AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture notes on Context Free Grammar(Non-Linear)

Prepared by:
Prof.Sangeeta V I
Assistant Professor

Department of Computer Science & Engineering
PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of contents

Examples Solved:

1. **Context Free Grammar (Non-Linear)**

CFG may have more than one variable/non-terminal on the right-hand side of the production.
LHS can only be a single variable/non-terminal.

Example 1:
Construct a CFG for $L=\{uvwv^R, |u|=|w|=2, |v|>1, w \in \{a,b\}^*\}$

A will take care of **u** and **B** will take care of **v w v^R**.
$|u|=|w|=2$
$|v|>1$, vv^R-palindrome.
S→ A B
A→ aa|bb|ab|bb (|u|= 2)
B→ aBa|bBb (to generate palindrome-vv^R)
B→ aBa|bBb|A (B→ A ,since |w|=|u|=2)

The CFG for $L=\{uvwv^R, |u|=|w|=2, |v|>1, w \in \{a,b\}^*\}$ is :
S→ A B
A→ aa|bb|ab|bb
B→ aBa|bBb|A

Example 2:

Construct a CFG for L={$n_a(w)=n_b(w)$, w ∈ {a,b}*}

Note that $n_a(w)=n_b(w)$ is not the same as a^nb^n.

$a^nb^n \subseteq n_a(w)=n_b(w)$

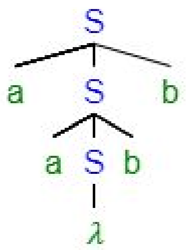
In $n_a(w)=n_b(w)$ there is no specific order.

S→ aSb|bSa|λ

From the above production rule we cannot generate "abba" which also has an equal number of a's and b's.

S⇒aSb⇒abSab⇒abab ,we cannot generate "abba".

Parse tree for deriving "abba"

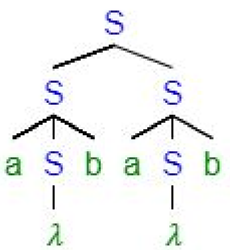


ab ,ab must occur side by side,to do that we introduce the production S→ SS

S→ aSb|bSa|λ|SS

S⇒SS⇒aSbS⇒aSbaSb⇒abab

Parse Tree for deriving "abab:



The CFG for L={$n_a(w)=n_b(w)$, w ∈ {a,b}*} is :

S→ aSb|bSa|λ|SS

Example 3:

Construct a CFG for L={$n_a(w)=n_b(w)+1$,w $\in$ {a,b}*}

The grammar is similar to $n_a(w)=n_b(w)$, here for $n_a(w)=n_b(w)+1$ we terminate with a, instead of terminating with λ.

The CFG for L={$n_a(w)=n_b(w)+1$, w $\in$ {a,b}*} is :

S→AaA

A→ aAb|bAa|AA|λ

Example 4:

Construct a CFG for L={$n_a(w) = 2\times n_b(w)$,w $\in$ {a,b}*}.

$n_a(w) = 2\times n_b(w)$,the number of a's =twice the number of b's.

Minimal strings= **λ,aab,baa,aba** (there is no order)

Since there is no order ,we can put an S between aab,baa,abb

S→ aSaSb|bSaSa|aSbSb|SS|λ **(**S→ SS will take care of any order)

The CFG for L={$n_a(w) = 2\times n_b(w)$,w $\in$ {a,b}*} is:

S→ aSaSb|bSaSa|aSbSb|SS|λ

Example 5:

Construct a CFG for L={$n_a(w) > n_b(w)$,w $\in$ {a,b}*}.

The number of a's is more than the number of b's.

S→ AaA

A→ aAb|bAa|aA|Aa|λ

The CFG for L={$n_a(w) > n_b(w)$,w $\in$ {a,b}*} is:

S→ AaA

A→ aAb|bAa|aA|Aa|λ

Example 6:

Construct a CFG for $L=\{n_a(w) \neq n_b(w), w \in \{a,b\}^*\}$.

$n_a(w) \neq n_b(w)$,number of a's are more or number of b's are more.

A will take care of the number **of a's** more than the number of b's.
B will take care of the number **of b's** bmore than the number of a's**.**

S→ AaA|BbB
A→ aAb|bAa|aA|Aa|λ
B→ aBb|bBa|bB|Bb|λ

The CFG for $L=\{n_a(w) \neq n_b(w), w \in \{a,b\}^*\}$ is:
S→ AaA|BbB
A→ aAb|bAa|aA|Aa|λ
B→ aBb|bBa|bB|Bb|λ

Example 7:

Construct a CFG for $L=\{a^nb^n \cup a^nb^{2n}\}$.

