Automata Theory - Lecture 11

Regular Languages and Context Free Languages

Lecturer: Martha Gichuki

Lecture learning outcomes

At the end of the lecture you will be able to:

(i) Describe Regular and Context Free Languages
(ii) Derive languages using Parse Trees
(iii) Formally describe Context Free Languages

Automata theory is closely related to **formal language theory** because automatons are often classified by the class of formal languages, they are able to recognize.

An automaton is always anchored on the basic concepts of **symbols, words alphabets and strings.** These are defined briefly as follows: -

Symbol – A character or letter (An arbitrary datum) that has some meaning to or effect on the machine. Symbols are sometimes called letters.

Word – A finite string formed by the concatenation of a number of symbols.

Alphabet – A finite set of symbols. It is frequently denoted by Σ, which is the set of letters in an alphabet.

Language - refers to a set of words formed by symbols in a given alphabet.

Kleene Closure – A language may be thought of as a sub-set of possible words. The set of all possible words may in turn be thought of as the set of all possible concatenations of strings. Formally, this set of all possible

strings is called a free monoid. It is denoted as * and the super script $^{(*)}$ is called the **Kleene Star**.

Forma Definition of Context Free Grammar (CFG)

A Context Free Grammar is a 4-tuple (V, T, S, P} where: -

(i) V – is a finite set of Variables
(ii)T – is a finite set of Terminals
(iii) S – is the Start Variable
(iv) P – is a finite set of production rules

Recall from Lecture Eight that a PDA recognizes Context Free Languages and these ae built upon Context Free Grammar. In a natural language like English, understanding a sentence begins with understanding the grammatical structure meaning knowing how it is derived from the grammar rules of the language.

Given a context free grammar e.g. one specifying the syntax of a programming language and a string that is derivable from the grammar it is often useful to know a derivation because that is what allows us to interpret the strings correctly. A natural way to represent these derivations is by use of derivation trees.

A natural way of exhibiting any structure is by deriving it using a derivation tree. The root of the tree represents the variable with which we begin the derivation (Start Variable). Another name for a derivation tree is a **parse tree**.

Each interior node of the tree corresponds to the variables of the grammar and its children correspond to the symbols in the entire string appearing on the right-hand side of the production rules.

The symbol → is used to mean "Could take the value".

Derivation of Languages:

This refers to the sequence of rules that produce the finished string of terminals from the start symbol. This is called derivation or production. The language f a CFG is the set of terminal symbols which we can derive using specified production rules.

Example 1:

Given the terminal – a

Non-terminal – S

Production rules S → aS, S→∧ where ∧ (empty string), the derivation for aaaa could be given as

S→ aS

→aaS

→aaaS

→aaaaS

→aaaa∧→ aaaa

Example 2:

Given the terminal a;

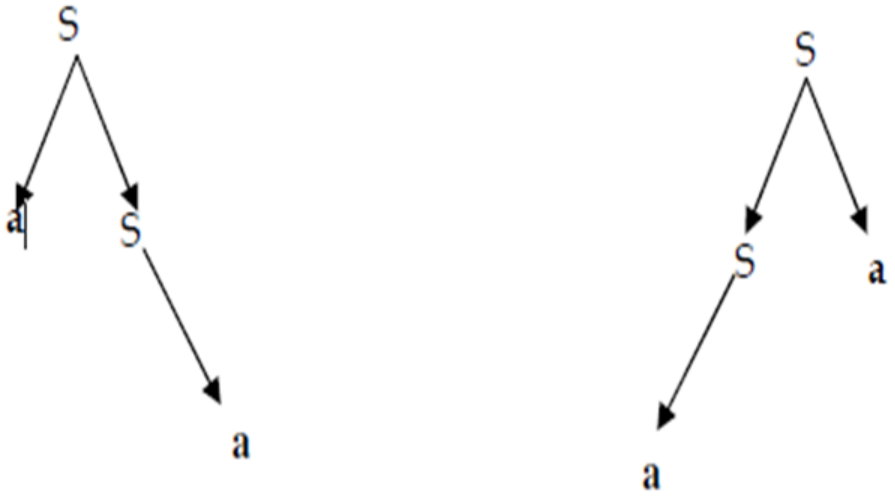
Non – terminal S

Productions: S→ aS / Sa / a

Derive the Tree for the string aa

Solution

The word "**aa"** can be generated using two trees: -



Example 3:

