Theory of Computation (CS F351)

Prof.R.Gururaj
CS&IS Dept.

BITS Pilani
Hyderabad Campus

Context Free Grammars And Context Free Languages

(Chapter-3)

Concepts

Language Recognizer: A device that accepts valid strings.
Finite Automata are type of language Recognizers.

Language Generators: Devices that produce valid strings.
Ex: Regular Expressions.

Now we study certain types of formal language generators.

Language Generators

- That device begins when a signal to start is given to construct the string.
- It operation determined by a set of rules.
- Eventually this process halts and produces the completed string.
- The language defined by the device is set of all strings that it can produce.
- It is difficult to produce a recognizer for English language.

- We are interested in generators of artificial languages such as Regular Languages and CFL.
- Regular Expressions can be viewed as Language generators.

$$a(a^* \cup b^*)b$$

How to generate a string according to the above RE

- First output an a. Then do the following two.
- Either output a number of as or output number of bs.
- Finally output a b

- The language associated with this language generator is set of all strings that can be produced by the process above.

- Now we study more complex language generators called as *Context Free Grammars* (CFGs).

- CFGs are based on more complete understanding of the structure of the strings belonging to the language.

Ex: $a\ \underbrace{(a^* \cup b^*)}_{M}\ b$

↗ Start ↖ end

If S be the String in the Language and M be the Symbol for the middle part

$$S \rightarrow aMb$$

Read as 'can be'. We call such expression a Rule

What can 'M' be?

Obviously it is either String of a's or String of b's

We express this by adding a rule

$M \rightarrow A$

$M \rightarrow B$

A } are new symbols that stand for
B } strings of a's or b's respectively.

What is a string of a's, it can be 'e' also.

$A \rightarrow e$

$A \rightarrow aA$

Similarly $B \rightarrow e$

$B \rightarrow bB$

Then the language denoted by RE $a(a^* \cup b^*)b$ can be defined alternatively by the following language generator —

$S \to aMb$

$M \to A$

$M \to B$

$A \to e$

$B \to e$

$A \to aA$

$B \to bB$

1. Start with the string containing the symbol S.
2. Find a symbol in the current string that appear to the left of '⟶' in one of the rules above.
3. Replace an occurrence of this symbol with the string that appears to the right of '⟶' in the same rule.

Repeat this process until no such symbol can be found.

CFG

- In CFG symbols that do not appear on the LHS of a production rule are known as terminal symbols.
- In the process of producing a string, by using CFG, we see only terminal symbols in the string. Then we stop further replacements and the result is a valid string according to the language.

CFG Definition

- A CFG $G = (V, \Sigma , R, S)$

V is an alphabet

Σ is set of terminal symbols and subset of V

R set of rules where each rule $(V - \Sigma) \times V^*$

S is the start symbol

and is an element of $(V - \Sigma)$

CFG Definition

CFG for L = $\{a^n b^n : n \geq 0\}$ // it is not a Regular Language

CFG $G = (V, \Sigma, R, S)$

V = {S, a, b}

Σ ={a,b}

R ={ S→ aSb; S→e}

S =is the start symbol

CFG Definition

- A derivation in G of w_n from w_0 may be any string in V^*, and n is the length of the derivation , may be any natural number including zero.

- We say that the derivation has n steps.

CFG Definition

CFG for L = {$a^n b^n$: $n \geq 0$}   CFG G = (V, ∑ , R, S)

V = {S, a, b}   ∑ ={a,b}

R ={ S→ aSb; S→e}   S =is the start symbol

One possible derivation:

S→ aSb → aaSbb → aaaSbbb → aaabbb

w_0 w_1 w_2 w_3 w_4

Here the length of derivation is 4

CFG and PL

- Computer programs written in any language must satisfy some rigid criteria in order to be syntactically correct, and therefore amenable for mechanical interpretation.
- The syntax of most of the languages can be captured by CFG
- If a programming language is described by CFG, it will be easy for parsing.
- Parsing is the process of analyzing a program to find the syntax.

RL and CFL

All RLs are CFL.

$M = (K, \Sigma, \delta, s, F)$ *and corresponds to a Regular language L*

$G(M) = (V, \Sigma, R, S)$

$V = K \cup \Sigma$

$S = s$

$R = \{ Q \rightarrow aP \mid \delta(Q, a) = P \} \cup \{ Q \rightarrow e \text{ if } Q \in F \}$

Summary

- Language generators and Recognizers.
- What is CFL and CFG
- Significance of CFG
- Formal description of CFG
- Derivation
- CFL / CFG and RL
- Constructing CFG for FA