Lexical and Syntax Analysis

Hui Chen

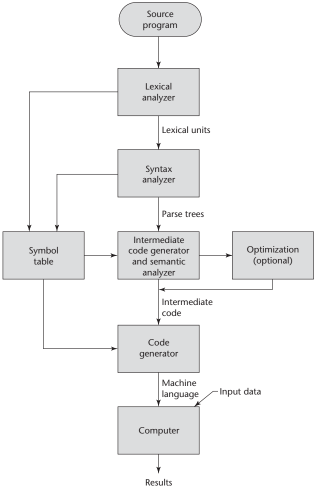
Computer Science
Virginia State University
Petersburg, Virginia

January 20, 2016

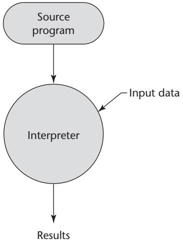
Acknowledgement

- Slides are prepared based on the textbook [Sebesta, 2012].

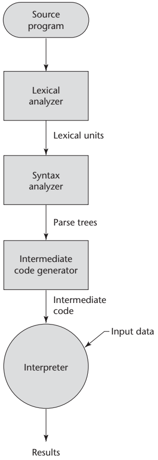
Language Implementation



(a) Compilation



(b) Pure Interpretation



(c) Hybrid Implementation

Syntax Analysis

- Consisting of two parts
  - Lexical analyzer (a finite automaton/finite state machine based on a regular grammar)
  - Syntax analyzer (a pushdown automaton based on a context-free grammar)

Lexical Analyzer

- Front-end for the parser
- Identifies *lexemes* and the tokens to which they belong
- Example: consider Java statement

```
index = 2 * count + 17;
```

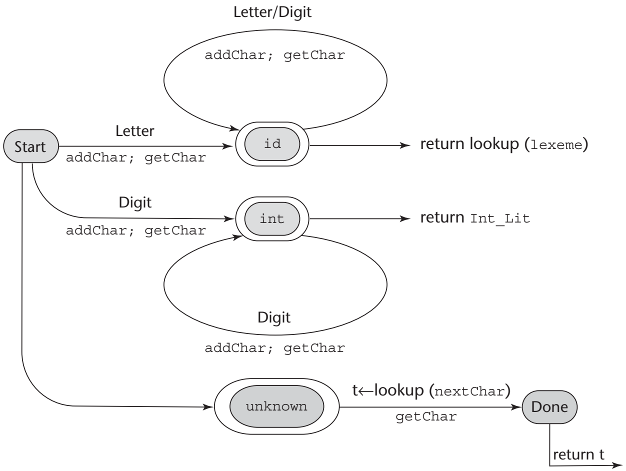
Lexeme	Token
index	identifier
=	equal_sign
2	int_literal
*	mult_op
count	identifier
+	plus_op
17	int_literal
;	semicolon

Building Lexical Analyzer

- Directly implementing the state diagram of a finite automaton from scratch
    - Design a state diagram that describes the tokens
    - write a program that implements the state diagram
- Implementing the state diagram of a finite automaton using a table-driven approach
    - Design a state diagram that describes the tokens
    - Hand-construct a table-driven implementation of the state diagram
- Implementing a finite automaton using a table-driven approach with a software tool
    - Write a formal description of the tokens
    - Use a software tool that constructs a table-driven lexical analyzer from formal description of tokens

An Example of Lexical Analyzer

- State Diagram



An Example of Lexical Analyzer

- Implementation: In Github

Obtaining Program from Github and Run Example on Linux System

```
$ git clone https://github.com/huichen-cs/sebesta.git
$ cd sebesta/lexer
$ make lexer
$ make test
```

The Parsing Problem

- To be continued ...

References I

Sebesta, R. W. (2012).
Concepts of Programming Languages.
Pearson, 10th edition.