Computer Engineering Department

Fundamentals of Compiler Design

Assignment 2

Ali Sedaghi
97521378

Table of contents

Q1- Roman numerals CFG **1**

Q2- Combining lexical analyzer with parser **1**

Q3- String acceptance **2**

Q4- Lexical analyzer **2**

Q5- Language from grammars **3**

First grammar 3

Second grammar 3

Q1- Roman numerals CFG

Roman numeral symbols and their values are listed in the table below.

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Note: This grammar only works on roman numerals less than 4000.

S → thousand hundred ten digit

thousand → M | MM | MMM | λ

hundred → smallHundred | CD | D smallHundred | CM

smallHundred → C | CC | CCC | λ

ten → smallTen | XL | L smallTen | XC

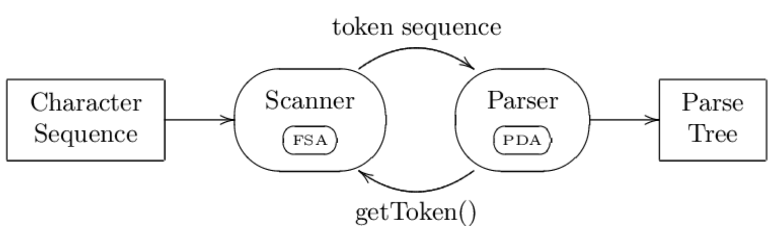
smallTen → X | XX | XXX | λ

digit → smallDigit | IV | V smallDigit | IX

smallDigit → I | II | III | λ

Q2- Combining lexical analyzer with parser[1]

Currently we are breaking it up into a pipeline of a lexer followed by a parser, executing concurrently.



But we can perform tokenization (lexical analyser) and parsing (parser) in a single step and it's called scannerless parsing or lexerless parsing. So the answer is yes. It's better to break it up into a pipeline of a lexer and parser. The cons and pros of scannerless parsing are:

+ Non-regular lexical structure is handled easily
+ Only one metalanguage is needed
+ Token classification is unneeded
+ Grammars can be compositional
- Resulting parser is more complicated
- Harder to understand and debug
- Less efficient with regard to both time and memory

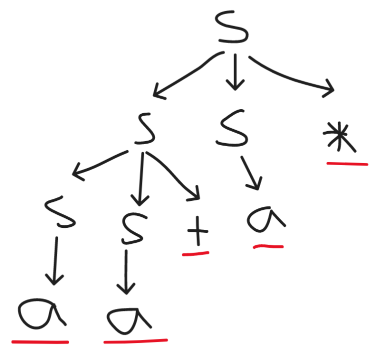
[1] https://en.wikipedia.org/wiki/Scannerless_parsing

Q3- String acceptance

S → SS+ | SS* | a

Leftmost derivative:
S → SS* → SS+S* → aS+S* → aa+S* → aa+a*

Rightmost derivative:
S → SS* → Sa* → SS+a* → Sa+a* → aa+a*



Q4- Lexical analyzer

T1: a?(b|c)*a
T2: b?(a|c)*b
T3: c?(b|a)*c
String: “bbaacabca”

Our lexical analyzer outputs the token that matches the longest possible prefix.
Matching prefixes:

- T1: “bba”
- T2: “bb”
- T3: “bbaac”

Longest matching prefix is “bbaac” which is generated by T3.

Remaining part: “abca”
Matching prefixes:

- T1: “abca”
- T2: “ab”
- T3: “abc”

Longest matching prefix is “abca” which is generated by T1.

Tokens generated by lexical analyzer: bbaac abca: T3 T1

Q5- Language from grammars

First grammar

S → 0S1 | 01
01, 0011, 000111, 00001111

$$L = \left\{0^n 1^n \ \mid \ n > 0\right\}$$

Second grammar

S → S (S) S | λ
(), ()(())(), …

$$L = \{w \ \mid \ w \ is \ a \ string \ with \ symmetrical \ parentheses\}$$