Compiler I: Syntax Analysis

Thursday, March 31, 2022 6:08 PM

"Neither can embellishments of language be found without arrangement and expression of thoughts be made to shine without the light of language."
- Cicero (106-43 B.C.)

10.1 Background

Syntax analysis

- *tokenizing*
- *parsing*

10.1.1 Lexical Analysis

In the Jack language tokens *into 5 categories:*

- *keywords (class, while)*
- *symbols (+ and <)*
- *Integer Constants (17 and 314)*
- *String Constants ("Hello" and "FAQ")*
- *identifiers (labels used to name variables)*

Together this is known as the language Lexicon

lexical analysis = scanning = tokenizing

Once everything has been tokenized the compiler will read in a list of tokens rather than characters

10.1.2 Grammars

We will make a grammar (CFG) to recognize the tokens as a member of their predefined group

keywords:

'class' | 'constructor' | 'function' |

'method' |'field' |'static' |'var' |
'int' |'char' |'boolean' |'void' |
'true' |'false' |'null' |'this' |'let' |
'do' |'if' |'else' |'while' |'return' |

Symbol:
'{' |'}' |'(' |')' |'[' |']' |'.' |',' |';' |
'+' |'-' |'*' |'/' |'&' |'|' |'<' |'>' |'=' |'~' |

Integer Constant:
a decimal integer in the range 0…32767

String Constant:
' " ' a sequence of characters, not including double quote or newline

identifier:
a sequence of letters, digits, and underscore '_', not starting with a digit.

There are five qualifiers:
|: Used to say it could be either (or)
*: Used to denote any number of occurrences
?: Used to denote 0 or 1 times
(and): Used for grouping grammar elements

10.1.3 Parsing

If a derivation tree can be constructed then the sequence of characters in questions is valid within that language.

10.1.4 Parser

A parser is an agent that operates according to a given grammar.

We will compose a set of routines such as
compileStatement ->
compileStatements
compileLet
compileIf

These will be our variables within our CFG

10.2 Background

10.2.1 The Jack Language Grammar

```
'xxx'  :  Represents language tokens that appear verbatim
xxx    :  Represents names of terminal and nonterminal
elements
()     :  Used for grouping
x | y  :  Either x or y
x y    :  c is followed by y
x ?    :  x appears 0 or 1 times
x *    :  x appears 0 or more times
```