f, a, a) \longrightarrow (f, \varepsilon)$

This means PDA is in state f, reads a from the input tape, pops a from stack top, moves to state f and pushes nothing onto the stack.

PDA now is,



Now there are no more symbols in the input string, stack is empty. PDA is in final state, f.

So the string $abacaba$ is accepted by the above pushdown automation.

Exercises:

1.

Consider a PDA,

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

where

$Q = \{q_0, q_1, q_2, q_3\}$

$\sum = \{0, 1\}$

$\Gamma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given as follpws:

1. $(q_0, 0, \varepsilon) \longrightarrow (q_1, 0)$
2. $(q_1, 0, 0) \longrightarrow (q_1, 00)$
3. $(q_1, 1, 0) \longrightarrow (q_2, \varepsilon)$
4. $(q_2, 1, 0) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, \varepsilon, \varepsilon) \longrightarrow (q_3, \varepsilon)$

Check whether the string 000111 is accpeted by the above pushdown automation.

2.

Consider a PDA,

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

where

$Q = \{q_0, q_1, q_2, q_3\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given as follpws:

1. $(q_0, a, \varepsilon) \longrightarrow (q_1, a)$
2. $(q_1, a, a) \longrightarrow (q_1, aa)$
3. $(q_1, b, a) \longrightarrow (q_2, a)$
4. $(q_2, a, a) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, \varepsilon, \varepsilon) \longrightarrow (q_3, \varepsilon)$

Check whether the string $aabaa$ is accpeted by the above pushdown automation.

3.

Consider a PDA,

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

where

$Q = \{q_0, q_1, q_2, q_3\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given as follpws:

1. $(q_0, a, \varepsilon) \longrightarrow (q_1, a)$
2. $(q_1, a, a) \longrightarrow (q_1, aa)$
3. $(q_1, b, a) \longrightarrow (q_2, a)$
4. $(q_2, b, a) \longrightarrow (q_3, \varepsilon)$
5. $(q_3, b, a) \longrightarrow (q_2, a)$
5. $(q_3, \varepsilon, \varepsilon) \longrightarrow (q_4, \varepsilon)$

Check whether the string $aabbbb$ is accpeted by the above pushdown automation.

2.

Consider a PDA,

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

where

$Q = \{s, q, f\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b\}$

$q_0 = \{s\}$

$F = \{f\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (s, a)$

2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, \varepsilon, \varepsilon) \longrightarrow (q, \varepsilon)$
4. $(q, a, a) \longrightarrow (q, \varepsilon)$
4. $(q, b, b) \longrightarrow (q, \varepsilon)$
5. $(q, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

Check whether the string $aabbaa$ is accepted by the above pushdown automation.

8 Deterministic Pushdown Automata (DPDA)

A PDA is said to be deterministic, if

1. $\delta(q, a, b)$ contains at most one element, and
2. if $\delta(q, \varepsilon, b)$ is not empty, then

   $\delta(q, c, b)$ must be empty for every input symbol, c.

For example, consider the following PDA,

Example 1:

Consider the PDA,

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

where

$Q = \{s, f\}$

$\sum = \{a, b, c\}$

$\Gamma = \{a, b\}$

$q_0 = \{s\}$

$F = \{f\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (s, a)$
2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$
4. $(f, a, a) \longrightarrow (f, \varepsilon)$
5. $(f, b, b) \longrightarrow (f, \varepsilon)$

Above is a deterministic pushdown automata (DPDA) because it satisfies both conditions of DPDA.

Example 2:

Consider the PDA,

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

where

$Q = \{s, f\}$

$\sum = \{a, b, c\}$

$\Gamma = \{a, b\}$

$q_0 = \{s\}$

$F = \{f\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (s, a)$
2. $(s, b, \varepsilon) \longrightarrow (s, b)$
3. $(s, c, \varepsilon) \longrightarrow (f, \varepsilon)$
4. $(f, a, a) \longrightarrow (f, \varepsilon)$
5. $(f, b, b) \longrightarrow (f, \varepsilon)$

Above is a deterministic pushdown automata (DPDA) because it satisfies both conditions of DPDA.

9 Non-Deterministic Pushdown Automata (NPDA)

A PDA is said to be non- deterministic, if

1. $\delta(q, a, b)$ may contain multiple elements, or
2. if $\delta(q, \varepsilon, b)$ is not empty, then

   $\delta(q, c, b)$ is not empty for some input symbol, c.

