Parsing Tool Documentation

March 7, 2024

1 Introduction

This document provides instructions for using the parsing tool implemented with Flex and Bison. It includes compilation and execution instructions along with explanations of the command line options.

2 Tools Used

The following tools were used for implementing the parsing tool:

- Flex (The Fast Lexical Analyzer)
- Bison (GNU Parser Generator)
- g++ (GNU Compiler Collection) - Used for compiling C++ code
- Graphviz - Graph visualization software used for generating graphical representation of Abstract Syntax tree
- Visual Studio Code - An integrated development environment (IDE) used for writing, editing, and collaborating on code

3 Compilation and Execution

To compile and execute the parsing tool, follow these steps:

1. Generate the lexer using Flex:

```
$ flex lexer.l
```

2. Generate the parser using Bison:

```
$ bison -d par.y
```

3. Compile the parser and lexer files:

```
$ g++ -o parser par.tab.c lex.yy.c -lfl
```

4. Execute the parser with an input file and redirect output to a DOT file:

```
$ ./parser input.txt > graph.dot
```

5. Convert the DOT file to PDF using Graphviz:

```
$ dot -Tpdf graph.dot -o graph.pdf
```

4 Processing:

1. Input: The input for the parsing tool is a Python program provided in a file named `input.txt`.
2. Output: The output of the parsing tool is the Abstract Syntax Tree (AST) corresponding to the Python program, represented as a graph in PDF format. The graph is saved in a file named `graph.pdf`.

5 Conclusion

This document provides comprehensive instructions for using the parsing tool implemented with Flex and Bison. It includes compilation and execution steps, along with explanations of command line options.