Compiler Construction (CS-636)

Sadaf Manzoor
UIIT, Rawalpindi

Outline

1. The Scanning Process
    1. The Function of the Scanner
    2. The Categories of Tokens
    3. Relation between Tokens and its String
    4. Some Practical Issues of the Scanner
2. Types of Errors
3. Language Specifications
4. Regular Expressions
5. Summary

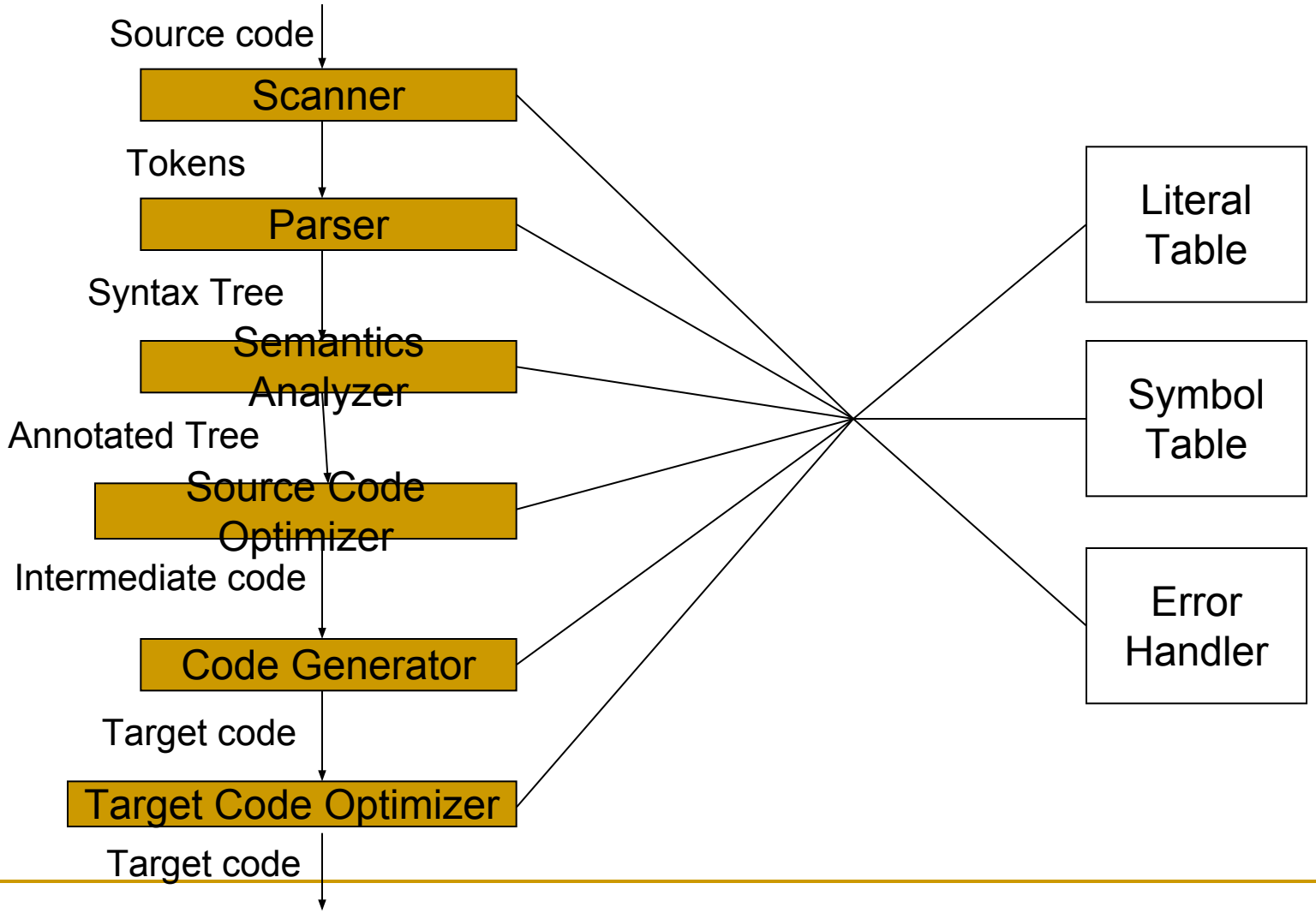
Lexical Analysis (Scanning)

