Project Title: Generalized Language Abstraction and Specification System (GLASS)

Team Members:
Tommy Galletta (tgalletta2022@my.fit.edu)
Alexander Lockard (alockard2022@my.fit.edu)

Faculty Advisor/Client:
Dr. Stansifer (ryan@fit.edu)
Florida Institute of Technology, Department of Computer Science

Date of Meeting with Client for Developing Plan:
8/30/24

Goals and Motivation

Goals

- To examine and understand the inner workings of parser generators, both from a theory standpoint as well as by examining pre-existing tools.
- To build our own parser generator tool that attempts to resolve some of the issues of pre-existing tools in an elegant manner.
- To implement a GUI system and a script system around the parser generator that gives the tool additional flexibility while maintaining ease of use.

Motivation

- We hope to learn about the inner workings of parser generators, as well as the theory that goes along with it.
- We hope to develop a tool that addresses the complexities of other parser generators by having a simpler way to specify syntax and semantics.
- We hope to build a small extension for the parser generator, that being the script system, that allows for parsed code to be interpreted to a certain degree.

Approach

Grammar specification and parser generation

- Users can specify a desired grammar using a new grammar specification syntax unique to GLASS, but based on the grammar specification languages of other parser generators.
- GLASS can receive a grammar specification along with a source code file, and parse it into a parse tree.
- If the user's source files contain errors conflicting with the grammar specification they provided, GLASS will provide a series of error messages to the user to assist in debugging the source file.

Language binding utility APIs

- Users can use libraries for Python, Java, and potentially other languages to execute functions inside of GLASS.
- Users can build parse trees from grammars that can be used in their own applications.

GUI *based grammar specification tool*

- Users can build a grammar graphically through an interface.
- Users will interact with a visualization similar to a railroad/syntax diagram.
- Users can export the constructed diagram to a grammar file for use with GLASS.

Script interpretation tool

- Users can write scripts which can be executed against a parse tree generated by GLASS.
- Users can call predefined functions on the generated parse tree to manipulate the parse tree's contents.
- Users can use this script tool to write basic interpreters for their defined language grammar.

Extensive documentation

- All features within GLASS are well documented on a publicly available, actively maintained documentation website.
- Tutorial-like sections within the documentation allow users to quickly learn new features of GLASS.

Novel Features

Integration of grammar definition file and language interpretation file

- Almost all parser generator tools have both the grammar definitions for a user-defined language and the functionalities tied to those parts of the grammar contained within a singular file. Our tool aims to separate grammar definition and the language's functionalities into two separate files, while internally integrating them to perform the same operations that other parser generator tools would allow.
- The end goal of this approach is to make the input files both more readable and easier for the user to write themselves.

Algorithms and Tools

LR(1) Parsing Algorithm

- The LR(1) parsing algorithm is one of the two backbone algorithms in our project. This algorithm allows us to parse through a large number of grammars and generate a parse tree.

Parser Generation

- Critical to the function of our project, the LR(1) parser generator algorithm allows us to construct an LR(1) parse table from a series of grammar productions. These productions may be user-defined, allowing the user to generate parse tables for their own grammars.

React

- We have already used React to build our project website, and will continue using React as a means of providing our documentation in a well-organized manner for users to access via the Internet.

Researched Tools

- While not directly used in our project, research into other parser generator tools, such as ANTLR, bison, tree-sitter, lemon, JQ, and Visual BNF, has been incredibly helpful in providing insight into how we should go about developing our project.

Technical Challenges

Defining the structure of the syntax specification file

- A syntax specification language is something that all parser generators already have, however with the goal of our tool being simplicity, we will have to closely examine the pros and cons of the different specification languages used within these tools to determine which choices we should implement in our language and which we should avoid.

