Graph Theory Research

Conor Shortt

March 2020

Contents

1 Introduction

This document contains the research I have done as part of my Graph Theory Project. Our project involves writing a program in Python to execute regular expressions on strings using an algorithm known as Thompson's construction, named after the well-known computer scientist Ken Thompson.

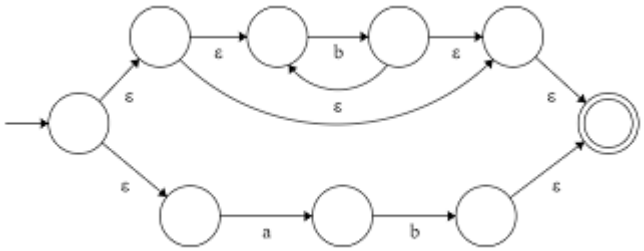

Figure 1: NFA Example for regex $a.b|b*$

1.1 GitHub Repository

· https://github.com/conorshortt123/graphtheory-project-repo

2 Thompson's Construction

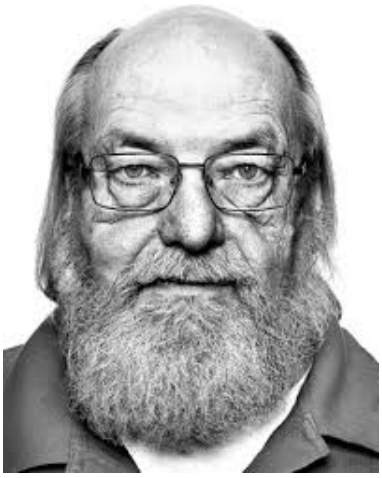

Figure 2: Ken Thompson

[5] Thompson's construction algorithm, is a way of converting a regular expression into equivalent non-deterministic finite automaton. NFA's are used to match strings against regular expressions. This algorithm is credited to Ken Thompson but can also be called the McNaughton-Yamada-Thompson algorithm.

Regex and NFA's are two representations of formal languages. (A formal language in computer science consists of words whose letters are taken from an alphabet and are well formed according to a specific set of rules.) Text processors use regular expressions to describe advanced search patterns, NFA's are better suited for execution using a computer, therefore Thompson's algorithm has a practical interest in relation to executing regular expressions.