Example 1:

Consider the following PDA,

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

where

$Q = \{q_0, q_1, q_2, q_3\}$

$\sum = \{a, b\}$

$\Gamma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given as follpws:

1. $(q_0, a, 0) \longrightarrow (q_1, 10), \ (q_3, \varepsilon)$
2. $(q_0, \varepsilon, 0) \longrightarrow (q_3, \varepsilon)$
3. $(q_1, a, 1) \longrightarrow (q_1, 11)$
4. $(q_1, b, 1) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, b, 1) \longrightarrow (q_2, \varepsilon)$
5. $(q_2, \varepsilon, 0) \longrightarrow (q_3, \varepsilon)$

Above is a non deterministic pushdown automata (NPDA).

Consider the transition 1, ie.

1. $(q_0, a, 0) \longrightarrow (q_1, 10),\ (q_3, \varepsilon)$

Here $(q_0, a, 0)$ can go to q_1 or q_3. That this transition is not deterministic.

Example 2:

Consider the following PDA,

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

where

$Q = \{q_0, q_f\}$

$\sum = \{a, b\}$

$\Gamma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_f\}$

δ is given as follpws:

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$
4. $(q_0, a, 0) \longrightarrow (q_0, 00)$
5. $(q_0, b, 0) \longrightarrow (q_0, \varepsilon)$
6. $(q_0, a, 1) \longrightarrow (q_0, \varepsilon)$
7. $(q_0, b, 1) \longrightarrow (q_0, 11)$

Above is a non deterministic pushdown automata (NPDA).

Consider the transitions, 1, 2 and 3.

1. $(q_0, \varepsilon, \varepsilon) \longrightarrow (q_f, \varepsilon)$
2. $(q_0, a, \varepsilon) \longrightarrow (q_0, 0)$
3. $(q_0, b, \varepsilon) \longrightarrow (q_0, 1)$

Here $(q_0, \varepsilon, \varepsilon)$ is not empty, also,

(q_0, a, ε) and (q_0, b, ε) are not empty.

Example 3:

Consider the following PDA,

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

where

$Q = \{s, p, q\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b\}$

$q_0 = \{s\}$

$F = \{q\}$

δ is given as follpws:

1. $(s, a, \varepsilon) \longrightarrow (p, a)$
2. $(p, a, a) \longrightarrow (p, aa)$
3. $(p, a, \varepsilon) \longrightarrow (p, a)$
4. $(p, b, a) \longrightarrow (p, \varepsilon),\ (q, \varepsilon)$

Above is a non deterministic pushdown automata (NPDA).

This is due to the transition,

4. $(p, b, a) \longrightarrow (p, \varepsilon),\ (q, \varepsilon)$

Example 4:

Consider the following PDA,

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

where

$Q = \{q_0, q_1, q_2, q_3, f\}$

$\sum = \{a, b\}$

$\Gamma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{f\}$

δ is given as follpws:

1. $(q_0, a, \varepsilon) \longrightarrow (q_1, a)$
2. $(q_1, a, a) \longrightarrow (q_1, a)$
3. $(q_1, b, a) \longrightarrow (q_2, \varepsilon)$
4. $(q_2, b, a) \longrightarrow (q_3, \varepsilon)$
5. $(q_3, b, a) \longrightarrow (q_2, \varepsilon)$
6. $(q_2, \varepsilon, a) \longrightarrow (q_2, \varepsilon)$
7. $(q_2, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$
8. $(q_1, \varepsilon, a) \longrightarrow (q_1, \varepsilon)$
9. $(q_1, \varepsilon, \varepsilon) \longrightarrow (f, \varepsilon)$

Above is a non deterministic pushdown automata (NPDA).

This is due to the transitions, 6 and 4, ie,

6. $(q_2, \varepsilon, a) \longrightarrow (q_2, \varepsilon)$

4. $(q_2, b, a) \longrightarrow (q_3, \varepsilon)$

Also due to the transitions, 8 and 2 and 3, ie,

8. $(q_1, \varepsilon, a) \longrightarrow (q_1, \varepsilon)$

2. $(q_1, a, a) \longrightarrow (q_1, a)$

3. $(q_1, b, a) \longrightarrow (q_2, \varepsilon)$

10 Design of Pushdown Automata

For every CFG, there exists a pushdown automation that accepts it.

