Graph Theory Project 2019

Student Number: G00348489

Student Name: Liam Higgins

SD\_Group: Group B

**Introduction**

The purpose of this project is to build a non-deterministic finite automaton (NFA) from a regular expression and to use the NFA to check if the regular expression matches any given string of text while using python. This document will record all research that I undergone to solve the problem.

**Infix to Postfix notation**

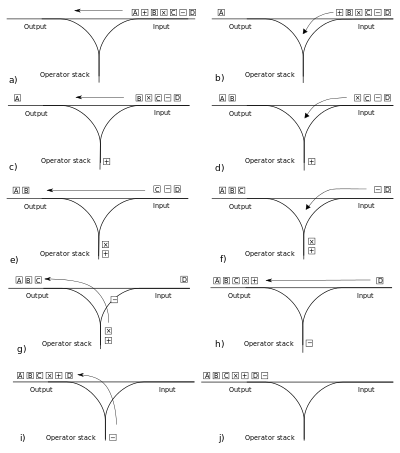
The most commonly used notation that is used is the infix notation. It is easily identified by there being an operator being present between operands that it is working on. Unlike both postfix and prefix notations, infix notations are reliant on the use of parentheses surrounding the operands to indicate the order of that the operations are meant to executed.

Postfix notation also known as Reverse Polish Notation operators follow their operands. They do not require parentheses to indicate the order of that the operations are meant to be executed in. Edsger Dijkstra invented the shunting algorithm to convert the infix notation to the postfix notation.

Example of Infic to Postfix notation: ( A+B = AB+ )

**Shunting Algorithm**

The **shunting yard** algorithm is a simple method for parsing infix expressions containing binary operators of varying precedence into the postfix notation. The order takes note of the order of precedence for each operator and assigns an operand its operator. It then pushes the operands from the stack to the output and pops any operands from the stack to the output as well.



**Thompsons Construction Algorithm**

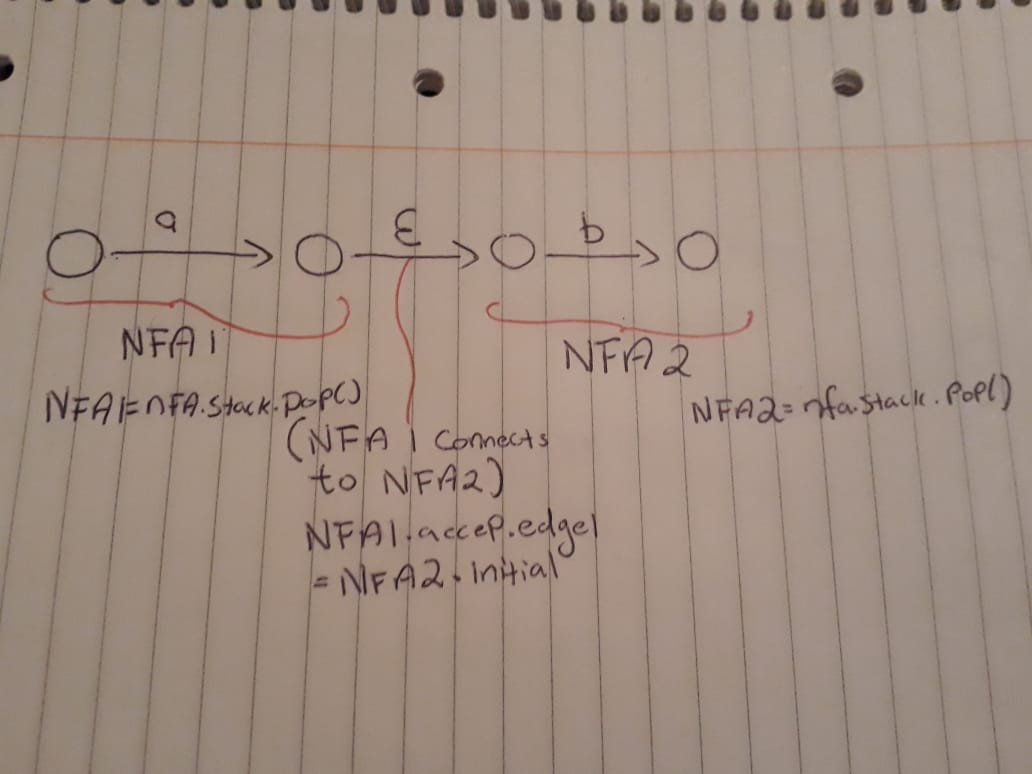
Thompsons Algorithm or also known as McNaughton-Yamada-Thompson algorithm converts a regular expression into a nondeterministic finite automaton. This NFA can be used to compare strings against regular expression and results true or false is they match.

