CS335A: COMPILER DESIGN
Milestone-1

Rishabh Arijeet-210842
Shishir Gujarey-210977
Prachi Choudhary - 210732

March 2024

1 Tools Used

We have used flex for lexer,bison for parser, both written in C++ and graph visualization tool called Graphviz for AST visualisation.
Python 3.8 is used to minimize the conflicts and grammar complexities.

2 Running the files

There is a Makefile inside the src directory and the executable **ast** can simply be made by running **make** inside the src directory.

The command line options are:

- **-h** or **–help**: Prints the manual for using the **ast** executable.
- **-i** or **–input**: Takes the input file.
- **-o** or **–output**: Takes the output dot file.
- **-v** or **–verbose**: Prints the derivation(parse tree) of the input.

The default input file is **test1.py** inside the tests directory and default output file is **graph.dot**. Now, the following command generates a PDF containing the visualization of the AST.

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

A successful execution is given below:

```
$ make
$ ./ast -i testcase1.py -o tree.dot -v
$ dot - Tpdf tree.dot -o tree.pdf
```

3 Retrieving AST from Parse Tree

The parse tree obtained from the parser contains extra nodes that need not be included in the AST. This tree is pruned according to following rules :

- Replacing nodes which have a single child
- Replacing nonterminal nodes with no children