Course: Compiler Construction

Week 4: Compiler Design Phases – Syntax Analysis

Lecturer: Martha Gichuki

Course Description

- The course begins with a coverage of basic Principles in Compiler Design with an introduction to Formal programming language translation and compiler design concepts; compilers and interpreters.
- A coverage of Language theory, Parsing Context-Free Languages (Top - down and bottom - up parsing), translation specifications and machine- independent code optimization will then follow.

Course Description …

- The main phases of compilation i.e. Lexical, Syntax and Semantic analysis will be taught.
- Code generation and Optimization, Symbol table design, Program compilation, Loading and execution will then be covered.
- Compilation techniques, Optimization, Design of a simple complete compiler. will culminate the course.

Learning outcomes Week 4: Compiler Design Phases – Syntax Analysis

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

(i) Define Parser, Grammar and Ambiguity

(ii)Describe Context Free Grammar (CFG)

(iii) Derive strings using parse trees

Syntax Analysis

- ✓Another term for Syntax Analysis is **Parsing**
- ✓This is the process of analyzing a sequence of tokens to determine their **grammatical structure** with respect to a **given formal grammar**[1].

1. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 216

Syntax Analysis intro…

- ✓ It is the most important phase of a compiler.
- ✓ A **syntax analyzer (parser)** checks for the correct syntax and builds a data structure (**parse tree**) implicit in the input tokens i.e. it considers the sequence of tokens for possible valid constructs of the programming language[2].

2. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 217

First Role of a Parser

1. To identify language constructs present in a given input program.

✓ If the parser determines input to be **valid**, it outputs the presentation of the input in form of a **parse tree**[3]

3. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 217

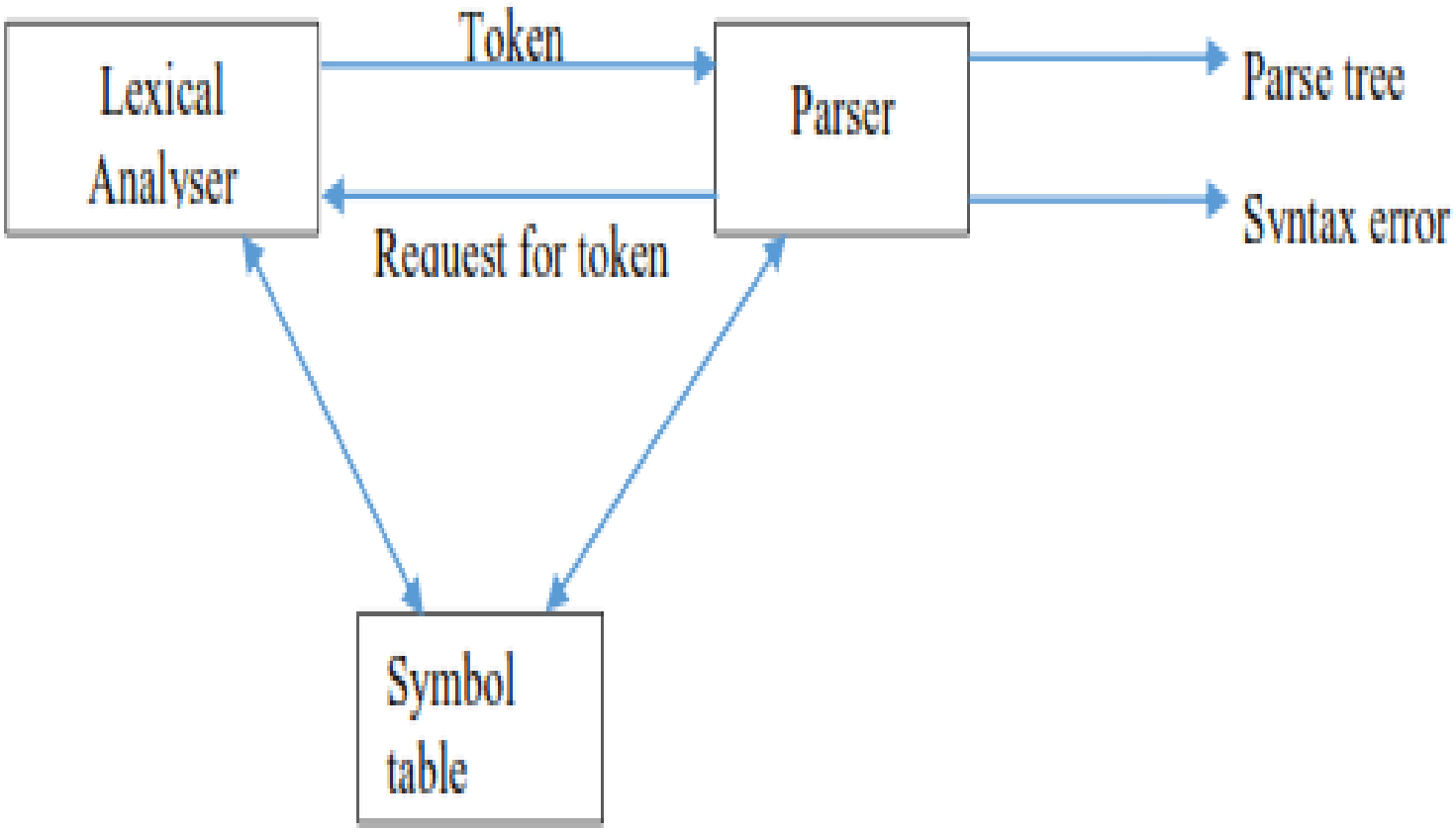
Second Role of a Parser

2. If the input is grammatically **incorrect**, the parser declares the detection of **syntax error** in the input.

✓ In this case, a parse tree is **not produced**[4].

4. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 217

Parser Roles Illustration[5]



5. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 217

Grammar

A grammar G is defined as a **four-tuple** with **<V_N, V_T, P, S>** where:-

I. **V_N** is the set of **non-terminal symbols** used to write the grammar[6].

6. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 222

II. V_T is the **set of terminals** (set of words of the language, lexicon or dictionary of words).

III. **P** is the **set of production rules** that define how a sequence of terminal and non-terminal symbols can be replaced by some other sequence[7]

7. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 222

IV. S: - S Є V_N is a special non-terminal called the **start symbol** of the grammar.

- ✓ The language of the grammar G= <V_N, V_T, P, S> denoted by **L(G)** is defined as all those strings **over V_T** that can be generated by starting with the start symbol **S** then applying the production rules in **P** until no more non-terminal symbols are present[8].

8. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 222

Example: Consider the grammar to generate arithmetic expressions consisting of numbers and operator symbols i.e. +, -, *,/.

Rules of the grammar can be written as follows[9]:-

$$E \rightarrow EAE$$
$$E \rightarrow (E)$$
$$E \rightarrow number$$
$$A \rightarrow +$$
$$A \rightarrow -$$
$$A \rightarrow *$$
$$A \rightarrow /$$

9. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 193

- ✓ We can apply these rules to derive the expression **2 ∗ (3 + 5 ∗ 4)** as follows:-

$$E \rightarrow EAE \rightarrow EA(E) \rightarrow EA(EAE) \rightarrow EA(EAEAE) \rightarrow EA(EAEA\ 4) \rightarrow EA(EAE * 4) \rightarrow EA(EA\ 5 * 4) \rightarrow EA(E + 5 * 4) \rightarrow EA(3 + 5 * 4) \rightarrow E * (3 + 5 * 4) \rightarrow 2 * (3 + 5 * 4)$$

- ✓ In the grammar, ***E*** and ***A*** are **non-terminals** while the rest are **terminals**[10].

10. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 193

Context Free Grammar (CFG)

- ✓This is grammar that **defines** Context Free Languages and consist of production rules in which:-

i. the left hand side contains only **a single non-terminal and no terminals**;

ii. the right hand side consists of **either terminals, non-terminals or both**.

