Compilers Portfolio

William Roberts
risause4@gmail.com
Montana State University
Bozeman, Montana

Jessica Jorgenson
jorgenson.jess@gmail.com
Montana State University
Bozeman, Montana

J. Beckett Sweeney
jbeckettsweeney@gmail.com
Montana State University
Bozeman, Montana

ACM Reference Format:
William Roberts, Jessica Jorgenson, and J. Beckett Sweeney. 2020. Compilers Portfolio. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Blank Description.

2 BACKGROUND

Blank Description.

3 METHODS AND DISCUSSION

3.1 Tools and Setup

To begin our project, we used the "Getting Started with ANTLR v4" guide provided by antlr.org.

We followed the necessary steps to set up ANTLR for Java on Windows and after working through a few minor bumps, we successfully got it working.

For version control, we have set up a GitHub repository so that the three of us can work remotely and safely have access to previous versions of our project.

As for team management, we have decided to use Trello. Its simple interface and powerful tools paired with us already being familiar with it makes it a perfect tool for our team.

Communication will take place on Discord, where we can easily send messages and files back and forth.

Both Trello and Discord have mobile apps as well so planning and communication can be done even when we are away from our computers.

3.2 Scanner

Scanners are powerful yet simple tools that constitute the first step and process in program compilation. The multi-step process of turning written languages like Java, Python, etc., into code that computers can understand begins with the Scanner, possibly making it one of the most important parts, even in its simplicity.

Scanners are token recognizers (or tokenizers, lexers). They sift through the entire files and convert every piece of language (characters, some combined) into a token. It then outputs the list of tokens. The process is quite simple, and utilizes simple regular expressions in logic. There are a few complications in this is however, first that every possible input must be defined, and second that the tokens outputted may not be legal expressions for the given language. Scanners only recognize the tokens, they do not have a understanding of the program, or output that much. That job is up to the parser, which uses the tokens as input.

ANTLR (Another Tool for Language Recognition), is the tool we used to generate the source code for the lexer-rather than coding it all ourselves. ANTLR takes in as input a .g4 file, which contains the regular expressions that define the tokens within a language, in our case Little, and creates a program that will scan source files in that language. It is important to note that ANTLR also can generate parsers, and we found it simpler to define the parser immediately, so we can more adequately test the first stages of our compiler (see Parser below).

However, a list of tokens isn't especially useful to our schooling assignment, so we used the IntelliJ IDE and wrote a simple program (in Java) to translate the list of tokens into a more readable format. We had to match this format exactly to how our Grading TA wanted it. There were several small challenges in completing this, mainly of which came down to outputting the tokens correctly.

The process of preparing our Scanner began with the example grammar file we were provided, "grammar.txt". This file contained all of the rules that we would need to implement for both the Scanner and the Parser, but they were not quite in the format we needed. The grammar file was carefully translated into the appropriate format needed for a .g4 file, which is what ANTLR uses for its Scanner and Parser grammars. We named our file "g.g4".

This process of translating took some time and was prone to error. At certain points we were confused regarding whether certain symbols were to be input directly into our g.g4 file, or if we were supposed to type their named counterpart. For example, should "(thing)" be included as "(thing)", or as "LPAREN thing RPAREN"? The answer seemed to consistently be the latter, which made this process less ambiguous as time went on.

In addition to translating the text from its given format into an acceptable .g4 format, we also needed to translate some English into regular expressions for things such as "IDENTIFIER" and "INTLITERAL". Some of these translations were simpler than others, but in the end we had them all finished.

3.3 Parser

Blank Description.

3.4 Symbol Table

Blank Description.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.5 Code Generation

Blank Description.

3.6 Full-Fledged Compiler

Blank Description.

4 CONCLUSION AND FUTURE WORK

Blank Description.