Lecture: 3 & 4

Compiler Components

- Six Components
  1. Scanner
  2. Parser
  3. Semantic Analyzer
  4. Source Code Optimizer
  5. Code Generator
  6. Target Code Optimizer
- Three Auxiliary Components
  1. Literal Table
  2. Symbol Table
  3. Error Handler

The Compilation Process



The Function of the Scanner

- To read characters from the source code and form them into logical units called Tokens
- Tokens are logical entries that are usually defined as an enumerated type;
  - For example in C, we may define it as;

```
Typedef enum
   {IF, THEN, ELSE, PLUS, NUM, ID,…}
   TokenType;
```

The Categories of the Tokens

- RESERVED WORDS
  - Such as IF and THEN, which represent the strings of characters “if” and “then”
- SPECIAL SYMBOLS
  - Such as PLUS and MINUS, which represent the characters “+” and “-“
- OTHER TOKENS
  - Such as NUM and ID, which represent numbers and identifiers

Relationship between Tokens and its String

- Token string is called STRING VALUE or LEXEME
- Some tokens have only one lexeme, such as reserved words
- A token may have infinitely many lexemes, such as the token ID.
- Any value associated to a token is called an attributes of a token
  - A NUM token may have a string value such as “32767” and actual value 32767
  - A PLUS token has the string value “+” as well as arithmetic operation +

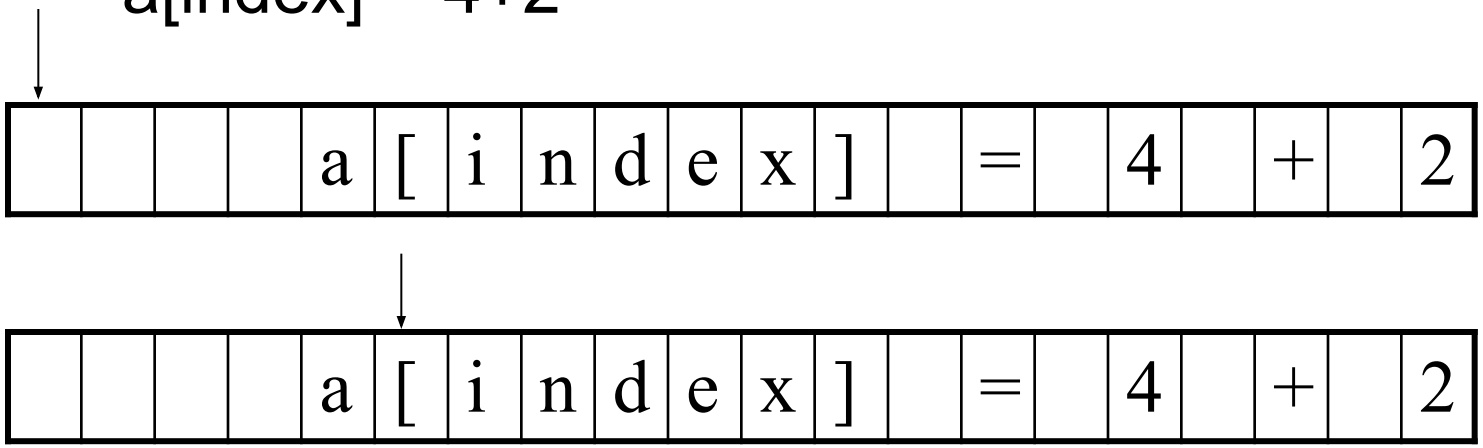
Relationship between Tokens and its String (Continue…)

- The token can be viewed as the collection of all of its attributes
  - Only need to compute as many attributes as necessary to allow further processing
  - The numeric value of a NUM token need not compute immediately

Some Practical Issues of Scanner

- One structured data type to collect all the attributes of a token, called a token record
- The scanner returns the token value only and places the other attributes in variables
- Scanner may not scan all source code at once