The algorithm recursively splits an expression into constituent subexpressions and a NFA will be built using a set of rules depending on the read in operator.

**Thompsons Construction Operators**

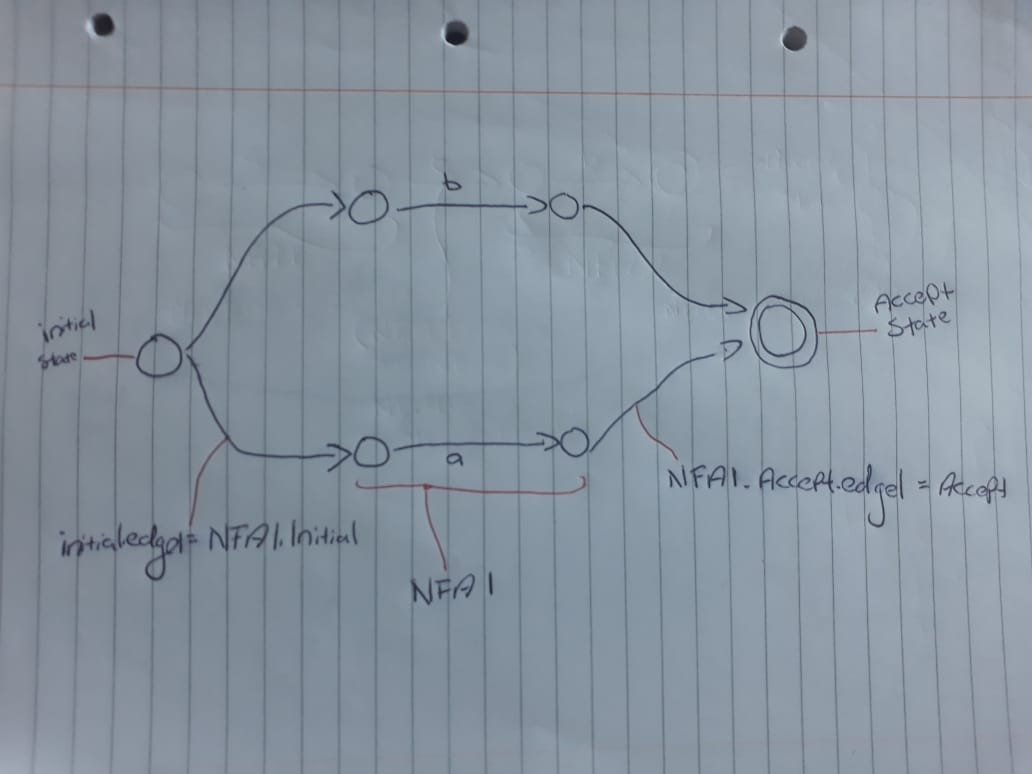
**Concatenation Operation**

The concatenation Operator or the “Dot” operator means that two operators must follow one another. For example when an infix notation is ( A.B ), for this statement to return true the string must be “ab”.



