Simple MATLAB Parser using Flex and Bison

Tegala Sravani
Dept. of CSE
NIT Andhra Pradesh, India
Email: teegala.sravani@gmail.com

Kusam Bhavitha
Dept. of CSE
NIT Andhra Pradesh, India
Email: bhavithakusam@gmail.com

I. INTRODUCTION

The name MATLAB stands for matrix laboratory. MATLAB is a high-performance computing language for computation, visualization, and programming. It is an interactive system whose basic data element is an array. As most of the mathematical or computational problems can be easily solved using matrix and vector formulations thus, MATLAB is widely used in many technical programming fields. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

Applications of MATLAB are as follows:

- Deep Learning And Machine Learning
- Image And Video Processing
- Signal Processing And Communications
- Computational Finance And Computational Biology, Etc.

The objective of this report is to present a simple MATLAB parser generated using flex and bison. Flex and Bison are unix utilities that write very fast parsers for almost arbitrary file formats. They implement Look-Ahead-Left-Right parsing of non-ambiguous context-free grammars. Flex and Bison are compatible newer versions of lex and yacc. Flex is an open source program designed to automatically and quickly generate scanners whereas bison is bottom-up parser that converts a grammar description for an LALR(1) context-free grammar into a C program to parse that grammar. We have designed a MATLAB parser using the above tools that accepts a simple MATLAB code and generates valid tokens. These tokens are then verified against the grammar rules defined to find if there exists any syntax error in the MATLAB code. The remaining part of the report gives the detailed explanation of the MATLAB parser generated.

II. MATLAB PARSER

We have designed this MATLAB parser for parsing a basic MATLAB code.The following is the scope of this MATLAB parser is, it scans:

- Single Line comments
  - Ex: %This is a comment.

- Simple statements, mathematical and logical operations
  - Ex: p = 5; r = 10 + (5 * p); q = r & p;

- Array declarations and their basic operations
  - Ex: a = [1 2 3; 4 5 6; 7 8 10]; p = a.*a;

- Simple and Nested Selection Statements
  - Ex: if else, if elseif else.

- Simple and Nested iterative Statements
  - Ex: for, while, for while, for for for, etc.

- Functions

- Terminating Statements
  - Ex: break and continue are used in iterative statements
  - Ex: return is used in Functions

We have created a flex file named **matlab.l** and a bison file named **matlab.y**. The bison tool outputs a header file and a C file. The header file includes the information of the defined grammar for the MATLAB. This header file is included in the flex file as **matlab.tab.h**. The flex file reads the input from the source code using yyin pointer and invokes the flex tool. yylex() is responsible for invoking flex tool which generates tokens with the help of regular expressions defined, and returns zero at the end of the input stream. Here the tokens are identifiers, constants, operators, keywords, separators, etc. The identifiers are defined as alphanumeric or strings whereas constants are defined as combination of digits only. Once the input file scanned till the EOF then yywrap() is called by flex. By default yywrap() always return 1 if EOF is reached.

The output of the flex tool is given as input to the bison tool which verifies the grammar of the source code. For every token generated in flex file, yyparse() builds parse tree. The Bison parser detects the syntax error whenever it reads a token which violates the grammar rule defined. To handle this error, bison explicitly calls yyerror(). For a syntax error, the output of this function is a string, i.e, **syntax error**.

Different grammar rules are defined in the bison file to perform the syntax analysis on the MATLAB input code. Few of the grammar rules are as follows:

- simple statements should consist of declarations, assignments, array elements or anything similar to them. It can also be defined in '()', '{}' or '[]', in case of mathematical and logical statements.
- selection and iterative statements should be terminated with **end** command.
- **break** and **continue** statements can only be used in **for** or **while** loops whereas **return** for **functions** only.
- Every simple statement should terminate with **;** .
- **end**, **break**, **return** should not be followed by any terminating symbol whereas **continue** is followed by **;** .
- The function can have variable length of parameters.
- Nested iterative or selection statements should terminate with their corresponding **end** statements.
- Different logical and mathematical operations can be performed on arrays.

The parser reads the source code line by line and performs the grammar check. If the parser finds any violations of the grammar, it stops parsing and comes out of the parser. The violation is displayed as error in the source code and outputs the line number where **syntax error** has occurred by using the function named POS_NUMBER(). If the source code is from syntax error, it outputs **CODE SUCCESSFULLY PARSED**.

III. CONCLUSION

A MATLAB parser is generated using flex and bison tools. It parses a simple MATLAB code that is generated according to the grammar rules defined. The parser verifies the grammar of the source code and displays the position of the syntax error if any. The generated parser can detect one syntax error (if any) in one pass. Thus, this parser can be used for any simple MATLAB code to verify its syntax. This MATLAB parser can also be extended to parse some complex codes by defining more unambiguous grammer rules in the bison tool.