To design a pushdown automation corresponding to a CFG, following are the steps:

Step 1:

Let the start symbol of the CFG is S. Then a transition of PDA is,

$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$

Step 2:

For a production of the form, $P \longrightarrow AaB$, a transition of PDA is,

$\delta(q, \varepsilon, P) \longrightarrow (q, AaB)$

For a production of the form, $P \longrightarrow a$, a transition of PDA is,

$\delta(q, \varepsilon, P) \longrightarrow (q, a)$

For a production of the form, $P \longrightarrow \varepsilon$, a transition of PDA is,

$\delta(q, \varepsilon, P) \longrightarrow (q, \varepsilon)$

Step 3:

For every terminal symbol, a in CFG, a transition of PDA is,

$\delta(q, a, a) \longrightarrow (q, \varepsilon)$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum =$ set of terminal symbols in the CFG

$\Gamma =$set of terminals and non-terminals in the CFG

$q_0 = p$

$F = q$

δ is according to the above rules.

Example 1:

Consider the following CFG,

$S \longrightarrow aA$

$A \longrightarrow aABC|bB|a$

$B \longrightarrow b$

$C \longrightarrow c$

where S is the start symbol.

Design a pushdown automata corresponding to the above grammar.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 3:

Consider the production, $S \longrightarrow aA$.

A transition is

$$\delta(q, \varepsilon, S) \longrightarrow (q, aA)$$

Consider the production, $A \longrightarrow aABC|bB|a$.

Transitions are

$$\delta(q, \varepsilon, A) \longrightarrow (q, aABC),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, bB),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, a).$$

Consider the production, $B \longrightarrow b$.

Corresponding transition is

$$\delta(q, \varepsilon, B) \longrightarrow (q, b)$$

Consider the production, $C \longrightarrow c$

Corresponding transition is

$$\delta(q, \varepsilon, C) \longrightarrow (q, c)$$

Step 3:

The terminal symbols in the CFG are, a, b, c.

Then the transitions are,

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

$$\delta(q, c, c) \longrightarrow (q, \varepsilon)$$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b, c\}$

$\Gamma = \{S, A, B, C, a, b, c\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aA)$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, aABC),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, bB),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, a).$$

$$\delta(q, \varepsilon, B) \longrightarrow (q, b)$$

$$\delta(q, \varepsilon, C) \longrightarrow (q, c)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

$$\delta(q, c, c) \longrightarrow (q, \varepsilon)$$

Example 2:

Consider the CFG,

$S \longrightarrow AB|BC$

$A \longrightarrow aB|bA|a$

$B \longrightarrow bB|cC|b$

$C \longrightarrow c$

where S is the start symbol.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 3:

Consider the production, $S \longrightarrow AB|BC$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, AB)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, BC$$

Consider the production, $A \longrightarrow aB|bA|a$

Transitions are

$$\delta(q, \varepsilon, A) \longrightarrow (q, aB),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, bA),$$

$$\delta(q, \varepsilon, A) \longrightarrow (q, a).$$

Consider the production, $B \longrightarrow bB|cC|b$

Transitions are

$$\delta(q, \varepsilon, B) \longrightarrow (q, bB)$$

$$\delta(q, \varepsilon, B) \longrightarrow (q, cC),$$

$$\delta(q, \varepsilon, B) \longrightarrow (q, b)$$

Consider the production, $C \longrightarrow c$

Transitions are

$$\delta(q, \varepsilon, C) \longrightarrow (q, c)$$

Step 3:

The terminal symbols in the CFG are, a, b, c.

Then the transitions are,

$$\delta(q,a,a) \longrightarrow (q,\varepsilon)$$

$$\delta(q,b,b) \longrightarrow (q,\varepsilon)$$

$$\delta(q,c,c) \longrightarrow (q,\varepsilon)$$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b, c\}$