- ✓Note: Most programming language constructs belong to Context Free Languages[11]

11. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 193

Notational Conventions[12]

The following is the notational convention that would be used when defining grammars:

1. The following will be taken as **terminals**:

- ✓ all operator symbols
- ✓ punctuation symbols including parenthesis '()'
- ✓ digits
- ✓ lower case letters of the alphabet such as a, b, c
- ✓ Lexemes such as id, number, while, etc.

12. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 223

2. The following will represent **non-terminals**:

- ✓Upper case letters of the alphabet such as A, B, C.
- ✓The letter S will represent the start symbol
- ✓Lowercase names such as expr, stmt etc[13].

13. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 224

3. A set of **productions** with the same left hand side such as: $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2$ … $A \rightarrow \alpha_n$ will be written as $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$.
4. If no start symbol is represented, then the non-terminal appearing on the left hand side of the production rule will be considered as the **start symbol**[14]

14. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 224

Derivation

- ✓This is the process of generating a sequence of intermediary strings to expand the start symbol of the grammar to the desired string of terminals.
- ✓The **graphical depiction of a derivation** is a parse tree[15].

15. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 224

Derivation …

- ✓Each step of the derivation modifies an intermediary string to a new one by replacing a substring of it that matches the **left hand side** of the **production rule** by a string on **the right hand side of the rule**[16].

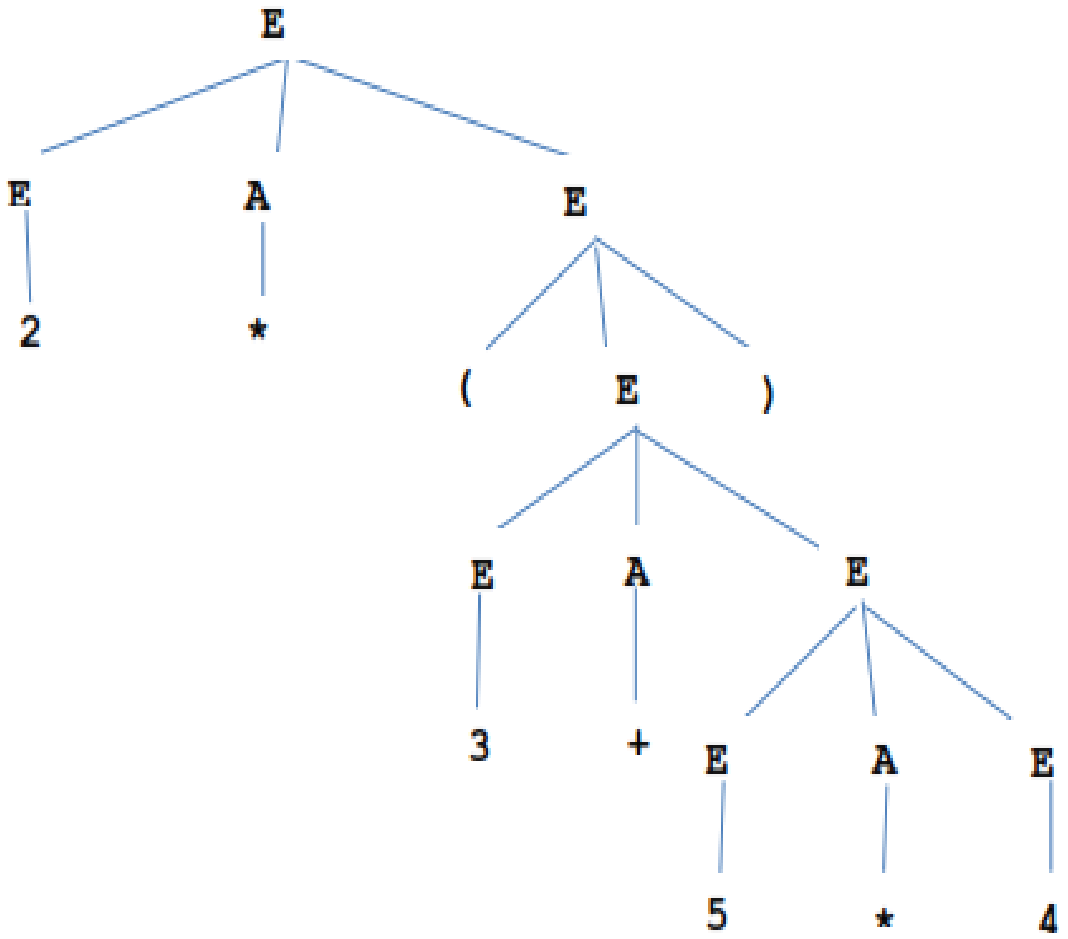
16. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 225