Consider a context free grammar that defines the language pal of all palindromes over the alphabet ∑ {a, b} using the following production rules: -

(i.) (S→∧) - S could take the null value

(ii.) (S→a/b) - S could take the value a or b

(iii.) (S→aSa/bSb) - S could take the form aSa or bSb

From these rules we can write S→aSa→abSba→ab∧ba → abba (which are all palindromes)

To summarize the 3-steps derivation of abba in which rule two is applied twice and rule one is applied once, using the → notation. The rules can be written as: -

S→a/b/∧

S→aSa/bSb

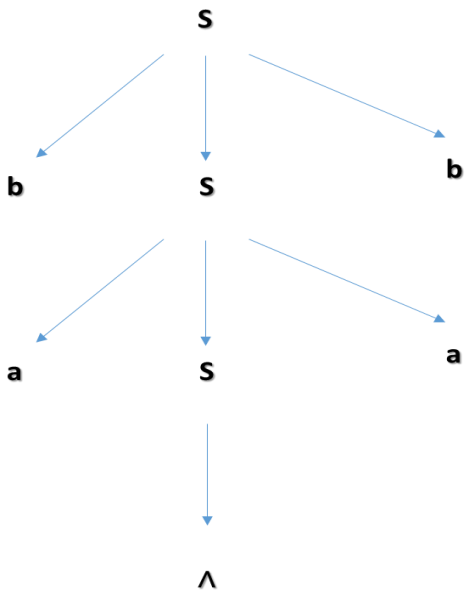
S→aSa→bSb→ab∧ba=abba

In an expression such as aSa or bSb, the two alternatives are aSa & bSb and not a & b direct.

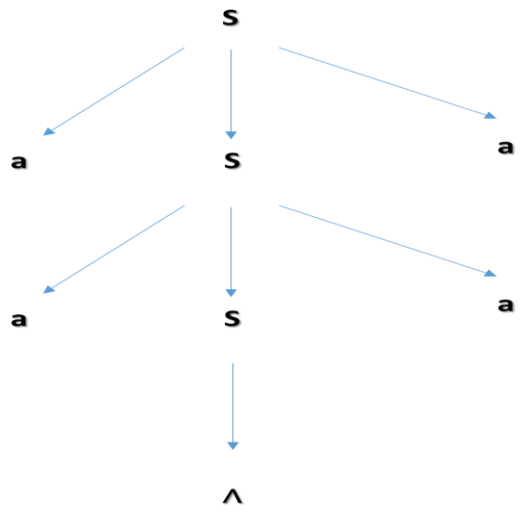
Assignment:

Derive the following strings using the above production rules:

