Visual Nondeterministic-Finite-Automaton Builder And Simulator

Introduction

Programming at its core is finding a way to communicate between people and computers. This communication is difficult as people work with words made up of letters and computers at the lowest lever have no understanding of them. Programming languages work as a way for communication to occur. However for any communication to happen a programming language must be translated into a form that the computer understands. Compilers are the programs that do this, for a compiler to understand what has been written in a program it needs to make sure what has been written is correct and in a language it understands. A nondeterministic finite automaton (NFA) is a simple way for a computer with low memory to analyze an input and determine if it is in a language that the compiler will accept.

A nondeterministic finite automaton (NFA) is a computational model that is useful to visualise what computers can do with a limited amount of memory. A NFA is a recognizer; an input string is taken and the NFA either accepts or denies this input. It can be visually represented by a sort of flow-chart or graph, which has a starting point and one or more end-points. The input is read and each character of the input is checked to see if it can move through the graph from one point to another from the starting-point. The difference between a nondeterministic and deterministic automaton is that in a NFA there can be more than one valid choice of the 'next' point from any given point in the graph. What determines the allowed 'next' point depends on an input, which is read exactly once.

A formal definition of a NFA states that it is a 5-tuple $(Q, \Sigma, \delta, q0, F)$, where

1. Q is a finite set of states,

2. Σ is a finite alphabet,

3. $\delta : Q \times \Sigma\varepsilon \longrightarrow P(Q)$ is the transition function,

4. $q0 \in Q$ is the start state, and

5. $F \subseteq Q$ is the set of accept states.

NFA are useful as an education tool as these automata provide a fundamental look at text-processing which lies at the base of a large area of computing. They are not only useful in Computing but other subject areas like mathematics, linguistics and logic.

In this project I plan to create an application that allows users to build and simulate Nondeterministic-Finite-Automaton for education purposes. I plan to design and create a program allowing people with some prior knowledge in lexical analysis to build and simulate an NFA to gain a better understanding of NFA's and how an input is passed through them. This project should provide users with a visual representation of valid inputs or regular expressions to better help them understand a NFA and the subsequent machine and language. This will be achieved using a graphical interface. This application will allow the user to define the set of symbols which the NFA accepts using a simple text interface, create 'state' nodes, define 'transitions' between state nodes and show when a state node 'accepts' the input.

Related Work

Across the internet there are quite a few tools that allow users to create automata to test languages. While researching for this project I looked at a few of these websites to get an idea of what tools exist

already, the websites I looked at can be found in the references of this document. Many of these tools allow for users to visually see how a language is accepted by a Nondeterministic-Finite-Automaton. This being the case I however have found that many of these tools are presented with missing features that would allow for a full understanding of what NFAs are used for. These tools allow for the user to build an automaton either graphically or from a list of given example languages. However no program that I found generated a complete NFA graphical diagram from a user imputed language of a regular expression.

What makes my project different from the other tools out there is that ideally I plan to allow users to build a graphical representation of an NFA from Regular expressions or regular languages, some preset languages and also to build graphically. I believe allowing users to construct a NFA from a regular expression to be a very key and useful feature as regular expressions are a core concept in building knowledge with compilers and the tool that I am building will allow for clear graphical from these regular expressions for user learning and understanding.

Requirements Analysis

This project is targeted mainly towards students who are studying compilers or more specifically regular expressions. However it may all prove useful for any other that have a small amount of required background knowledge but may not have full understanding of NFAs. This tool aims to help users acquire a clear understanding of how NFAs work as well as gain an insight into some basic knowledge in compilers around regular expressions and languages. Many people find that Nondeterministic-Finite-Automaton are difficult to visualise without a prebuilt graph due to them evolving an empty string as an input instead of or in addition to the reset of the symbols of the input language.

As a primary objective for the project to meet the needs of the targeted group I plan to write and create a tool that allows for users to select a language from a few given examples or create one of their own, then with the language they have chosen or writen take an input string from the user as an input against the language and inform the user if it passes or fails. To further meet the needs of the target group a graphical interface to clearly show the user a visual representation of the Nondeterministic-Finite-Automaton created from the language given will be needed from my tool.

An ideal system would also allow for the construction and representation of a NFA from any applicable starting input. Be that from a regular expression, built graphically by constructing state nodes and drawing transitions between them or from a set of premade NFAs ready for the user to gain understanding in how they work. Ideally it would also allow for saving of previous languages and using an input for more than one NFA at the same time for comparison.