$\Gamma = \{S, A, B, C, a, b, c\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p,\varepsilon,\varepsilon) \longrightarrow (q,S)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,AB)$$

$$\delta(q,\varepsilon,S) \longrightarrow (q,BC)$$

$$\delta(q,\varepsilon,A) \longrightarrow (q,aB)$$

$$\delta(q,\varepsilon,A) \longrightarrow (q,bA)$$

$$\delta(q,\varepsilon,A) \longrightarrow (q,a)$$

$$\delta(q,\varepsilon,B) \longrightarrow (q,bB)$$

$$\delta(q,\varepsilon,B) \longrightarrow (q,cC)$$

$$\delta(q,\varepsilon,B) \longrightarrow (q,b)$$

$$\delta(q,\varepsilon,C) \longrightarrow (q,c)$$

$$\delta(q,a,a) \longrightarrow (q,\varepsilon)$$

$$\delta(q,b,b) \longrightarrow (q,\varepsilon)$$

$$\delta(q,c,c) \longrightarrow (q,\varepsilon)$$

Example 3:

Design a pushdown automata that accepts the language, $L = \{wcw^R | w \in (a,b)^*\}$

We learned earlier that the CFG corresponding to this language is,

$S \longrightarrow aSa$

$S \longrightarrow bSb$

$S \longrightarrow c$

where S is the start symbol.

Next we need to find the PDA corresponding to the above CFG.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 3:

Consider the production, $S \longrightarrow aSa$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, aSa)$$

Consider the production, $S \longrightarrow bSb$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, bSb),$$

Consider the production, $S \longrightarrow c$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, c)$$

Step 3:

The terminal symbols in the CFG are, a, b, c.

Then the transitions are,

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

$$\delta(q, c, c) \longrightarrow (q, \varepsilon)$$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b, c\}$

$\Gamma = \{S, a, b, c\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aSa)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bSb)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, c)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

$$\delta(q, c, c) \longrightarrow (q, \varepsilon)$$

Example 4:

Design a pushdown automata that accepts the language, $L = \{w|w$ contains equal number of a's and b's $\}$ from the input alphabet {a,b}.

First, we need to find the CFG corresponding to this language. We learned in a previous section, the CFG corresponding to this is,

$S \longrightarrow aSbS|bSaS|\varepsilon$

where S is the start symbol.

Next we need to find the PDA corresponding to the above CFG.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 3:

Consider the production, $S \longrightarrow aSbS|bSaS|\varepsilon$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, aSbS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bSaS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, \varepsilon)$$

Step 3:

The terminal symbols in the CFG are, a, b.

Then the transitions are,

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b\}$

$\Gamma = \{S, a, b\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aSbS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bSaS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, \varepsilon)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

Example 5:

Design a pushdown automata that accepts the language corresponding to the regular expression, $(a|b)^*$.

First, we need to find the CFG corresponding to this language. We learned in a previous section, the CFG corresponding to this is,

$S \longrightarrow aS|bS|a|b|\varepsilon$

where S is the start symbol.

Next we need to find the PDA corresponding to the above CFG.

Step 1:

Here start symbol of CFG is S. A transition is,

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

Step 3:

Consider the production, $S \longrightarrow aS|bS|a|b|\varepsilon$

Transitions are

$$\delta(q, \varepsilon, S) \longrightarrow (q, aS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, a)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, b)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, \varepsilon)$$

Step 3:

The terminal symbols in the CFG are, a, b.

Then the transitions are,

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

Thus the PDA is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b\}$

$\Gamma = \{S, a, b\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, bS)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, a)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, b)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, \varepsilon)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

Exercises:

1. Design a pushdown automation to accept the language, $L = \{ww^R | w \in \{a, b\}^*\}$.
2. Design a pushdown automation to accept the language, $L = \{a^n b^n | n > 0\}$.
3. Design a pushdown automation to accept the language, $L = \{a^n b^{2n} | n > 0\}$.
4. Design a pushdown automation for the regular expression, $r = 0^*1^*$.
5. Design a pushdown automation to accept the language, $L = \{a^n b^{n+1} | n = 1, 2, 3, ...\}$.
6. Design a pushdown automation to accept the language, $L = \{a^n b^m | n > m \geq 0\}$.
7. Construct PDA equivalent to the grammar $S \longrightarrow aAA,\ A \longrightarrow aS/bS/a$.
8. Construct PDA for the language $L = \{x \in (a, b)^* / n_a(x) > n_b(n)\}$.
9. Construct a PDA that accepts the language $L = \{a^n b a^m / n,\ m \geq 1\}$ by empty stack.

11 Applications of Pushdown Automata

An application of PDA is in parsing.

Parsing is the process of checking whether the syntax of a program is correct or not. For example, a PDA can be constructed to check the syntax of all C programs.

