Honors Project Proposal
Fall 2024 - Spring 2025
Leon Zeltser
Adviser: Dr. Charles Thomas Wilkes
Committee Member: Dr. Matteo Cimini

Parse Tree Visualizer

For code to be compiled from a human readable form to machine code that a computer is able to run, there is a number of intermediate steps the compiler needs to take. In the first step, called lexical analysis or tokenization, the compiler reads and recognizes tokens, such as variable names, numbers, operators, function names and calls, etc.. In the second step, called parsing, the compiler builds a tree that shows the relationship between the tokens, for example recognizing that addition will be performed between two numbers if there is a + between them. There are a number of different ways which can be used to build the tree, with the two main types of algorithms being left-left (LL) or top-down, and left-right (LR) or bottom-up, with the first left referring to the direction the original code is read and converted to tokens. The former algorithm reads the token list from left to right and uses a list of rules to predict what rule a segment of code should follow based on the tokens it sees, and creates the parse tree starting from the root at the top and building down to the leaf nodes. This can be accomplished using a series of recursive functions, or functions that call themselves, or can be accomplished using a state table, which is always in a certain state and takes action based on its current state and what token it sees next. LR type algorithms on the other hand read the token list from right to left and tries to match them to a rule from a list, also using a state table. They make the tree starting from the leaf nodes then build up with each matched rule.

The program I will be making for my honors project will be a tool that shows how a parse tree is created step-by-step using the three algorithms described above. It is based off one written in the 1990s as an applet in Java, a now deprecated feature of the language. Two students previously worked to improve that program as part of their honors projects. I will be porting the tool to Python, a more modern programming language. Currently the program can just build a parse tree for a programming language called the calculator language, which is simple but is still Turing complete (i.e. anything that can be done on any computer can be done using this language) using the three algorithms described above. After implementing this in Python I hope to be able to add more functionality to the program, such as allowing the program to build a tree for several different programming languages based on some rules inputted by the user, the ability to export the parse tree as an image file, and allow users to change the font size to improve accessibility. With this project I hope to make a tool that can be used to teach about parse trees by showing how they are made and how they look like, as well showing the differences between different types of compilers and different programming languages.

The deliverables for this project will include all of the code files themselves with instructions on how to run them, the presentation, and a write up about the project where I will describe my experiences and what I learned. In my project presentation I will explain what the program does and show a demonstration.

Tentative Timeline:

Goals for the end of the fall semester:

- Find a faculty mentor and committee member.
- Write and file project proposal.
- Determine which graphical user interface library I will be using and learn how to use it.
- Create the program GUI.
- Make the program be able to draw a tree.
- Write three different parsers (LL recursive descent and table driven, LR) for the calculator language.

Goals for the end of the spring semester:

- Allow users to input any language rules and the program will build a tree based on them.
- Allow users to export the tree as an image file.
- Allow users to zoom in and out of the tree.
- Allow users to change the font size.
- Complete a write up about my project and experiences.
- Make a presentation and present the project.