For the tool I plan to build it will include the ability for the user to define the set of symbols of which the NFA will accept using a text interface. It will allow the user to be able to create nodes that each have a state either being a start node, accept or end node or neither a start or end node. Define transitions between the state nodes based on the language the user wants to test the input against. Return where the NFA accepts the input language of the user or not.

Further to these basic features the tool I will build will also allow for the user to be able to design and save certain NFAs as well as being able to watch a step by step traversal of the NFA on a given input string, achieved by giving a visual indicator of the current state node as the NFA checks through an input string.

Within the time allocated to this project I expect to achieve the majority of the goals I have laid out above. I expect to create a tool that gives the ability for the user to be able to create nodes, transitions and define the nodes therefore constructing a model of a NFA. I also expect the tool I create to allow the user to step through a NFA when given an input language from a text box, producing a text based

description of the input passing through the language showing what state the NFA is in for each step of the input string. Finally the ability to create a NFA from a regular expression is a feature I expect to implement.

For This project as ideally the tool will be a web based graphical model of a NFA I plan to learn how to use React Node GUI as it will allow me to pick up a new skill. React Node GUI will be suitable for the project as it is a framework that can be used to build an app in javascript that allows for the application to be cross platform and work for all kinds of operating systems. It also takes up low CPU and memory allowing the app to be accessible to as many different users as possible on all different types of devices.

Features that I would ideally include but may prove unachievable would be the feature for different types of construction of the NFA. Either by building graphically with building blocks, from a regular expression or from a preset language. Providing a visual indicator on the graphical representation of the NFA also may prove unachievable as I am not well versed with GUI and I see this as not a key feature if a text based description will already be provided.

Professional and Ethical Considerations

In this project there will be no involvement of any human participants with or without their knowledge or consent at the time. I have before thought about getting users to test the program that I will construct to see if the tool is fit for purpose and is useful as an education tool. However due to the time frame and other constraints of this project I found it unlikely that user testing would be needed and therefore not needed to complete a functional project. However if the project goes well and the time is found I may come back to reassess the need for testing for the educational tool.

During this project I will have no access to any personal information or data that allows me to identify individuals or confidential corporate or company information. The tool will be built from the ground up by myself and no outside information is needed or relevant to creating the tool so no information can or will be used in a harmful manner. Even if any testing was to take place no information of any of the users would be asked for only details of their opinions of the tool.

The research project Visual Nondeterministic-Finite-Automaton Builder And Simulator does not present any risk toward the environment or society. This project is to build an educational tool for understanding and getting visual representation of anf NFA, therefore no risk is present and no ethical issues are raised by this project.

Project Management

For this project I have already done much background reading and research into Automata and compilers in general. I have studied from the books Compilers: Principles, Techniques, and Tools as well as Introduction to the Theory of Computation. These books have allowed me to increase my knowledge and understanding of the subject area surrounding my project so that I will be able to create a suitable educational tool. As well as this I have created a github page to track my progress as well as to store all of my files in a secure backed up location so nothing is lost. I have also started to construct a basic pseudo code for the project so that when it comes to constructing the tool it goes smoothly.

Across the next term I would like to have made considerable progress or have completed the code base for the project. Before making any sort of graphic interface for the tool it is very important to have working code that can work as a base for building the visuals of the tool. So by the start of the next term I would like for the basic code to be set up for building NFAs in a graph-like format.

As I plan to use React Node GUI to build the graphical base of my project I would also like to have done research and learning towards using this as it will be a new tool that I will add to my arsenal.

Each task that I have set out for myself to complete this project are all interdependent. As the tasks for completing the project work together as building blocks and one task cannot be started until the previous one has been completed.

References

Aho, Alfred V., editor. *Compilers: Principles, Techniques, and Tools*. 2. ed, Pearson new intern. ed, Pearson, 2014.

Sipser, Michael. *Introduction to the Theory of Computation*. 3rd Ed, Course Technology Cengage Learning, 2012.

Dickerson, K., 2021. *Automaton Simulator*. [online] Automatonsimulator.com. Available at: <https://automatonsimulator.com/> [Accessed 10 November 2021].

Madebyevan.com. 2021. *Finite State Machine Designer - by Evan Wallace*. [online] Available at: <http://madebyevan.com/fsm/> [Accessed 10 November 2021].