Also, parsing is the process of checking whether a sentence written in a natural language is correct or not.

Parsing is of two types,

Top down parsing, and

Bottom up parsing.

Top Down Parsing

An example for top down parsing is given below:

Example 1:

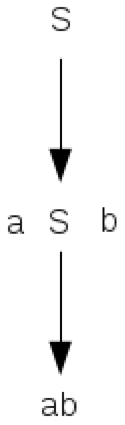
Consider the following CFG,

$S \longrightarrow aSb \,|\, ab$

where S is the start symbol.

Check whether the syntax of string $aabb$ is according to the above CFG.

Top down parsing is shown below:



Thus top down parsing is done beginning from the start symbol, by following a set of productions, reach the given string. If the string cannot be reached, its syntax is not correct.

Above diagram is called a derivation tree (parse tree).

Here, by performing top down parsing , we got a^2b^2. So a^2b^2 belongs to the above CFL.

PDA for Top down Parsing

A top down parser can be implemented by using a Pushdown automation. Using the pushdown automation, we can parse a given string.

Example 1:

For example, the PDA corresponding to the CFG, $S \longrightarrow aSb\,|\,ab$

is,

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

where

$Q = \{p, q\}$

$\sum = \{a, b\}$

$\Gamma = \{S, a, b\}$

$q_0 = p$

$F = q$

δ is given as follows:

$$\delta(p, \varepsilon, \varepsilon) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, aSb)$$

$$\delta(q, \varepsilon, S) \longrightarrow (q, ab)$$

$$\delta(q, a, a) \longrightarrow (q, \varepsilon)$$

$$\delta(q, b, b) \longrightarrow (q, \varepsilon)$$

This PDA acts as a top down parser for the CFL corresponding to above CFG.

Bottom Up Parsing

This approach can also be used to check whether a string belongs to a context free language. Here, the string w is taken and an inverted tree is made. This is done by performing a number of reductions starting from the given string using the productions of the grammar.

Example 1:

Consider the grammar,

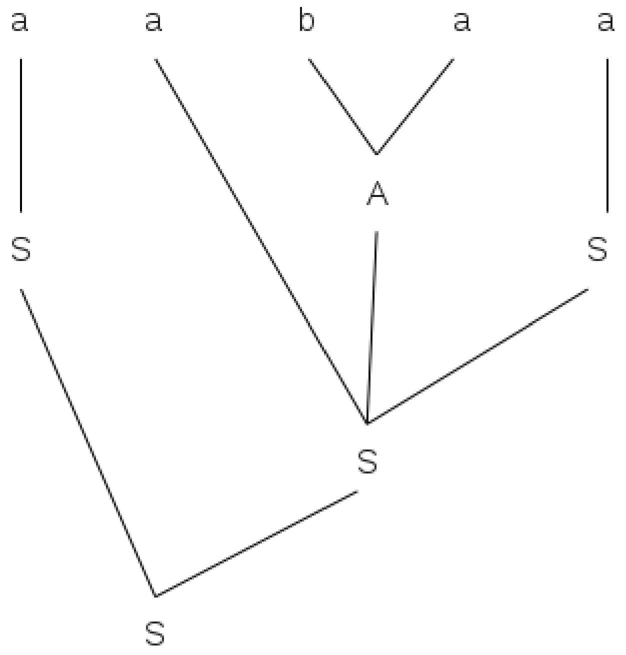
$S \longrightarrow aAS|a|SS$

$A \longrightarrow SbA|ba$

where S is the start symbol.

Check whether the string aabaa belongs to the CFL associated with the above grammar.

Bottom up parsing is shown below:



Above is a bottom up parse tree.

Thus by performing a number of reductions, we finally reached the start symbol. So the string aabaa belongs to the above CFL.

This parsing process can be done using a PDA.

PDA for Bottom down Parsing

A top down parser can be implemented by using a Pushdown automation. Using this pushdown automation, we can parse a given string.

Example 1:

Consider the grammar,

$$S \longrightarrow aAS|a|SS$$

$$A \longrightarrow SbA|ba$$

where S is the start symbol.

