Theory of Automata
Context Free Grammars

Hafiz Tayyeb Javed

Week 9 Lecture 01

Contents

- Syntax As a Method for Defining Languages
- Symbolism for Generative Grammars
- Trees
- Lukasiewicz Notation
- Ambiguity
- The Total Language Tree

Context Free Grammars

- Three fundamental areas covered in the book are
    1. **Theory of Automata**
    2. **Theory of Formal Languages**
    3. **Theory of Turing Machines**

- We have completed the first area.

- We begin exploring the second area in this chapter.

Syntax as a Method for Defining Languages

- In Chapter 3 we recursively defined the set of valid arithmetic expressions as follows:

  Rule 1: Any number is in the set AE.

  Rule 2: If x and y are in AE, then so are

  (x), -(x), (x + y), (x - y), (x * y), (x/y), (x ** y)

where ** is our notation for exponentiation

- Note that we use parentheses around every component factor to avoid ambiguity expressions such as 3 + 4 - 5 and 8/4/2.
- There is a different way for defining the set AE: **using a set of substitution rules similar to the grammatical rules**.

Defining AE by substitution rules

- Start → AE

  AE → (AE + AE)

  AE → (AE - AE)

  AE → (AE * AE)

  AE → (AE/AE)

  AE → (AE ** AE)

  AE → (AE)

  AE → -(AE)

  AE → d

Example

- We will show that ((3 + 4) * (6 + 7)) is in AE

Start $\Rightarrow$ AE $\Rightarrow$ (AE * AE)

$\Rightarrow$((AE + AE) * (AE + AE))

$\Rightarrow$((3 + 4) * (6 + 7))

Definition of Terms

- A word that cannot be replaced by anything is called **terminal**.
  - In the above example, the terminals are the phrase AnyNumber, and the symbols + - * / ** ( )
- A word that must be replaced by other things is called **non-terminal**.
  - The non-terminals are Start and AE.
- The sequence of applications of the rules that produces the finished string of terminals from the starting symbol is called a **derivation** or a **generation** of the word.
- The grammatical rules are referred to as **productions**.

Symbolism for Generative Grammars

Definition:

- A **context-free grammar (CFG)** is a collection of three things:

1. An alphabet ∑ of letters called **terminals** from which we are going to make strings that will be the words of a language.

2. A set of symbols called **non-terminals**, one of which is the symbol S, standing for “start here”.

3. A finite set of **productions** of the form:

One non-terminal → finite string of terminals and/or non-terminals

where the strings of terminals and non-terminals can consist of only terminals, or of only non-terminals, or of any mixture of terminals and non-terminals, or even the empty string. We require that at least one production that has the non-terminal *S* as its left side.

Definition:

- The **language generated by a CFG** is the set of all strings of terminals that can be produced from the start symbol S using the productions as substitutions.
- A language generated by a CFG is called a **context-free language (CFL)**.

Notes:

- The language generated by a CFG is also called the **language defined by the CFG**, or the **language derived from the CFG**, or the **language produced by the CFG**.
- We insist that **non-terminals be designated by capital letters**, whereas **terminals are designated by lowercase letters and special symbols**.

Example

- Let the only terminal be *a* and the productions be

  Prod1 S → aS

  Prod2 S → Λ
- If we apply Prod 1 six times and then apply Prod 2, we generate the following:

  S ➔ aS ➔ aaS ➔ aaaS ➔ aaaaS

  ➔ aaaaaS ➔ aaaaaaS ➔ aaaaaaΛ = aaaaaa

- If we apply Prod2 without Prod1, we find that Λ is in the language generated by this CFG.
- Hence, this CFL is exactly a*.
- Note: the symbol "→" means "can be replaced by", whereas the symbol "➔" means "can develop to".

- Let the terminals be a and b, the only non-terminal be S, and the productions be

  Prod1 S → aS

  Prod2 S → bS

  Prod3 S → Λ

- The word *ab* can be generated by the derivation

  S ➔ aS ➔ abS ➔ abΛ = ab

- The word baab can be generated by

  S ➔ bS ➔ baS ➔ baaS ➔ baabS ➔ baabΛ = baab

- Clearly, the language generated by the above CFG is

  (a + b)*.