In a parse tree

- ✓The **start symbol** of the derivation becomes the **root**.
- ✓All **leaf nodes** are **terminals**
- ✓All **interior nodes** are **non-terminals**
- ✓**In-order traversal** gives original **input string**
- ✓A parse tree depicts **associativity and precedence of operators**[17]

17. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 226

Example: The following is the derivation of the string 2∗(3+5∗4) [18]



18. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 227

Associativity & Precedence

A. Associativity

- ✓If an operand has operators on both sides, left associativity dictates that the operand be taken by the left operator
- ✓If the operation is right-associative, the right operator will take the operand[19]

19. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 218

B. Precedence

- ✓To decrease the chances of ambiguity in a language or its grammar, **operands in brackets have the highest precedence**, followed by division, multiplication, addition and **subtraction comes last**[20]

20. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 218

Types of Derivation

There are two types of derivation:-

i)**Left-Most Derivation**

- ✓Here, the left-most non-terminal is replaced at each step.
- ✓The intermediate strings are called **left-sentential forms.** They consist of grammar symbols (both terminal & non-terminals) [21]

21. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 199

ii) Right-Most Derivation

- ✓In this derivation the right-most non-terminal is replaced at each step.
- ✓The intermediary strings are called **right-sentential forms.**
- ✓Right-most derivation is often referred to as **canonical representation**[22].

22. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 201

Ambiguity

- ✓A grammar is said to be **ambiguous** if there exists **more than one parse tree for the same sentence**[23]

