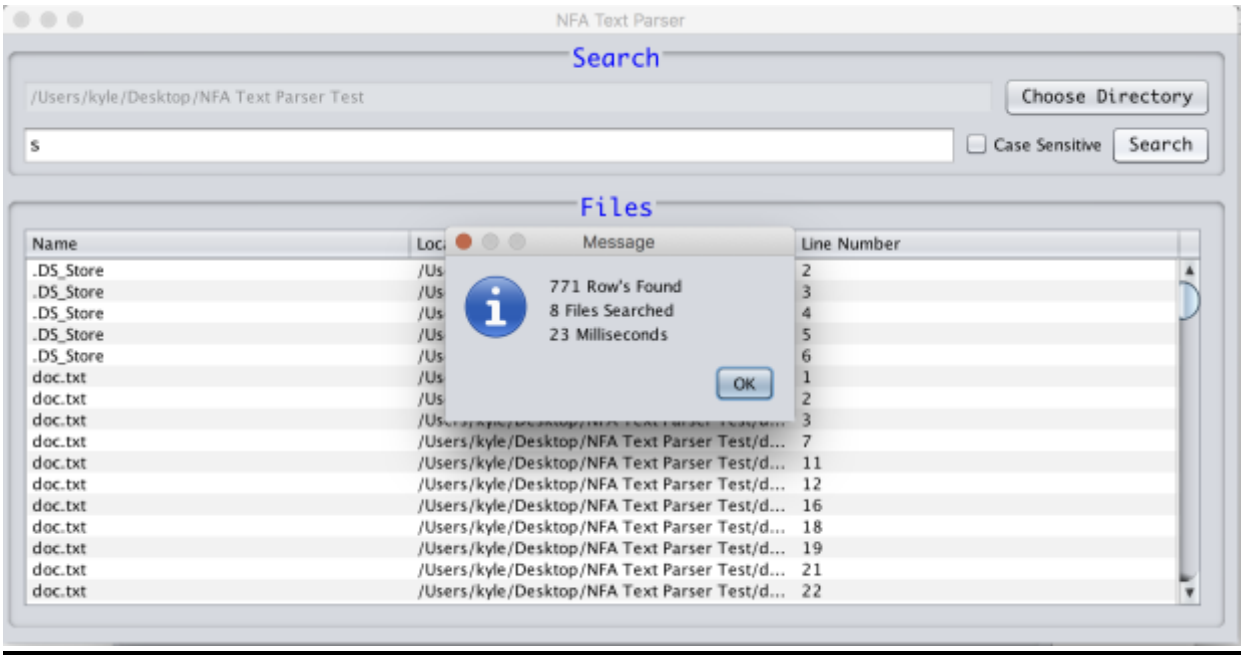
Kyle Bradshaw

Professor Rivas

Formal Languages Project

May 2nd, 2017

NFA TEXT PARSER



Kyle Bradshaw

NFA TEXT PARSER

Table of Contents

Abstract ---------- **3**

- Brief overview of the paper

Introduction ---------- **3**

- Describes the motivation of this work

Detailed System Description ---------- **3**

- Describes what the system does and how users interact with it

Requirements ---------- **4**

- Describes the physical requirement of the system resources

Literature Survey ---------- **5**

- Describes how mine is different from others

User Manual ---------- **5**

- Explains how the system is used

Conclusion ---------- **5**

- Summary of the paper

References ---------- **6**

Kyle Bradshaw

NFA TEXT PARSER

ABSTRACT

Non-Deterministic Finite Automatons are important in Computer Science because they are a main methodology in solving today's complex problems with software. In this document, I will explain why I decided to use a NFA, how the system works and the resources needed to complete this application. Lastly, I will explain how my NFA text parser is different from others that I have seen.

INTRODUCTION

The application was inspired by a program I have on my windows computer called AstroGrep. This program allows to you loop through files in a directory to search for a set of letters or characters. This is very helpful when creating big applications and you want to change a method but don't know how many times or where the method is called so you use AstroGrep to find all the instances of the value that you are looking for. Since this application is written in C# and only works on windows I wanted to create a text parser that would work on any operating system that has Java installed.

DETAILED SYSTEM DESCRIPTION

The application shows users which files and at what lines the word they are searching for appears in. The NFA will start by having the user choose a directory to look in by showing a JFileChooser as a JOptionPane. The JFileChooser will only let the user select a directory to reduce human error. Once the directory has been selected it will set a disabled text field with the path to the directory. Now that the directory has been selected the user can enter a word to search in the text box below. The user also has the option to filter the results with a check box for case sensitive and case insensitive searches. The user can start the NFA by clicking the Search button in the search panel. When the button is clicked, the program will reset the results in the JTable to clear any previous attempts and build a list of files. Next the code will call a function which will loop through the files. This loop has an if statement to check if the file

is an actual file and not another folder. If the file is a directory then by using recursion it will call the same function again with the list of files from that current file's directory. Then that file is passed to a function which will read the file line by line using the InputStreamReader class. Each line gets passed to another function where the NFA process starts. By using regular expressions and patterns we can see if the line from the file contains the keyword from the search text box. If the line has a match then we add another row to the results tables with the file name, path and the line number that we just read.

Once the program finishes looping through the files it will display another option pane which will give the user some statistics on the search. The message shows the number of lines with a match, the total number of files searched, and the time it took to run the process in milliseconds.

For another added feature, I also added a click listener to the results table, so that when the user double clicks the row it will open that file in the user's default text editor. This makes it easier for the user to open that file and make any changes they would like.

REQUIREMENTS

These are some of the requirements for the NFA application. The user must have the latest version of Java 1.8 installed on the host machine. To run the application the user can either compile and run the java file from the command line or click on the jar file to run the application. As far as the requirements for the application, the program uses multiple java imports. Some of the imports include java swing to create the interface and StreamReader to be able to read files line by line. Another import this application uses are java utilities. This is for the Patterns or regular expressions to find the matching string and the Date to tell the user how long the process took to run.

LITERATURE SURVEY

My application is different from others because of the way the code dynamically computes the NFA to find the matching word. By using regular expressions, my program can find matching words in a sequence of characters without creating a multi-dimensional array for each transition. Some DFAs that I have come across use this multi-dimensional array to hold the transition values when going from state to state. This method is known as a table-driven approach {1}.

USER MANUAL

This NFA text parser was designed to help users find a word in multiple files quickly. The user must first pick a folder to look in. After that the user enters in the word they want to search for. Once the user clicks the search button the application will start the search for the keyword in the list of files. When the NFA finds a match, it will then add that file and the line number to a table. In the table the user can click the row and it will open that file in their default text editor so they can quickly edit the file if they need to.

CONCLUSION

In conclusion, we can see why Non-Deterministic Finite Automatons are important in programming for their practicality on finding solutions for complex problems. I have also showed how my NFA works and the requirements that the user needs. Lastly, I have demonstrated how my NFA application is different from others that I have seen.

REFERENCES

1. Webber, Adam Brooks. *Formal Language: A Practical Introduction*. Wilsonville, Or.: Franklin, Beedle & Assoc., 2008. Print.