A pushdown automation for performing bottom up parsing is shown below:

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

where

$$Q = \{p, q\}$$

$$\sum = \{a, b\}$$

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

$$q_0 = p$$

$$F = q$$

δ is given as follows:

$$\delta(q, \varepsilon, S) \longrightarrow (p, \varepsilon)$$

$$\delta(q, \varepsilon, SAa) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, a) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, SS) \longrightarrow (q, S)$$

$$\delta(q, \varepsilon, AbS) \longrightarrow (q, A)$$

$$\delta(q, \varepsilon, ab) \longrightarrow (q, A)$$

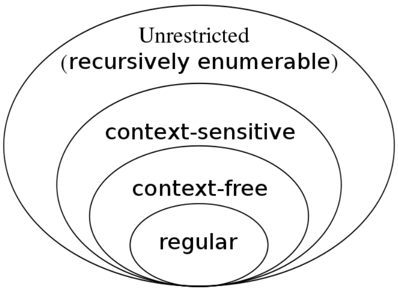
$$\delta(q, a, \varepsilon) \longrightarrow (q, a)$$

$$\delta(q, b, \varepsilon) \longrightarrow (q, b)$$

This PDA acts as a bottom up parser for the CFL corresponding to above CFG.

Part V. Pumping Lemma for Context Free Languages (CFLs)

Consider the following figure:



From the diagram, we can say that not all languages are context free languages. All context free languages are context sensitive and unrestricted. But all context sensitive and unrestricted languages are not context free from the above diagram.

We have a mechanism to show that some languages are not context free. The pumping lemma for CFLs allow us to show that some languages are not context free.

12 Pumping lemma for CFLs

Let G be a CFG. Then there exists a constant, n such that if W is in $L(G)$ and $|W| \geq n$, then we may write $W = uvxyz$ such that,

1. $|vy| \geq 1$ that is either v or y is non-empty,
2. $|vxy| \leq n$ then for all $i \geq 0$,

   uv^ixy^iz is in $L(G)$.

This is known as pumping lemma for context free languages.

Pumping lemma for CFLs can be used to prove that a language, L is not context free.

We assume L is context free. Then we apply pumping lemma to get a contradiction.

Following are the steps:

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n$. Write $W = uvxyz$ using the pumping lemma.

Step 3:

Find a suitable k so that $uv^kxy^kz \notin L$. This is a contradiction, and so L is not context free.

Examples:

Example 1:

Show that $L = \{a^nb^nc^n | n \geq 1\}$ is not context free.

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n.$ Write $W = uvxyz$ using the pumping lemma.

Let $W = a^n b^n c^n$. Then $|W| = 3n > n.$

Write $W = uvxyz,$ where $|vy| \geq 1,$ that is, at least one of v or x is not $\varepsilon.$

Step 3:

Find a suitable k so that $uv^k xy^k z \notin L.$ This is a contradiction, and so L is not context free.

$uvxyz = a^n b^n c^n$. As $1 \leq |vy| \leq n,$ v or x cannot contain all the three symbols a, b, c.

i) So v or y is of the form $a^i b^j$ (or $b^j c^j$) for some i,j such that $i + j \leq n.$ or

ii) v or y is a string formed by the repetition of only one symbol among a, b, c.

When v or y is of the form $a^i b^j$, $v^2 = a^i b^j a^i b^j$ (or $y^2 = a^i b^j a^i b^j$). As v^2 is a substring of the form $uv^2 xy^2 z,$ we cannot have $uv^2 xy^2 z$ of the form $a^m b^m c^m$. So $uv^2 xy^2 z \notin L.$

When both v and y are formed by the repetition of a single symbol (example: $u = a^i$ and $v = b^j$ for some i and j, $i \leq n, j \leq n$), the string uxz will contain the remaining symbol, say a_1. Also a_1^n will be a substring of uxz as a_1 does not occur in v or y. The number of occurrences of one of the other two symbols in uxz is less than n (recall $uvxyz = a^n b^n c^n$), and n is the number of occurrences of a_1. So $uv^0 xy^0 z = uxz \ \notin L.$

Thus for any choice of v or y , we get a contradiction. Therefore, L is not context free.

Example 2:

Show that $L = \{a^p | p$ is a prime number$\}$ is not a context free language.

This means, if $w \in L,$ number of characters in w is a prime number.

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n.$ Write $W = uvxyz$ using the pumping lemma.

Let p be a prime number greater than n. Then $w = a^p \in L.$

We write $W = uvxyz$

Step 3:

Find a suitable k so that $uv^k xy^k z \notin L.$ This is a contradiction, and so L is not context free.