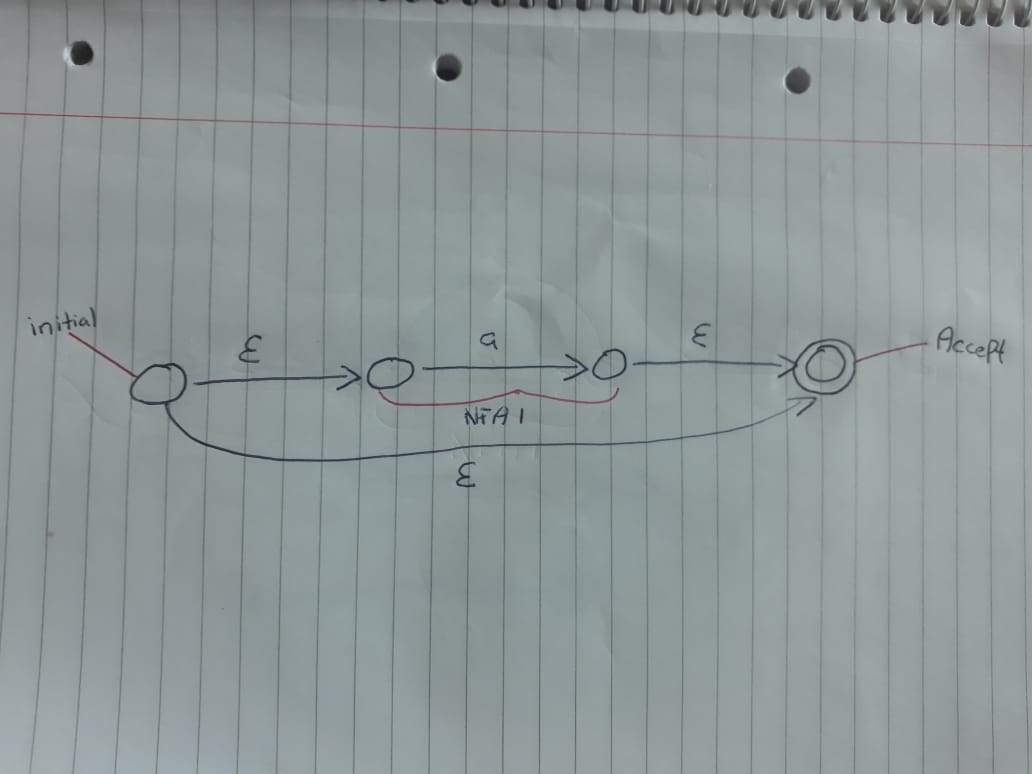
**OR Operator**

The union operator means the regular expression will return true if the string contains any of the two operands. Example ( A|B will return true if the string is an “A” or a “B” ).



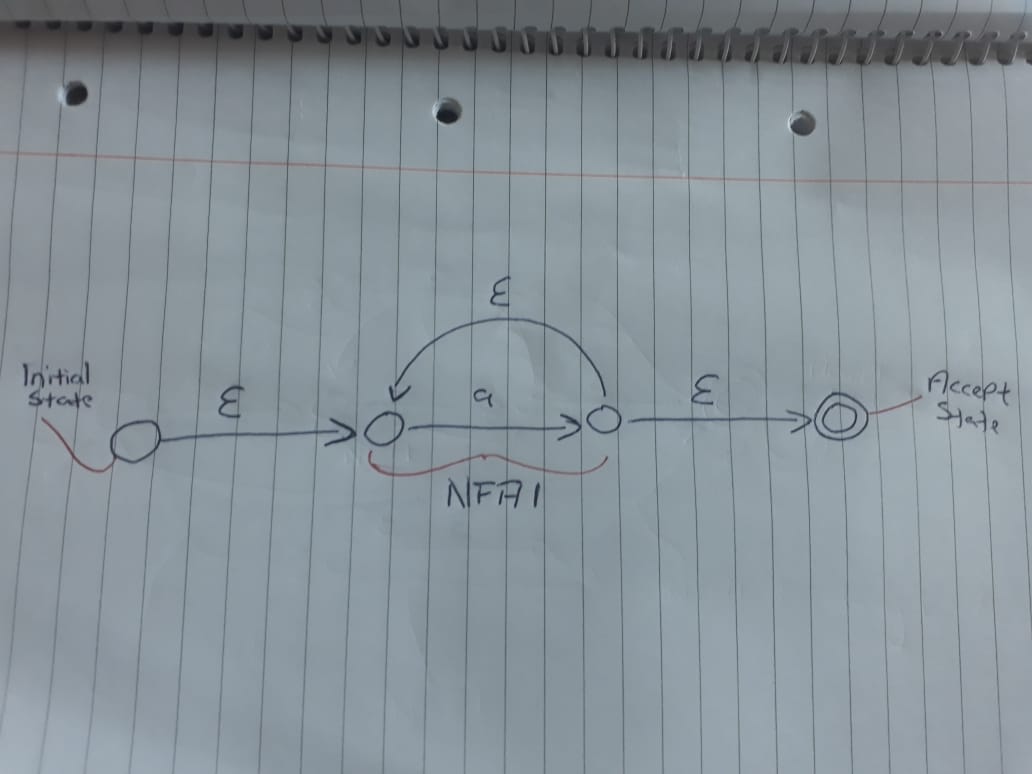
**Kleene Star Operation**

The Kleene star operator means any number of the operand. Example( a\* will return true if the string is “ ” or “aaa” …etc)