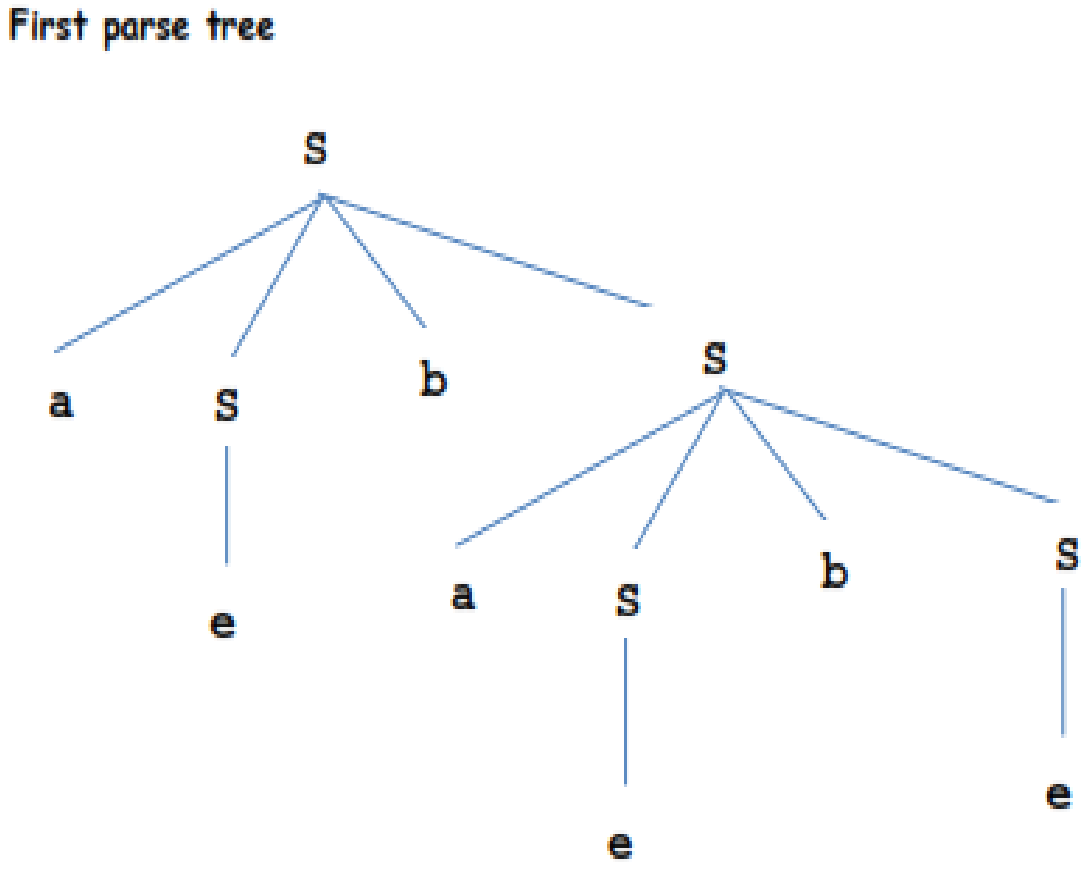
Example:- Consider the following grammar: $S \rightarrow aSbS \mid bSaS \mid e$

- ✓To show that the grammar is ambiguous, we present two different parse trees for the string "***abab***".

23. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 186

Consider the following grammar: $S \rightarrow aSbS \mid bSaS \mid e$

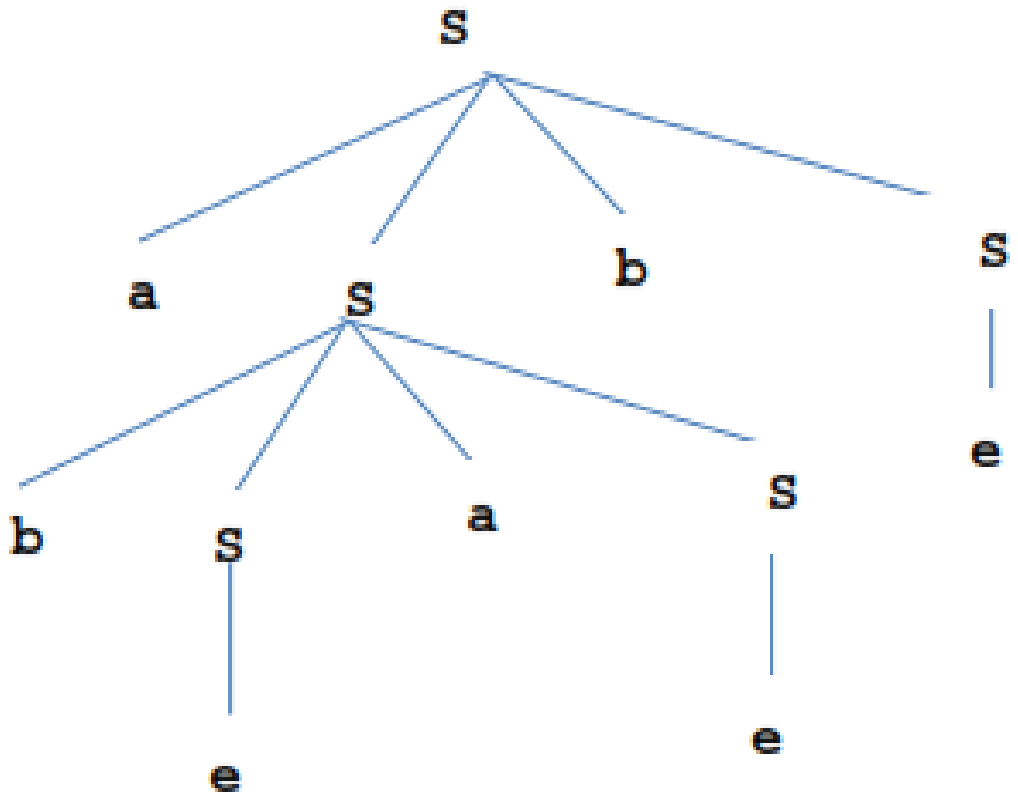
1. First Parse Tree for the string "$abab^{24}$"



24. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 204

Consider the following grammar:
$S \rightarrow aSbS \mid bSaS \mid e$

2. Second Parse Tree for the string "$abab^{25}$"



25. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 204

Dangling Else Ambiguity

- ✓Most programing languages have both if… then and if… then … else versions of the statement.
- ✓The grammar rules are as follows:-

stmt → if condition then stmt else stmt

| if condition then stmt[26]

26. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison-Wesley Pub Co, ISBN: 0201100886 (2007) page 235

Consider the following code segment

$$if\ a > b\ then$$

$$if\ c > d\ then\ x = y$$

$$else\ x = z$$

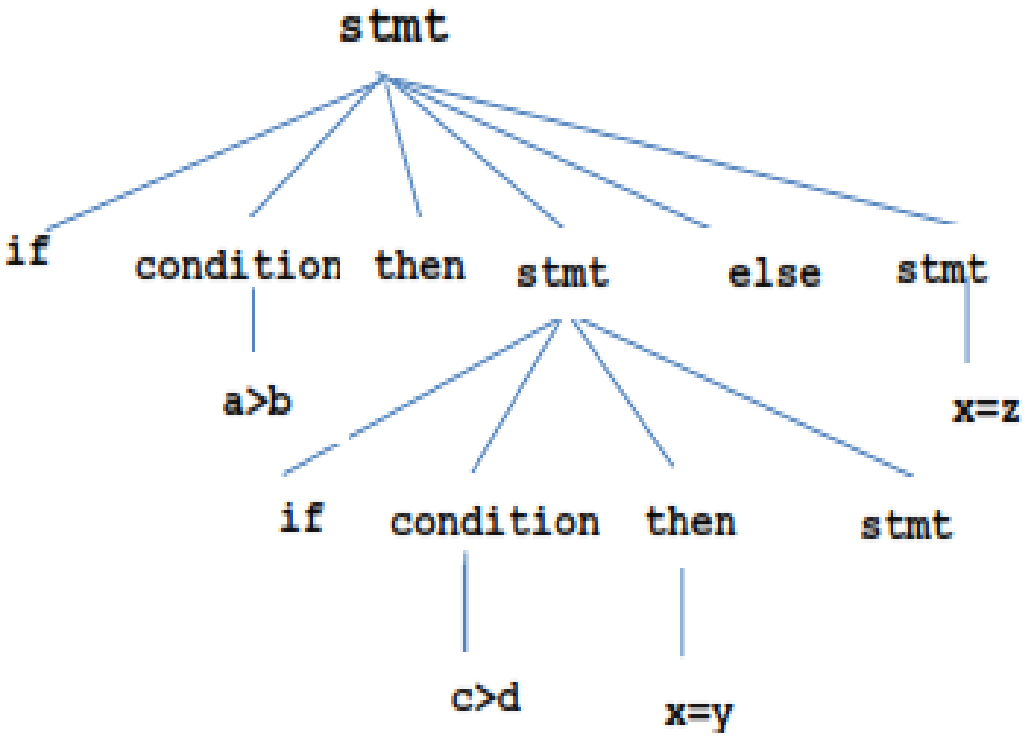
✓Two parse trees can be generated by the grammar as shown below[27]:

27. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 235

i) the else is taken with the outer if statement[28]. $if\ a > b\ then$

$$if\ c > d\ then\ x = y$$

$$else\ x = z$$



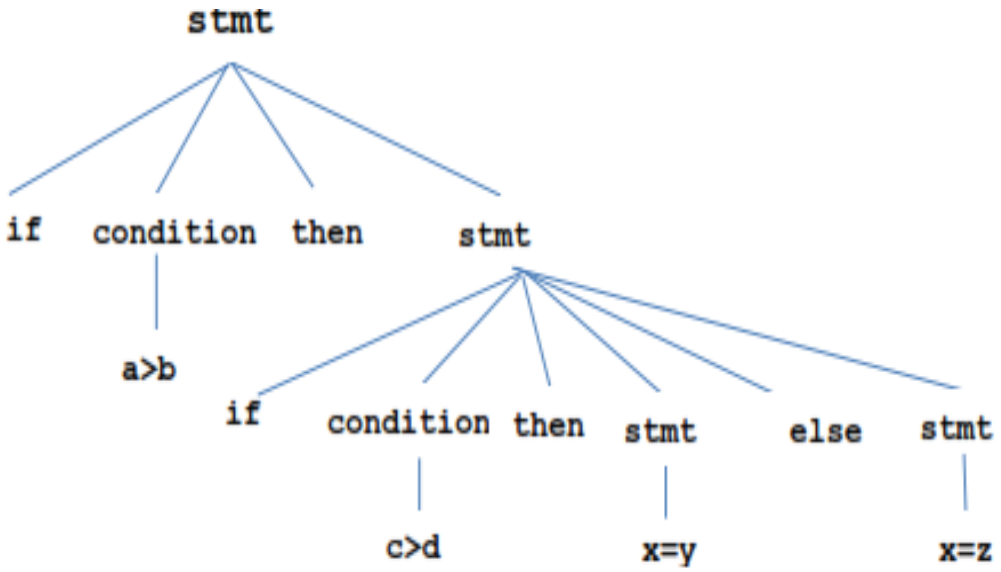
28. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 236

ii) the else is taken with the inner if statement[29].

$$if\ a > b\ then$$

$$if\ c > d\ then\ x = y$$

$$else\ x = z$$



✓**Note:** most programming languages accept this second parse tree as the correct syntax.

29. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 237

Eliminating Ambiguity[30]

Ambiguity may be eliminated by:-

i. Rewriting the grammar

ii. Modifying the grammar e.g. Many programming languages require that "if" should have a matching "end if " as shown below:-

$$stmt \rightarrow if\ condition\ then\ stmt\ else\ stmt\ endif$$

$$| \ if\ condition\ then\ stmt\ endif$$

30. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; Addison- Wesley Pub Co, ISBN: 0201100886 (2007) page 235

Content Covered in Week 4: Compiler Design Phases – Syntax Analysis

At the end of the lecture, we were able to:

i. Define Parsers, Grammar and Ambiguity
ii. Describe Context Free Grammar (CFG)
iii. Derive strings using parse trees

Course Text Books

1. Compilers: Principles, Techniques and Tools, Aho, Lam, Sethi, and Ullman; *Addison- Wesley Pub Co,* ISBN: 0201100886 (2007))
2. Compiler Construction: Principles and practices; Kenneth C. Louden; *Cengage Learning*; 1st edition, ISBN-10 : 0534939724 (1997)
3. Basics of Compiler Design: Torben Mogensen; *DIKU University of Copenhagen Universitetsparken 1 DK-2100 Copenhagen DENMARK* (2007)
4. Engineering a Compiler: 2nd edition; Keith D. Cooper and Linda Torczon; *Morgan Kaufmann Publishers* ISBN: 978-0-12-088478-0 (2003)
5. Compiler Design: Santanu Chattopadhyay; *PHI Learning Publishers*, ISBN 812032725X, 9788120327252 (2005)