a[index] = 4+2



Types of Errors

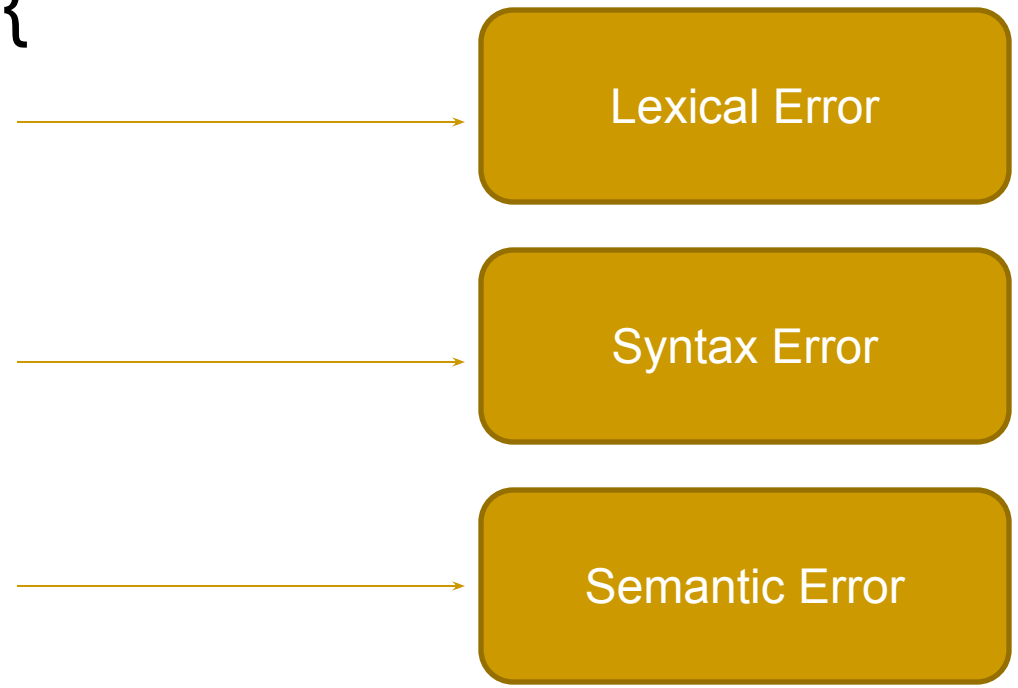
- During compilation, a compiler catches three types of errors:
  1. Lexical Errors
  2. Syntax Errors
  3. Semantic Errors

Types of Errors

- public class CC{
  int #a = 10;}

- if (x<10 (

- string s = 100;