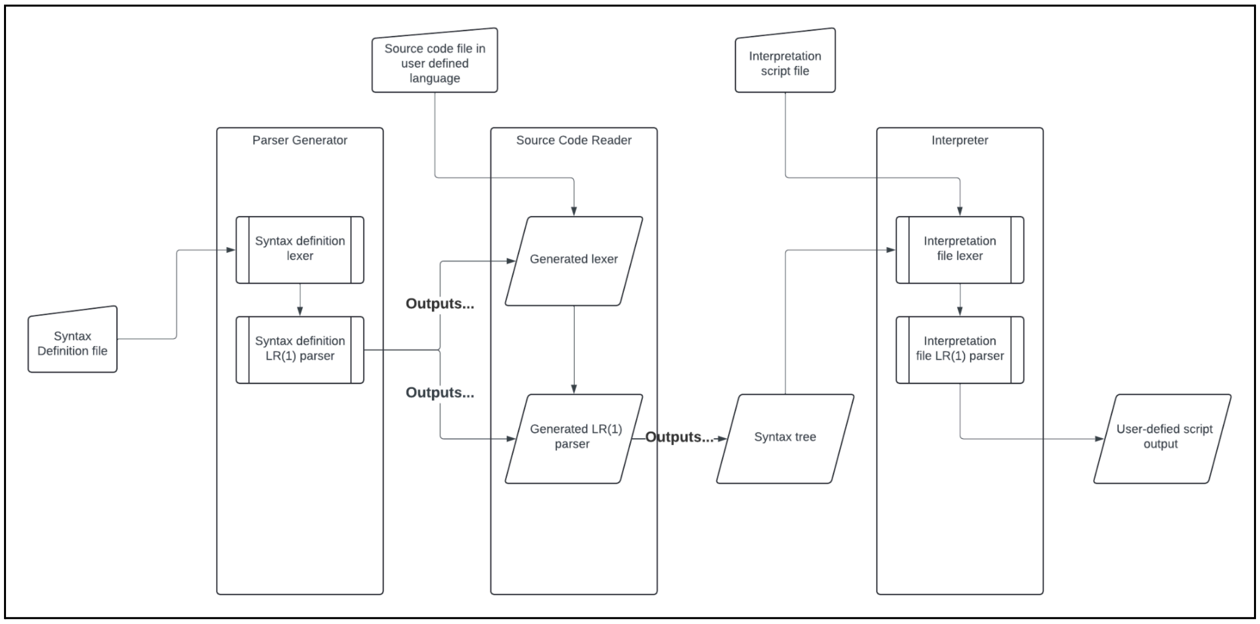
Making the parser generator

- Once we have an idea of what the structure of our syntax specification files will be, we will have to actually construct the parser generator to allow our tool to parse the specified language. This will require a dive into how parsers work, as well as the theory that goes along with it.

Designing the script interpretation tool

- While interpreting the script files will probably not be a terribly difficult task, we will have to decide which premade functions we should include in our application, and then determine how to implement them.

Design


Source code file in user defined language
Interpretation script file
Parser Generator
Source Code Reader
Interpreter
Syntax definition lexer
Generated lexer
Interpretation file lexer
Outputs...
Syntax Definition file
Syntax definition LR(1) parser
Interpretation file LR(1) parser
Outputs...
Generated LR(1) parser
Outputs...
Syntax tree
User-defied script output

Evaluation

Ease of use

- In order to gauge how long it takes for new users to begin using our tool, we will schedule times with several computer science students to "test drive" our tool.
- Each user will begin by filling out a small survey about their knowledge related to our tool (a person who has already taken formal languages is more likely to understand the concept of grammars).
- Then, the user will be asked to perform a series of predefined tasks, having full access to the documentation we provide for the tool. We will note how long it takes each user to complete each task.

User survey

- After all tasks have been completed and completion times have been recorded, there will be a short user survey asking about their experience using the tool, as well as potential changes the user would like to see.

Satisfying different use cases

- Part of our original inspiration for this project was two different use cases that seem to have a potential common solution. We will analyze if the final product is able to satisfy these use cases sufficiently, and if using the tool makes achieving the goal of these use cases easier than other approaches.

Progress Summary

Feature	Completion %	Todo
Grammar specification and parser generation	90%	Debugging and code cleaning, potential room for options and ease of use features
Language binding utility APIs	0%	Fully implement Java and Python bindings
GUI based grammar specification tool	0%	Create and integrate the GUI tool
Script interpretation tool	10%	Fully implement script interpretation
Extensive documentation	20%	Complete documentation for existing features, transfer documentation to project website

Milestone 4 (September 30th)

Documentation
- Continue writing documentation
- Host documentation on project website

GUI
- Create GUI based tool for creating syntax definitions
- Integrate GUI such that generated syntax definitions can be immediately used to parse a source code file

Debugging / Code cleaning
- Debug and clean code for pre-existing systems
- Ensure GitHub repo is well-structured

Ease of use features
- Implement features that allow for easier debugging of "broken" grammars
- Implement default values and settings that can be overridden for syntax definitions

---

Milestone 5 (October 28th)

Main system GUI
- Implement GUI for main system interactions (selecting syntax definition, source code, and interpretation script files)

Script interpretation
- Implement a system to read, parse, and process a user-defined script to be applied to a parse tree. The actions defined in the script should be executed at the appropriate time during the traversal of the parse tree.

Documentation
- Update and extend documentation on website as appropriate

Poster
- Create presentation poster

Milestone 6 (November 25th)

Evaluation
- Conduct evaluation and analyze results

Finalized Project Items
- Test/demo of the entire system
- Create user/developer manual
- Create demo video

Milestone 4 Task Matrix:

Task	Tommy	Xander
Continue writing documentation and host it on project website	75%	25%
GUI-based syntax specification tool	10%	90%
Debug / clean currently implemented systems	75%	25%
Additional ease of use features	75%	25%

Description of Milestone 4 Tasks

Continue writing documentation and host it on project website
- Currently, any documentation we have written exists in a Google Doc. The goal for this task is to further flesh out the documentation (especially the tutorial-like portions of it), and to move this documentation to our project website.
- The documentation will be structured in a way such that the sections of the documentation will be listed on the left side of the webpage, with subsections accessible via dropdown menus on a per-section basis.

GUI-based syntax specification tool

- For the purpose of creating grammar in a more interactive way, we will create a drag and drop GUI tool for defining the syntax of a language.

Debug / clean currently implemented systems

- We will ensure that the project's file structure as well as the structure of individual code files is clean, and that any known bugs are removed, as well as attempting to locate more bugs and fixing them.

Additional ease of use features

- As described in the Milestone 4 section above, features such as additional debug options and default values will be added with the goal of expediting the process of creating grammars.

Approval From Faculty Advisor

I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones.

Signature: ______________________________ Date: ________________