LEXER FOR PASCAL IN RUBY

Noel Jr Aguilar Aguilar , *141031*, *141031@upslp.edu.mx*

Abstract

Lexical Analizer*—is the first phase of a compiler. It takes the modified source code from language preprocessors that has written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.*

1 INTRODUCTION

PRoces of taking an input string of character (such as the source code of a computer program) and producing a sequence of symbols called lexical tokens, or just tokens which may be handled more easily by a parser.

The lexical analyzer reads the source text and, thus, it may perform certain secondary tasks.

- Eliminate comments and white spaces in the form of blanks, tab and newline characters.
- Correlate errors messages from the compiler with the source program (eg, keep track of the number of lines), ...

The interaction with the parser is usually do by making the lexical analyzer be a sub-routine of the parser.

2 ELEMENTS OF THE ANALYZER

2.1 Tokens

Lexemes are say to be a sequence of characters (alphanumeric) in a token. There are some predefined rules for every lexeme to be identify as a valid token. These rules are definited by grammar rules, by means of a pattern. A pattern explains what can be a token, and these patterns are definited by means of regular expressions.

In programming language, keywords, constants, identifiers, strings, numbers, operators and punctuations symbols could considered as tokens.

2.2 Specification of Tokens

Let us understand how the language theory undertakes the following terms:

Alphabets

Any finite set of symbols {0,1} is a set of binary alphabets, {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} is a set of Hexadecimal alphabets, {a-z, A-Z} is a set of English language alphabets.

Strings

Any finite sequence of alphabets is called a string. Length of the string is the total number of occurrence of alphabets, e.g., the length of the string tutorialspoint is 14 and is denoted . A string having no alphabets, . a string of zero length is know as an empty string and is denote by ε (epsilon).

Specific Simbols

A typical high-level language contains the following symbols:

Arithmetic Symbols	Addition(+), Subtraction(-), Modulo(%), Multiplication(*), Division(/)
Punctuation	Comma(,), Semicolon(;), Dot(.), Arrow(->)
Assignment	=
Special Assignment	+=, /=, *=, -=
Comparison	==, !=, <, <=, >, >=
Preprocessor	#
Location Specifier	&
Logical	&, &&, |, ||, !
Shift Operator	>>, >>>, <<, <<<

Languaje

A language is considered as a finite set of strings over some finite set of alphabets. Computer languages are considered as finite sets, and mathematically set operations can be performed on them. Finite languages can be described by means of regular expressions.

2.3 Longest Match Rules

When the lexical analyzer read the source-code, it scans the code letter by letter; and when it encounters a whitespace, operator symbol, or special symbols, it de-

- *F.A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305. E-mail: author@ boulder.nist.gov.*
- *S.B. Author Jr. is with the Department of Physics, Colorado State University, Fort Collins, CO 80523. E-mail: author@colostate.edu.*
- *T.C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309. On leave from the National Research Institute for Metals, Tsukuba, Japan E-mail: author@nrim.go.jp.*

Published by the IEEE Computer Society

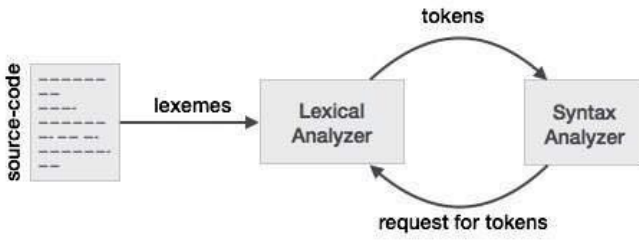
cides that a word is completed.

While scanning both lexemes till 'int', the lexical analyzer cannot determine whether it is a keyword int or the initials of identifier int value.

The Longest Match Rule states that the lexeme scanned should be determined based on the longest match among all the tokens available.

The lexical analyzer also follows rule priority where a reserved word, e.g., a keyword, of a language is given priority over user input. That is, if the lexical analyzer finds a lexeme that matches with any existing reserved word, it should generate an error.

2.4 Structure of Analyzer



3 Performance

When more than one pattern matches a lexeme, the lexical analyzer must provide additional information about the particular lexeme.
Example: pattern num matches
0 and
1. It is essential for the code generator to
know what string was actually matched.
The lexical analyzer collects information about tokens into their associated
attributes.
In practice: A token has usually only a single attribute - a pointer to the
symbol-table entry in which the information about the token is kept such as:
the lexeme, the line number on which it was first seen

3.1Lexical Analyzer vs. Parser

Lexical Analyser	Parser
Scan Input program	Perform syntax analysis
Identify Tokens	Create an abstract representation of the code
Insert tokens into Symbol Table	Update symbol table entries
It generates lexical errors	It generates a parse tree of the source code

4 Stages of Lexical Analizer

Scanner
Based on a finite state machine. If it lands on an accepting state, it takes note of the type and position of the
acceptance, and continues.
Sometimes it lands on a "dead state," which is a non-accepting state.
When the lexical analyzer lands on the dead state, it is done. The last accepting
state is the one that represent the type and length of the longest valid lexeme.
The "extra" non valid character should be "returned" to the input buffer.

Evaluator

Goes over the characters of the lexeme to produce a value. The lexeme's type combined with its value is what properly constitutes a token, which can be given to a parser. Some tokens such as parentheses do not really have values, and so the evaluator function for these can return nothing. The evaluators for integers, identifiers, and strings can be considerably more complex. Sometimes evaluators can suppress a lexeme entirely, concealing it from the parser, which is useful for whitespace and comments.

Please note that math equations might need to be reformatted from the original submission for page layout reasons. This includes the possibility that some in-line equations will be made display equations to create better flow in a paragraph. If display equations do not fit in the two-column format, they will also be reformatted. Authors are strongly encouraged to ensure that equations fit in the given column width.

References

[1] Lexical Analisys, TLC - M´ırian Halfeld-Ferrari, Compilers
[2] Compiler Design, Lexical Analisys. Tutorials Points.: SimplyEasyLearning
[3] RODRÍGUEZ Fernández, Mario (comp.). Compiladores. Santiago: Universitaria, impr. de 1995.
[4] Compiladores principis y tecnicas, Alfred, Ravy Ulmann 1990
[5] Compiladores 2da Ed. principios y tecnicas, Alfred, Ravy Ulmann 2008