Generate the grammar for a^nb^n and a^nb^{2n}.

So the Grammar $L=\{a^nb^n \cup a^nb^{2n}\}$,must generate string in a^nb^n and the strings in a^nb^{2n} or both . Union is an "or" operator .

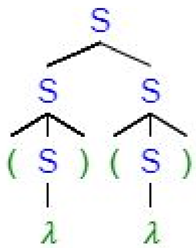
$L=\{a^nb^n \cup a^nb^{2n}\}$
$\mathbf{S} \rightarrow \mathbf{S_1} \mid \mathbf{S_2}$
$\mathbf{S_1} \rightarrow a\mathbf{S_1}b \mid \boldsymbol{\lambda}$
$\mathbf{S_2} \rightarrow a\mathbf{S_2}bb \mid \boldsymbol{\lambda}$

Example 8:

Construct a CFG for Language of proper nesting(parenthesis matching).
Let $\Sigma = \{(,)\}$ and let $L = \{w \in \Sigma^* \mid w$ is a string of balanced parentheses$\}$.

The string must start with an opening parenthesis.
At any point, counting from left to right (i.e., in any prefix), the number of closing parentheses encountered thus far cannot be greater than the number of opening parentheses. The number of opening and closing parentheses must be equal at the end of the string.
Look at the closing parenthesis that matches the first open parenthesis.
((( )( ))( ))
S→ (S) | SS
The empty string(λ) is a string of balanced parentheses.
S→ λ

Parse tree:



CFG for Language of proper nesting:
S→ (S) | SS
S→ λ

Example 9:

Construct a CFG for Language of proper nesting(parenthesis matching).
Let $\Sigma = \{(,\{,[\ ,],\},)\}$ and let $L = \{w \in \Sigma^* \mid w$ is a string of balanced parentheses$\}$.

Example of matched parenthesis:**[{ (({ })) }]**
This is similar to the previous example.

CFG for Language of proper nesting for $\Sigma = \{(,\{,[\ ,),\},]\}$ is:
S→(S)|{S}|[S]|SS|λ

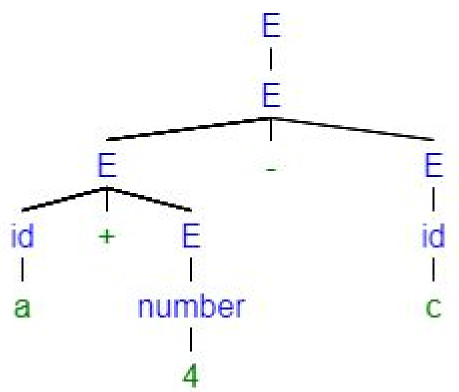
Example 10:

Construct a CFG for Language to generate arithmetic expressions.

Σ ={+,-,*,/,(), number [0-9],id}
id-identifiers/variable names .

Inplace of S we will use E as the start symbol.
E→ E+E | E-E | E*E | E/E | (E) | id| number

Parse tree for a+4-c:



CFG for Language to generate arithmetic expressions is:
E→ E+E | E-E | E*E | E/E | (E) | id| number

Example 11:

Construct a CFG for nested if else.
Σ = {if, condition, then, else, statement, {, } }

- {,} avoids dangling else problems .It associates else with corresponding if.
- An example string in the language is as follows:
  If condition then statement else { if condition then statement else statement }

S→ if condition then S
S→ if condition then S else S
S→ {statement } (braces to avoid the dangling else problem)

CFG for nested if else is:

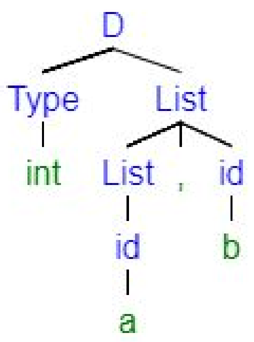
S→ if condition then S
S→ if condition then S else S
S→ {statement }

Example 12:
Construct a CFG to take care of variable declarations in C Language.

Declarations :
Example : int a, int a,b .Here every declaration is in the format type followed by variables also called as id's separated by comma.

D→ Type List   (List -list of variables)
List→ List,id|id   (List→ List,id list ending with id)
Type→ int|float|char

Parse Tree for declaration int a,b



Example 13:
Construct a CFG to generate nested while loops.

While has a condition,while(condition)
S→ while(condition)S|{statement}

Do while loop:
S→ while(condition)S|do S while (condition){statement}