CS444/644 Assignment 1 Technical Report

Ho-Yi Fung, Jianchu Li, Zhiyuan Lin

Abstract

bleh bleh bleh

Contents

1 Introduction **3**

2 Design **4**
2.1 Scanner 4
2.2 Parser 4
2.3 Weeder 4
2.4 Abstract Syntax Tree 4

3 Implementation **5**
3.1 Scanner 5

4 Testing **6**

5 Conclusion **7**

Chapter 1

Introduction

Introduction is stored in intro.tex. [1]

Chapter 2

Design

2.1 Scanner

section for scanner design.

2.2 Parser

2.3 Weeder

2.4 Abstract Syntax Tree

Chapter 3

Implementation

3.1 Scanner

As explained in [1], a DFA can be implemented in two forms: *table-driven*, or *explicit control*. The table-driven approach, usually used by the scanner generators, utilizes an explicit transition table that could be interpreted by a universal driver program. The explicit control form, on the other hand, incorporates the transitions of the DFA directly into the control logic of the scanner program. In this project, we choose to handwrite the scanner in explicit control form for two reasons. First of all, incorporating the DFA transitions directly into control provides better performance. Secondly, the scanner produced in such manners is easier to debug and modify based on our requirements. The shortcoming of this approach is that, with the token definitions hard-coded into our program, the scanner could not be easily adapted for use elsewhere. This, however, is not a problem for the project.

Java defines a set reserved keywords that cannot be used as identifiers. In this implementation of the scanner, we read keywords in the same manners as identifiers and perform a hash table lookup afterwards to determine whether it is a reserved keyword. This approach has no affect asymptotically on the performance yet results in succinct and readable code.

Chapter 4

Testing

Chapter 5

Conclusion

Bibliography

[1] Charles N Fischer, Ronald K Cytron, and Richard J LeBlanc. *Crafting a compiler*. Addison-Wesley Publishing Company, 2009.