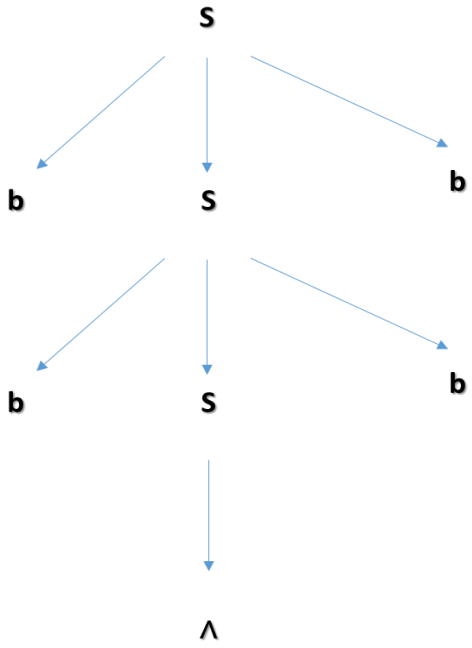
a) baab



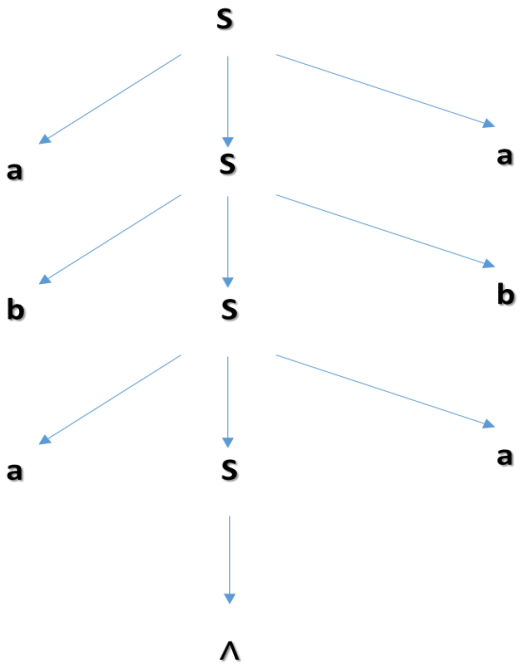
b) aaaa



c) bbbb



d) abaaba

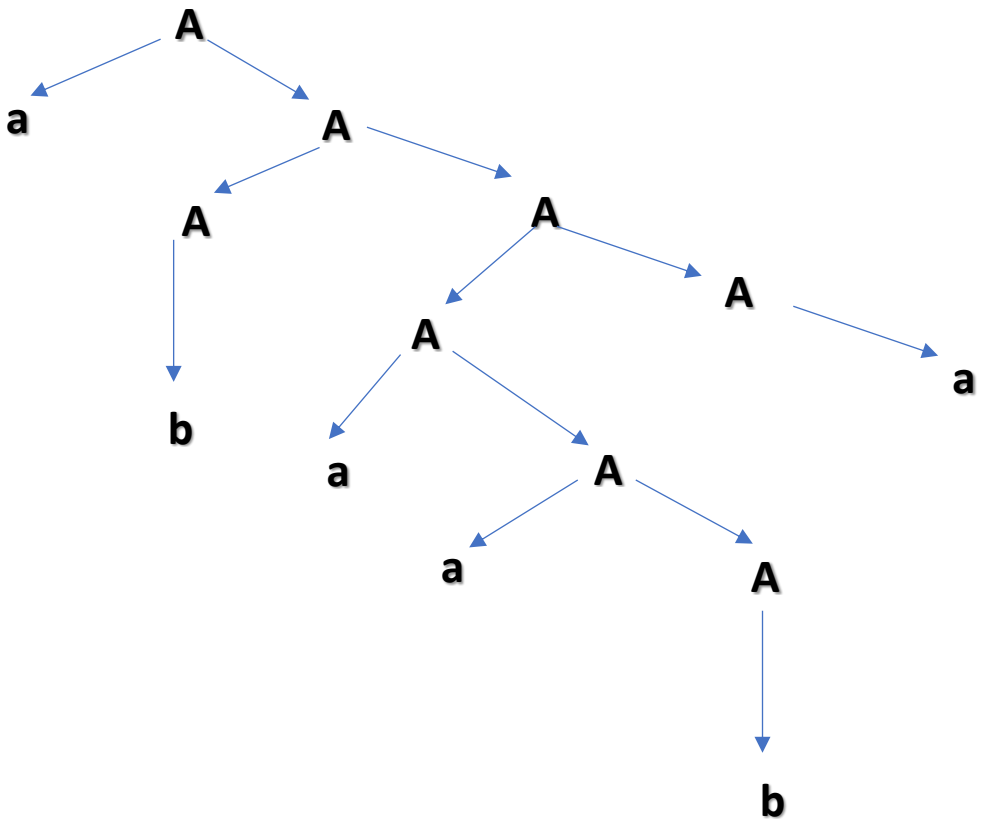


Example 4:

Given a CFG with the following definitions

i. Terminals a,b
ii. Non-terminals: S, A
iii. Production Rules
   - ✓ S→AAA/AA
   - ✓ A→AA/aA/Ab/A/B

String abaaba has the derivation tree:



When we concatenate the leaves of any derivation tree, we get a string which is known as the yield of the tree. The above tree has 5 yields.