2.1 Example NFA's

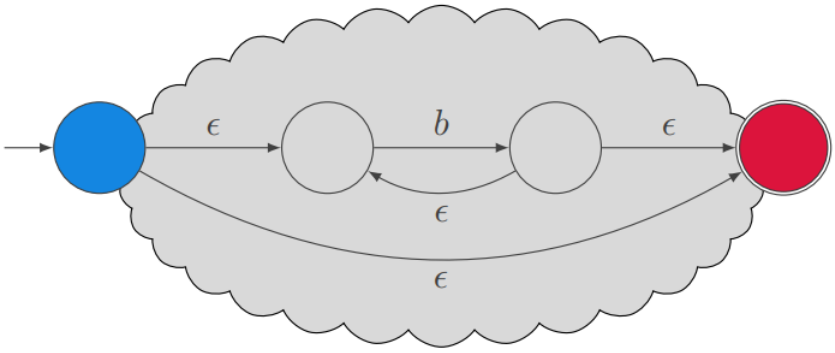

Figure 3: NFA Example for regex $b*$

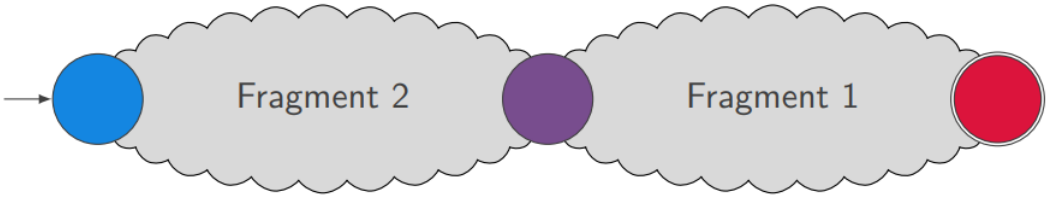

Figure 4: NFA Example for a concatenation regex

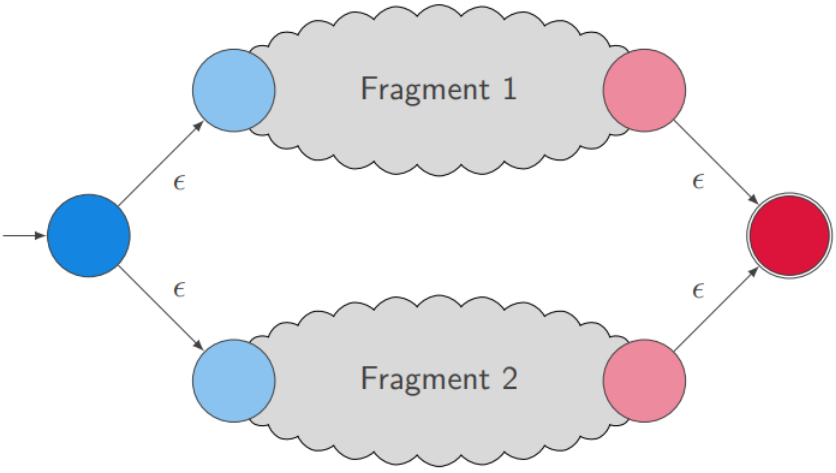

Figure 5: NFA Example for a Union regex

[4] *These are examples of NFA fragments. NFA fragments can be combined together to form larger NFA's.*

Figure 3:

As you can see in figure 3, this is an example of an NFA fragment for the Kleene star. The Kleene star is a unary operator, it is widely used for regular expressions, which is the context in which it was introduced by Stephen Kleene to characterize certain automata, where it means "zero or more".

Figure 4:

In figure 4 is an example of an NFA fragment for a concatenation regular expression. Concatenation combines fragments so the NFA can use multiple operators.

Figure 5:

In figure 5 is an example of an NFA fragment for a union regular expression. The union operator acts as an OR gate, which will accept one of two different inputs.

There are other operators such as the question mark operator, plus operator, caret operator etc. For the purposes of my particular project I included five operators ($*.|?+$). The dot operator in my project is not the standard dot operator used in regular expression engines, it is generally used to match any character bar empty strings, but I used it for concatenation.

3 Common Regular Expression Operators

Match-self Operator
This operator matches the character itself, for example operator 'A' will match 'A' and only 'A'.

Concatenation Operator
Concatenates two fragments with each other. Generally, concatenation doesn't have a special operator character, you just put b after a to match the string "ab", but in my python program concatenation is done using the special character '.'

Kleene Star Operator
This operator repeats the preceding character zero, one, or many times. Therefore this operator will accept an empty string. The regular expression $a*$ will accept strings '', 'a', 'aaaaa' and so on.

Plus Operator
This operator repeats the preceding character like the Kleene star, but only one or more times, therefore this operator will not accept the empty string.

Question Mark Operator
This operator matches zero or one of the preceding character. It matches the empty string and also whichever character it is preceding by.

Union Operator
This operator matches either of the specified fragments. Implemented with the | character. For example, the regex $a*b|c+$ will match either "abbbb" or could also match "cccc". [3]

4 Problem Statement

[4][2]*"You must write a program in the Python[1] programming language that can build a non-deterministic finite automaton (NFA) from a regular expression, and can use the NFA to check if the regular expression matches any given string of text. You must write the program from scratch and cannot use the re package from the Python standard library nor any other external library. A regular expression is a string containing a series of characters, some of which may have a special meaning. For example, the three characters ., —, and * have the special meanings concatenate, or, and Kleene star respectively. For example, the regular expression 0.1 means a 0 followed by a 1, 0—1 means a 0 or a 1, and 1* means any number of 1's. These special characters must be used in your submission. Other special characters you might consider allowing as input are brackets () which can be used for grouping, + which means at least one of, and ? which means zero or one of. You might also decide to remove the concatenation character, so that 1.0 becomes 10, with the concatenation implicit. You may initially restrict the non-special characters your program works with to 0 and 1. However, you should at least attempt to expand these to all the digits, and the characters a to z, and A to Z."*

4.1 Steps to Solve Problem Statement

- Take in two **inputs**, the **regular expression** and the **string** the user wishes to match against the regex
- Use the **shunting yard algorithm** to convert the regular expression from **infix** to **postfix**
- Create **NFA fragments** by popping characters off the postfix stack until you are left with a **singular NFA**.
- Read in the user **input string** into the NFA and **follow all epsilon arrows**, once user string is finished being read **return accept states**.
- If there are **accept states returned** then the **regular expression matches the input string.**

5 Conclusion

There are many ways to approach the problem. In my solution on my Github repository as highlighted in sub section 1.1, you will find implemented a full string match against the regular expression. In general, regular expressions are used to search for partial strings. In future I expect I will expand on my program and further add partial string matching and also add in many more operators that standard regular expression engines have to offer.

References

[1] Python Language. https://www.python.org/.

[2] Ian McLoughlin. https://github.com/ianmcloughlin.

[3] MIT Common Operators. http://web.mit.edu/gnu/doc/html/regex$_3$.*html*.

[4] Graph Theory. https://learnonline.gmit.ie/course/view.php?id=1599.

[5] Wikipedia. https://en.wikipedia.org/wiki/thompson