By pumping lemma, $uv^0 xy^0 z = uxz \in L.$

So $|uxz|$ is a prime number, say q.

Let $|vy| = r.$

Then $|uv^q xy^q z| = q + qr.$

Since, $q + qr$ is not prime, $uv^q xy^q z \notin L.$ This is a contradiction. Therefore, L is not context free.

Example 3:

Show that the language $L = \{a^n b^n c^n | n \geq 0\}$ is not context free.

Step 1:

Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2:

Choose $W \in L$ so that $|W| \geq n$. Write $W = uvxyz$ using the pumping lemma.

Let $W = a^n b^n c^n$.

We write $W = uvxyz$ where $|vy| \geq 1$ and $|vxy| \leq n$, then

pumping lemma says that $uv^ixy^iz \in L$ for all $i \geq 0$.

Observations:

a. But this is not possible; because if v or y contains two symbols from {a,b,c} then uv^2xy^2z contains a 'b' before an 'a' or a 'c' before 'b', then uv^2xy^2z will not be in L.

b. In other case, if v and y each contains only a's, b's or only c's, then uv^ixy^iz cannot contain equal number of a's, b's and c's. So uv^ixy^iz will not be in L.

c. If both v and y contains all three symbols a, b and c, then uv^2xy^2z will not be in L by the same logic as in observation (a).

So it is a contradiction to our statement. So L is not context free.

Step 3:

Find a suitable k so that $uv^kxy^kz \notin L$. This is a contradiction, and so L is not context free.

By pumping lemma, $uv^0xy^0z = uxz \in L$.

So $|uxz|$ is a prime number, say q.

Let $|vy| = r$.

Then $|uv^qxy^qz| = q + qr$.

Since, $q + qr$ is not prime, $uv^qxy^qz \notin L$. This is a contradiction. Therefore, L is not context free.

Questions (from Old syllabus of S7CS TOC)

MGU/Nov2011

1. Define a pushdown automata (4marks).

2. Design a context free grammar that accepts the language $L = \{0^n 1^n / n \geq 1\}$ (4marks).

3a. Design a pushdown automata that accepts the language, $L = \{w/w$ contains equal number of 0's and 1's } from the input alphabet {0,1}.

OR

b. Differentiate between deterministic and non-deterministic PDA with examples (12marks).

MGU/April2011

1. Define a context free language (4marks).

2a. Construct a context free grammar to generate $\{W \subset W^R / W \in \{a, b\}^*\}$.

OR

b. If L is given by $L = \{a^n b^n c^n\}/n \geq 1$, check whether L is CFL or not? Prove (12marks).

MGU/Nov2010

1. Define PDA and what are the languages accepted by PDA (4marks).

2a. ii. Show that $\{0^x/x$ is prime} is not context free (12marks).

3a. Construct a PDA to accept $L = \{w \in \{a, b\}^*/w = wR\}$.

OR

b. Prove that context free language is closed under union (12marks).

MGU/May2010

1. Define context free grammar. Construct context free grammar that generate the language $\{wcw^r?w \in \{a, b\}^*\}$(4marks).

2. Differentiate deterministic PDA from non-deterministic PDA (4marks).

3a. Construct context free grammar that generate the language $\{wcw^R/w \in \{a, b\}^*\}$.

OR

b. Construct a context free grammar for the given language $L = \{anbn1/n >= 1\} \cup \{amb\,2m/m > -1\}$ and hence a PDA accepting L by empty stack (12marks).

MGU/Nov2009

1. Give pumping lemma to prove that given language L is not context free (4marks).

2. Give an example of a language accepted by a PDA but not by DPDA (4marks).

3a. Prove that $\{op/p$ is prime} is not context free.

OR

b. Convert the grammar $S \longrightarrow ABb/a,\ A \longrightarrow aaA/B\ B \longrightarrow bAb$ into Greibach normal form (12marks).

MGU/Nov2008

1. Give the formal definition of context free grammar (4marks).

2a i. Construct automation that accepts the language $S \longrightarrow aA,\ A \longrightarrow abB/b,\ B \longrightarrow aA/a$ (6marks).

OR

b. Prove that if L is $L(M_2)$ for some PDA M_2, then L is $N(M_1)$, for some PDA M_1 (12marks).