Example 5:

If we are given the string "a+a+a" with the following production rules: -

(i.) S→S+S

(ii.) S→a

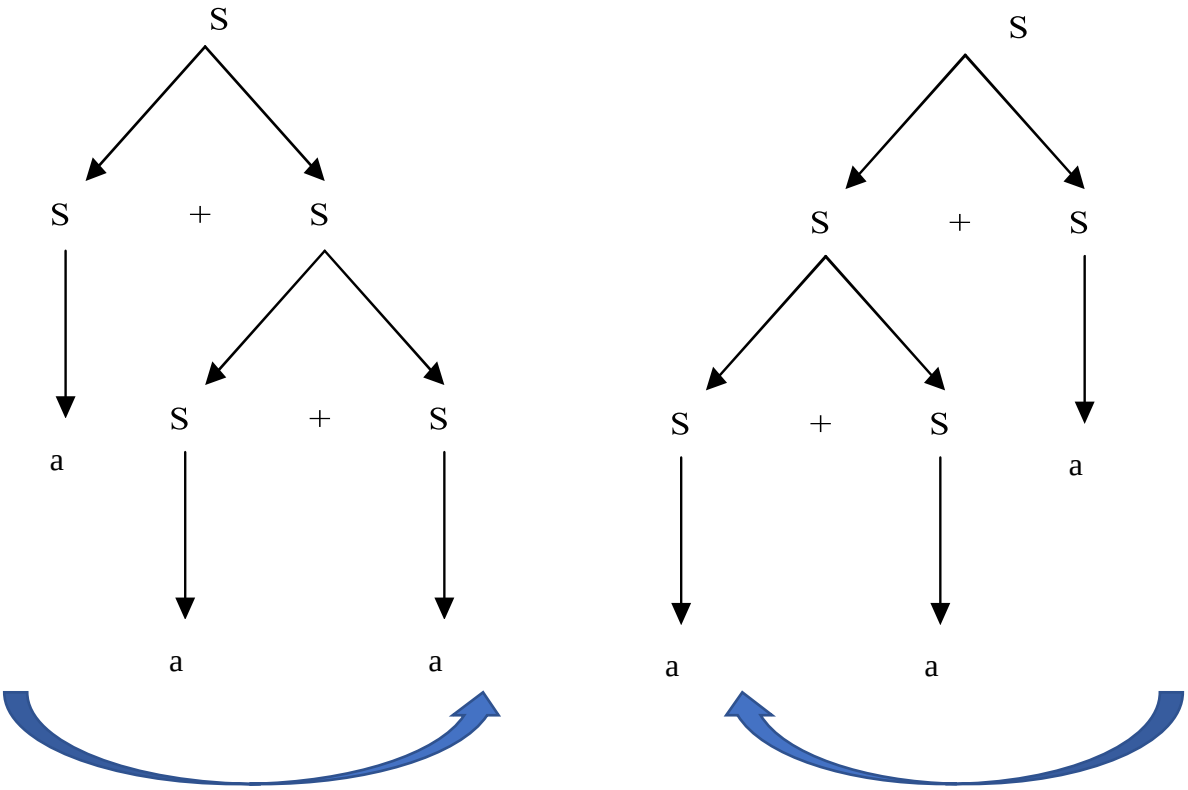
Left most derivation tree for some string – This is the process that looks at the string from Left to Right following the production rules provided.

The left most derivation takes the format: -

S→S+S→a+S→a+S+S→a+a+S→a+a+a=a+(a+a)

Right most derivation looks at the string from Right to Left. S→S+S→S+S+S→a+S+S→a+a+S→a+a+a=(a+a)+a)

We can exhibit this structure by using left - a+(a+a) and right - (a+a) + a derivation trees as follows: -

