

Compiler Construction Course

Lecture 3 *Syntax Analysis* *A Review on Formal Grammars*

Sayyed Kazem Shekofteh

The Basics

- What is a *grammar*?
 - Describing and analyzing a language.

sentence	->	<subject> <verb-phrase> <object>
subject	->	This Computers I
verb-phrase	->	<adverb> <verb> <verb>
adverb	->	never
verb	->	is run am tell
object	->	the <noun> a <noun> <noun>
noun	->	university world cheese lies

- Using the above rules or productions, we can derive simple sentences such as these:
 - This is a university.
 - Computers run the cheese.
 - I never tell lies.

The Basics

- Here is a leftmost derivation of the first sentence using these productions.

sentence -> **<subject> <verb-phrase> <object>**
 -> This **<verb-phrase> <object>**
 -> This **<verb> <object>**
 -> This is **<object>**
 -> This is a **<noun>**
 -> This is a university

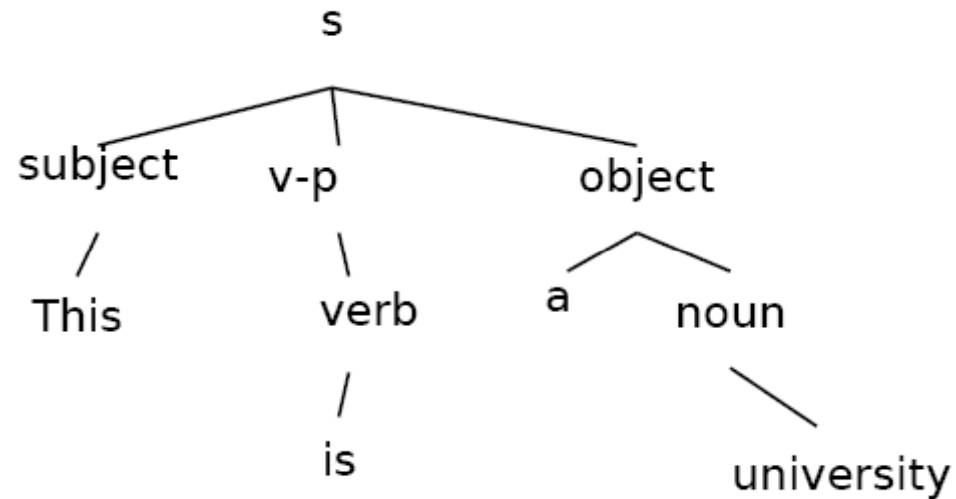
- Formal grammars are a tool for *syntax*, not *semantics*.
- In the syntax analysis phase, we verify structure, not meaning.

The Basics

- Grammar components.
 - *Nonterminal*
 - *Terminal*
 - *Production*
 - *Derivation*
 - *Start Symbol*
 - *Null symbol ε*
 - *BNF*

The Basics

- Parse representation
 - Derivation
 - Left-most
 - Right-most
 - Parse tree



- *it does not encode the order that productions were applied.*

Ambiguity

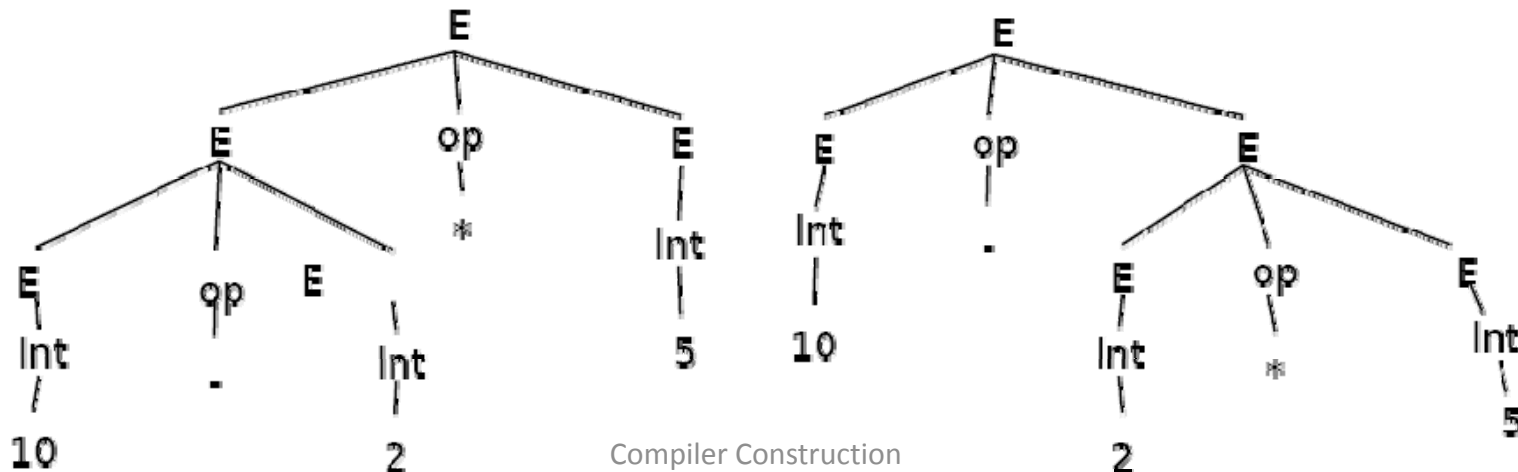
- Which grammar is ambiguous?

- If a grammar permits more than one parse tree for some sentences.

$E \rightarrow E \text{ op } E \mid (E) \mid \text{int}$
 $\text{op} \rightarrow + \mid - \mid * \mid /$

- This grammar denotes expressions that consist of integers joined by binary operators and possibly including parentheses

- Sample: $10 - 2 * 5$



Ambiguity

- It is an undecidable problem to determine whether any grammar is ambiguous.
- Much less to attempt to mechanically remove all ambiguity.
- We usually take pains.
 - *Nonterminal*
 - *Terminal*

Recursive Rules

- Productions are often defined in terms of themselves.
- For example a list of variables in a programming language grammar could be specified by this production:

$\text{variable_list} \rightarrow \text{variable} \mid \text{variable_list} , \text{variable}$

- Left
- Right

$X \rightarrow Xa \mid Xb \mid AB \mid C \mid DEF$



$X \rightarrow ABX' \mid CX' \mid DEFX'$
 $X' \rightarrow aX' \mid bX' \mid \epsilon$

Left Factoring

- Productions that have common first symbol(s)
- For example a list of variables in a programming language grammar could be specified by this production:

Stmt \rightarrow if Cond then Stmt else Stmt | if Cond then Stmt | Other |



Stmt \rightarrow if Cond then Stmt OptElse | Other | ...
OptElse \rightarrow else S | ϵ

Hidden Left Factoring

A \rightarrow da | acB
B \rightarrow abB | daA | Af

A \rightarrow da | acB
B \rightarrow abB | daA | daf | acBf

A \rightarrow da | acB
B \rightarrow aM | daN
M \rightarrow bB | cBf
N \rightarrow A | f

Hidden Left Recursion

$$\begin{aligned} S &\rightarrow Tu \mid wx \\ T &\rightarrow Sq \mid vvS \end{aligned}$$
$$\begin{aligned} S &\rightarrow Tu \mid wx \\ T &\rightarrow Tuq \mid wxq \mid vvS \end{aligned}$$
$$\begin{aligned} S &\rightarrow Tu \mid wx \\ T &\rightarrow wxqT' \mid vvST' \\ T' &\rightarrow uqT' \mid \varepsilon \end{aligned}$$