Example

- Let the terminals be a and b, the the non-terminal be S and X, and the productions be

  Prod 1 S → XaaX

  Prod 2 X → aX

  Prod 3 X → bX

  Prod 4 X → Λ

- We already know from the previous example that the last three productions will generate any possible strings of a's and b's from the non-terminal X. Hence, the words generated from S have the form

  anything aa anything

- Hence, the language produced by this CFG is

$$(a + b)^*aa(a + b)^*$$

which is the language of all words with a double a in them somewhere.

- For example, the word *baabb* can be generated by

$$S \Rightarrow XaaX \Rightarrow bXaaX \Rightarrow baaX \Rightarrow baaX$$

$$\Rightarrow baabX \Rightarrow baabbX \Rightarrow baabb\Lambda = baabb$$

Example

- Consider the CFG:

  $S \rightarrow aSb$

  $S \rightarrow \Lambda$

- It is easy to verify that the language generated by this CFG is the **non-regular** language $\{a^n b^n\}$.
- For example, the word a^4b^4 is derived by

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$

$\Rightarrow aaaaSbbbb \Rightarrow aaaa\Lambda bbbb = aaaabbbb$

Derivation and some Symbols

- If v and w are strings of terminals and non-terminals

$v \Rightarrow^{n} w$ » denotes the derivation of w from v of length *n* steps

$v \Rightarrow^{+} w$ » derivation of w from v in one or more steps

$v \Rightarrow^{*}_{G} w$ » derivation of w from v in zero or more steps of application of rules of grammar G.

Sentential Form

- A string w ε(n U ∑)* is a sentential form of G if there is a derivation

$$v \Rightarrow^* w$$

- A string w is a sentence of G if there is a derivation in G

$$v \Rightarrow^* w$$

- The language of G, denoted by L(G) is the set

$$\left\{ w \in \sum^* \;\; | \, S^* \overset{*}{\Rightarrow} w \right\}$$

Example

- It is not difficult to show that the following CFG generates the **non-regular** language $\{a^n b a^n\}$:

  S → aSa

  S → b

- Can you show that the CFG below generates the language PALINDROME, another **non-regular** language?

  S → aSa

  S → bSb

  S → a

  S → b

  S → Λ

Disjunction Symbol |

- Let us introduce the symbol | to mean disjunction (or).
- We use this symbol to combine all the productions that have the same left side.
- For example, the CFG

  Prod 1 $S \rightarrow XaaX$

  Prod 2 $X \rightarrow aX$

  Prod 3 $X \rightarrow bX$

  Prod 4 $X \rightarrow \Lambda$

can be written more compactly as

Prod 1 $S \rightarrow XaaX$

Prod 2 $X \rightarrow aX|bX|\Lambda$

Example

$S \rightarrow aSa \mid aBa$

$B \rightarrow bB \mid b$

- First production builds equal number of a's on both sides and recursion is terminated by $S \rightarrow aBa$
- Recursion of $B \rightarrow bB$ may add any number of b's and terminates with $B \rightarrow b$

- $L(G) = \{a^n b^m a^n \; n>0, m>0\}$

example

$L(G) = \{a^n b^m c^m d^{2n} \mid n>0, m>0\}$

- Consider relationship between leading a's and trailing d's.

$$S \rightarrow aSdd$$

In the middle equal number of b's and c's

- $S \rightarrow A$
- $A \rightarrow bAc$
- This middle recursion terminates by $A \rightarrow bc$.

- Grammar will be

$$S \rightarrow aSdd \mid aAdd$$

$$A \rightarrow bAc \mid bc$$

Example

Consider another CFG

$S \rightarrow aSb \mid aSbb \mid \Lambda$

- Language defined is

$L(G) = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$

Example

- A grammar that generates the language consisting of even-length string over {a, b}

  S → aO | bO | Λ

  O → aS | bS

- S and O work as counters i.e. when an S is in a sentential form that marks even number of terminals have been generated

- Presence of O in a sentential form indicates that an odd number of terminals have been generated.

- The strategy can be generalized, say for string of length exactly divisible by 3 we need three counters to mark 0, 1, 2

  S → aP | bP | Λ
  P → aQ | bQ
  Q → aS | bS

Regular Grammar

Given an FA, there is a CFG that generates exactly the language accepted by the FA.

- In other words, all regular languages are CFLs