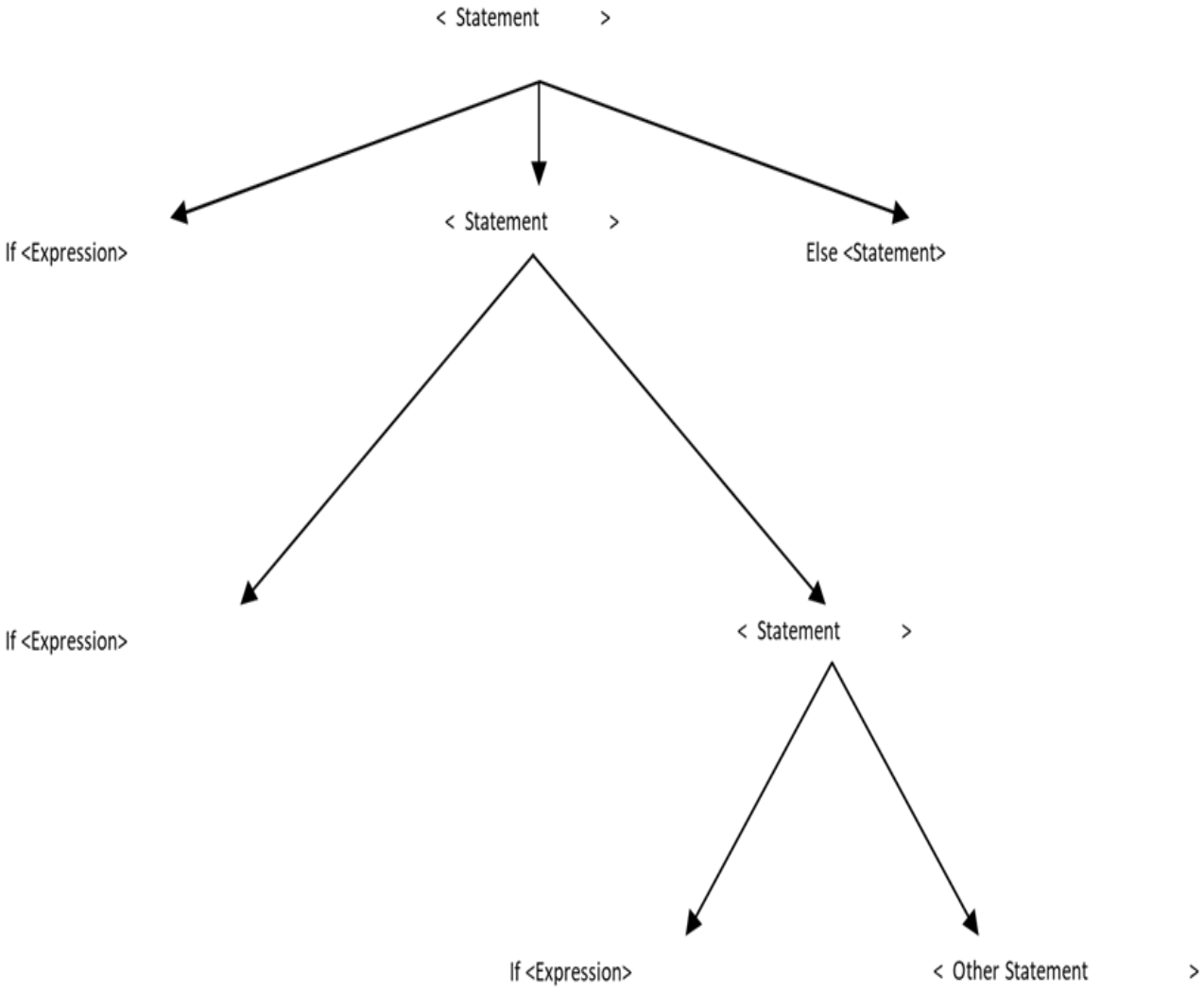


A two or more distinct derivation trees. The above CFG is said to be ambiguous because it generates two distinct derivation trees. (Note that there could be more than two).

A standard example of ambiguity is the If else Statements in Programming languages.

Consider implementing the following problem in a Visual Basic programming language: If Gender= M then male title is selected and if Gender=F then female title is selected.

We could go on and check if the person is above 18 years and the comment "Issue ID" is generated else "Deny ID" comment is generated.



References

1 Rowan G. & John T., (2009), *Discrete Mathematics: Proofs, Structures and Applications*, CRC Press, ISBN: 9781439812808.

2 W. D. Wallis (2003), *A Beginners Guide to Discrete Mathematics*, Springer Science & Business Media, ISBN: 978-0817642693.

3 Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012).

4 Introduction to languages and the theory of computation (3rd ed.), Martin, J., New York: McGraw-Hill. ISBN-13: 978-0072322002, (2002)