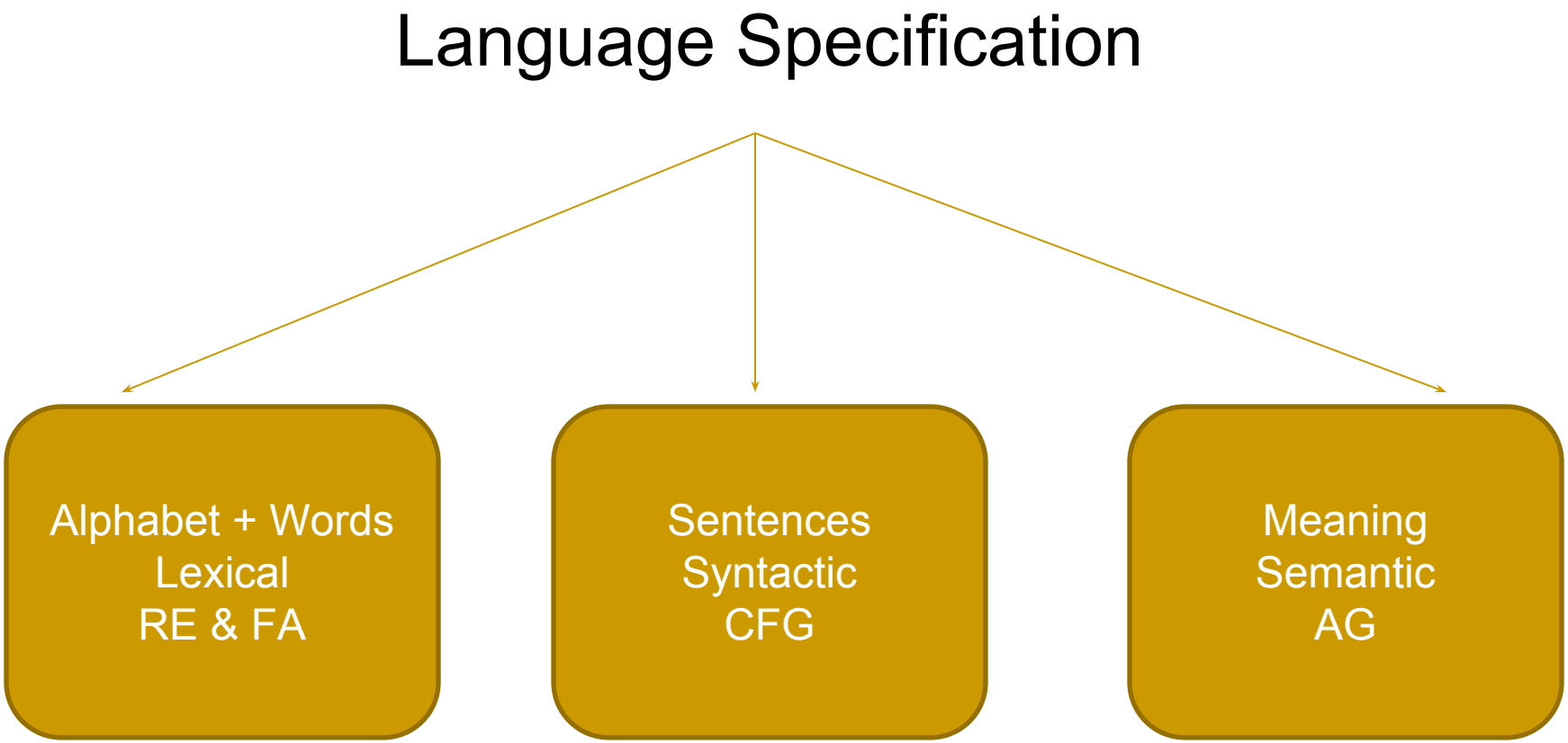
Language

- Letters, Words, Sentences
- Alphabets join to form words
- Words combine to form sentences
- Sentences combine to form paragraphs and so on

Languages

- How can you tell whether a given sentence belongs to a particular languages
  - Black is cat the
  - The tea is hot
  - I like chocolates two much
- Rules give a clue to forming as well as validating sentences

Language Specification



Regular Expressions

Regular Expressions

- Regular expressions
  - represent patterns of strings of characters.
- A regular expression *r*
  - completely defined by the set of strings it matches.
  - The set is called the language of r written as L(r)
- The set elements
  - referred to as symbols
- This set of legal symbols
  - called the alphabet and written as the Greek symbol ∑

Regular Expressions (Continue…)

- A regular expression *r*
  - contains characters from the alphabet, indicating patterns, such *a* is the character a used as a pattern
- A regular expression *r*
  - may contain special characters called *meta-characters* or meta-symbols
- An *escape character* can be used to turn off the special meaning of a meta-character.
  - Such as backslash and quotes

Regular Expression Operations

- Choice among alternatives, indicated by the meta-character: +
- Concatenation, indicated by juxtaposition without having any meta-character: ab
- Repetition or “closure”, indicated by the meta-character: *

Regular Expressions (Continue…)

- The symbols that appear in the regular expressions are
  - the letters of the alphabet ∑
  - the symbol for **Λ**
  - Parentheses **()**
  - the star operator *
  - the plus sign **+**

Regular Expressions (Continue…)

- Given $\sum = \{a,b\}$
- **a*** = {Λ, a, aa, aaa, aaa, aaaa, aaaaa, …}
- **ab*** = {a, ab, abb, abbb, abbbb, …}
- **a+b** = {a,b}
- **(ab)*** = {Λ, ab, abab, ababab, …}
- **(a+b)*** = {Λ, any string of a's and b's}

Regular Expressions (Continue…)

- The set of regular expression is defined by following rules
  - Every letter of ∑ and Λ is a regular expression
  - If r1 and r2 are regular expressions, then so are
    - (r_1)
    - $r_1 r_2$
    - $r_1 + r_2$
    - r_1^*
- Nothing else is a regular expression

Regular Expressions (Continue…)

- Whether following are RE if so what languages do they generate:

$$\sum = \{ a , b , c \}$$

- a(b + a)*
- bb(a+b)
- (a+b)(a+b)(a+b)
- (a+b)*ba
- (a+b)*a(a+b)*
- (a+b)*aa(a+b)*

Regular Expression Examples

- Example 1:
  - ∑={ a,b,c}
  - the set of all strings over this alphabet that contain exactly one b.
  - (a|c)*b(a|c)*

- Write a regular expression of language having words that end with ‘ab’
  - ∑={ a,b}

Regular Expression Examples (Continue…)

- Write a regular expression that starts with an ‘a’ and ends with a ‘b’

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words containing exactly one ‘a’

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words containing all ‘a’s or ‘b’s

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words containing at least one ‘b’

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words containing at most one ‘b’

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words that starts and ends with different letters

Regular Expression Examples (Continue…)

- Write a regular expression for language of all words that contains odd number of ‘a’s

Summary

Any Questions?