CS323 Documentation

Luc Dang

ID: 888858644

1. **Problem Statement**

The assignment is to build a lexical analyzer using an Finite State Machine that will correctly identify tokens as identifiers, numbers, separators, and operators.

2. **How to use my program**

From the terminal of a Unix operating system (preferably Ubuntu), run pre-compiled executable file "lexer" with following command:

./lexer.

If there is any problem with "*lexer*" file, compile "*lexical_analyzer_FINAL.cpp*" with following command, then execute again.

clang++ -std=c++17 lexical_analyzer_FINAL.cpp -o lexer

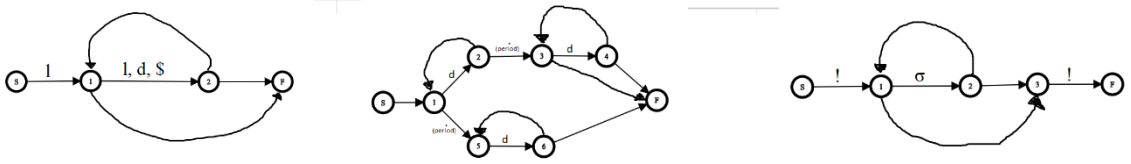
The program will prompt for input file name for lexical analysis. You must include the file extension in the name in order for the file to be read because the program will not assume .txt for reading. The program will then prompt for the desired name of your output file which the analysis will be printed to. The program will then print list of tokens and lexemes to both the output file and the terminal window.

3. **Design of my program**

The major component of my program is the DSM table which is defined as a 2D array of ints. The lexer() function changes the current state using the DSM table based on the current state and the input char. If the current state is a final state, the lexeme will be identified with its proper token type and printed.

		L	D	S	O	SP	$	!	.	\n, \t	back up
starting state	1	2	4	8	9	1	2	6	4	1	
in identifier	2	2	2	3	3	3	2	3	3	3	
end of identifier	**3**	1	1	1	1	1	1	1	1	1	y
in number	4	5	4	5	5	5	5	5	4	5	
end of number	**5**	1	1	1	1	1	1	1	1	1	y
in comment	6	6	6	6	6	6	6	7	6	6	
end of comment	**7**	1	1	1	1	1	1	1	1	1	n
separator token	**8**	1	1	1	1	1	1	1	1	1	n
operator token	**9**	1	1	1	1	1	1	1	1	1	n

RE (identifier) = L(L|D|$)*
RE (number) = D^+ | (.D^*)
RE (comment) = ! (σ)* ! ---- where σ is any character



Above are the visualized FSM for each major regular expression and its numerically represented table. Vectors were used to store the required keywords, separators, and operators for identification.

4. **Any Limitations**

   Program will stop printing after a certain amount of chars or lines read. For example, my 169 line *lexical_analyzer_FINAL.cpp* file was analyzed but only 162 lines worth of tokens were printed. The reason why is unknown.

   Also, '!', ' ', '$', '.' were not included in the separators vector and were treated as their own chart type with their own column in the DSM table.

5. **Any Shortcomings**

   - program will improperly identify "<=" as two separate operators instead of one.
   - program will ignore most escape sequences except for '\n' and '\t'. All others will be parsed and included in the token.
   - program will not differentiate between an integer and a real. Both will be identified as a "NUMBER." I was not aware that there had to be a difference and had no time to adjust DSM table and functions accordingly.