MGU/May2008

1. State pumping lemma for context free languages (4marks).

2. What is acceptance concept of push down automata (4marks).

3. Explain two normal forms of context free grammar (4marks).

4a. ii. Show that $(0^{n/n}$ is prime} is not context free (6marks).

5a. i. Construct PDA for the language $L = \{x \in (a,b)^*/n_a(x) > n_b(n)\}$.

ii. Explain applications of PDA.

OR

b. Construct PDA equivalent to the grammar $S \longrightarrow aAA,\ A \longrightarrow aS/bS/a$ (12marks).

MGU/Dec2007

1. Define context free language (4marks).

2a ii. Convert CFG to NDFA for $S \longrightarrow ABaC,\ A \longrightarrow BC,\ B \longrightarrow b/E,\ C \longrightarrow D,\ D \longrightarrow d$ (6marks).

3a. Construct PDA to accept the language given by $\{w \in (a,b)^*/w$ has the same number of a's as that of b's }.

OR

b. Is $\alpha = \{0^n/n$ is prime } is context free or not. Prove the answer (12marks).

MGU/July2007

1. When do you call a context free grammar ambigous? Give an example (4marks).

2. Give a CFG that generates all palindromes over {a,b} and show that derivation of ababa (4marks).

3a. i. Construct a PDA which accepts the language $\{wcw^R/w \in \{a,b\}^*\}$ and show the processing of abcba.

ii. Briefly explain parsing using PDA with an example.

OR

b. i. Construct a PDA which accepts the language $\{ww^R/w \in \{a,b\}^*\}$ and show the processing of abba.

ii. Design a CFG for the language $L = \{a^n b^m/n$ not equal to m} (12marks).

MGU/Jan2007

1. Find a CFG for the language $L = \{a^n b^{2n} c^m/n, m \geq 0\}$ (4marks).

2. Find a CFG for the language of all strings over [a,b] with exactly one a or exactly one b (4marks).

3a. i. Remove all unit productions, useless productions and ε productions from the grammar-

$S \longrightarrow aA|aBB$

$A \longrightarrow aaA|\varepsilon$

$B \longrightarrow bB|bbC$

$C \longrightarrow B$

What language does the grammar generate?

ii. Convert the following grammar to Chomsky normal form:

$S \longrightarrow ABa$

$A \longrightarrow aaB$

$B \longrightarrow Ac$

OR

b. Prove that for any context free language L there exists a non-deterministic PDA, M such that L=L(M). (12marks)

MGU/July2006

1. What is a context free language? Explain with an example (4marks).

2. Define a PDA (4marks).

3a. Construct a PDA that accepts the language $L = \{a^n ba^m / n,\ m \geq 1\}$ by empty stack.

OR

b. i. Given the CFG $(\{S\}, \{a, b\}, \{S \longrightarrow SaS,\ S \longrightarrow b\},\ S)$, draw the derivation tree for the string bababab.

ii. Give a CFG which generates the language $L = \{a^n b^n c^n / n \geq 1\}$ (12marks).

MGU/Nov2005

1. Give the formal definition of a grammar (4marks).

2. Define a push down automata (4marks).

3. What is meant by an inherently ambigous CFL (4marks)?

4a. Prove that if L is $L(M_2)$ fro some PDA M_2 then L is $N(M_1)$ for some PDA M_1.

OR

b. i. Give a context free grammar which generates the language $L = \{w/w$ contains twice as many 0s and 1s}.

ii. Consider the grammar: $S \longrightarrow SS + /SS * /a$. Explain how the string $aa + a*$ can be generated by the grammar (12marks).

References

.

Pandey, A, K (2006). An Introduction to automata Theory and Formal Languages. Kataria & Sons.

Mishra, K, L, P; Chandrasekaran, N (2009). Theory of Computer Science. PHI.

Nagpal, C, K (2011). Formal Languages and Automata Theory. Oxford University Press.

Murthy, B, N, S (2006). Formal Languages and Automata Theory. Sanguine.

Kavitha,S; Balasubramanian,V (2004). Theory of Computation. Charulatha Pub.

Linz, P (2006). An Introduction to Formal Languages and Automata. Narosa.

Hopcroft, J, E; Motwani, J; Ullman, J, D (2002). Introduction to Automata Theory, Languages and Computation. Pearson Education.

website: http://sites.google.com/site/sjcetcssz