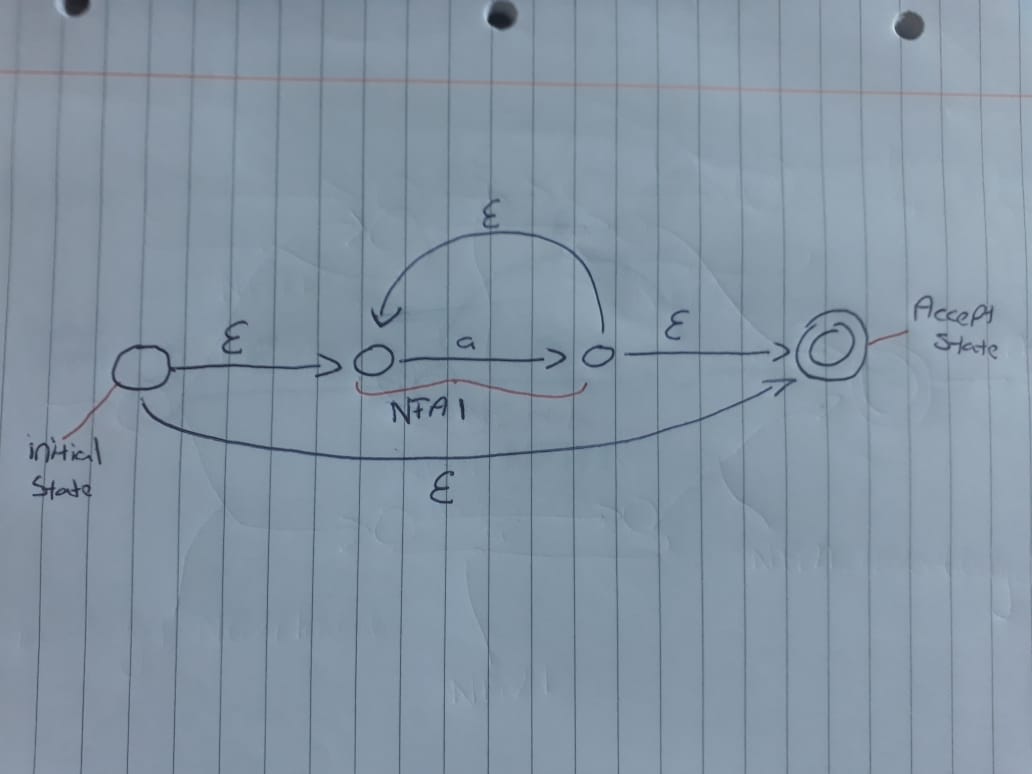
**Union Operator**

The union operator means that the string must contain at least 0 or 1 of the operands. Example( +A means for the expression to return true the string must contain “ “ or “a” )



**Optional Operator**

The optional operator matches a 0 or a 1, this effectively makes it optional. Example( !A will only return true if the string is “ “ or “a”.



**Things I learned from this project**

* I learned how to break a big complex problem into smaller easier to solve tasks.
* How to use the available resources to me to research more of a topic that I originally didn’t have much knowledge of, but now am more confident to work with in the future.
* I learned how to code more efficiently in Python.
* I learned how the shunting algorithm and the Thompsons Construction algorithm are used when making a nondeterministic finite automaton.
* I learned the difference between an infix and postfix notions and how to convert an infix notation to a postfix notation through pushing and popping from the stack.
* I learned how to solve a problem by first drawing it out before coding it up first

**References and Links I used in this project**

https://en.wikipedia.org/wiki/Infix\_notation

https://en.wikipedia.org/wiki/Reverse\_Polish\_notation

https://en.wikipedia.org/wiki/Shunting-yard\_algorithm

<https://brilliant.org/wiki/shunting-yard-algorithm/>

http://www.fon.hum.uva.nl/praat/manual/Regular\_expressions\_1\_\_Special\_characters.html

http://www.jflap.org/tutorial/regular/index.html

https://regexr.com/

https://en.wikipedia.org/wiki/Regular\_expression

http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1\_chap09.html#tag\_09\_04\_08