Visual Nondeterministic -Finite-Automaton Builder And Simulator

Christy Dunlop **CandNo: 215875**

BSc Computer Science **Informatics**

Supervisor **Dr Niel De Beaudrap**

2022

Statement of Originality

This report is submitted as part requirement for the degree of Computer Science at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years

Signed:

Introduction

Compilers And Their Importance In Computer Science

Programming at the base level is to provide a computer with coded instructions for an automatic performance of a task. Computer Science students and all other programmers alike are constantly learning and improving on finding ways to communicate between themselves and the computer. Communication between computers and humans has always been a challenge as there is a large language barrier between the two. Computers at the lowest level store and manipulate information using 1s and 0s or binary numbers while humans rely on letters and words for communication. Compilers are programs that map source programs into semantically equivalent programs allowing for translation to happen between two different languages. Through the use of Compilers programs can be translated to languages more suited for computers to understand and execute, through one or many translations using compilers a program that a computer can understand and run is produced. Within compilers an input is analysed, code is generated and optimised into target machine code using symbol tables. For compilers 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. Finite Automata are used within compilers as a simple way for a computer with low memory to analyse an input and determine if it is in a language that the compiler will accept.

Non-Deterministic Finite State Automaton (NFA)

A nondeterministic finite automaton (NFA) is a computational model that when drawn out in a graph structure is useful to visualise what computers can do with a limited amount of memory. You can think of them as simple flow diagrams that have conditions to advance through with a clear start point and end point. 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 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 consists of:

1. A finite set of states S.

2. A set of input symbols Σ, the input alphabet. We assume that ε, which stands for the empty string, is never a member of Σ.

3. A transition function that gives, for each state, and for each symbol in $\Sigma \cup \{\varepsilon\}$ a set of next states.

4. A state s_0 from S that is distinguished as the start state (or initial state).

5. A set of states F, a subset of S, that is distinguished as the accepting states (or final states).

Why Is It Useful To Have An Understanding of NFA

Real computers are complex machines that are often overwhelming to the uninformed. The mathematical concepts that allow computers to operate are unmanageable for human “computers”. Therefore ideal computers are used as a computation model to help understand the theory behind how some computers may operate, a NFA is such a computation model.
Having a clear understanding of NFA is key for all programmers, beginners and experts alike. By gaining a knowledge of different automata and their underlying theories the study of computation becomes simpler. Automatons help with the theory of complexity and computation giving a simpler model for computer scientists to understand and introduce concepts relevant to other non theoretical areas of computing. They provide a fundamental look at text-processing which lies at the base of a large area of computing such as uses within programming languages as well as areas of artificial intelligence.
Without non determinism automata are simpler to wrap your head around, having determinism with every step of a computation there is a clear next step. When a machine is in a certain state and given an input we alway know what the next state will be. With non determinism several choices may be available for a state with the same input, this is sometimes a difficult concept to either understand or visualise.

The Simulation Application

This report serves as a purpose to discuss the process of making a simulator to simulate nondeterministic finite automata and the reasons behind the simulation's construction. The purpose of such a simulator is for users to educate themselves with the workings of NFA and to help visualise how an input will pass through such a automata. With a basic knowledge of computation knowing about lexical analysis, which is the first phase of a compiler where it reads an input and regular expressions, which are the languages accepted by the automata.

Who Is The Target of The Simulator

The simulator hopes to provide a clarification and understanding of NFA to its users. As automata theory is a concept closely linked with computing it is thought that the users of the simulator will be

people with an interest in computational theory. Prior knowledge will be required to use the simulator as it helps as a tool to improve understanding and help visualise NFA, therefore it is targeted to be used by first year computer science students or any other software engineers with a similar knowledge level to them.

Why Is Visual Learning So Important

“our experience in the world involves constant multisensory stimulation. For instance, visual and auditory information are integrated in performing many tasks that involve localizing and tracking moving objects. “ (Shams and Seitz, 2008)

“multisensory-training protocols can better approximate natural settings and are more effective for learning.” (Shams and Seitz, 2008)

From the journal sourced above it is clear that multisensory simulation is key in people taking and developing an understanding of new knowledge. Automata and NFA is a topic area within computing that serves as a base for further understanding within the topic and therefore having a clear understanding of their workings is vital for all aspiring software engineers or people working with computation. In particular NFA as a lexical analyzer are somewhat unique and difficult to comprehend without the use of visual stimuli or examples. Finite automata are basically graphs and trying to understand how they are used in computation with only a definition and regular expression is almost impossible, therefore the visual aspect of a simulator with clear examples is paramount to develop one's understanding of them.

Professional Considerations

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.

Academic Relevance

Throughout my study of computer science at sussex university I have come face to face with many different computation models. After being introduced to many new concepts at the start of my university life such as mathematical concepts needed for an informatics degree and an introduction to programming concepts and techniques, the regular expressions and finite state automata learned from these modules allowed me to more easily pick up further learning within the degree. I plan to use knowledge I gained through my degree in order to construct this simulator. Using the object oriented programming skills I gained from my early university modules and the knowledge from Compilers & Computer Architecture and Natural Language Engineering will be key to allow myself to have the knowledge and skills to implement an education tool of simulating NFA.

Related Works

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. By looking through and using these simulators I was able to get further ideas towards what is uitable to implement with a simulator. After using various different web based automata simulators I came to understand what key features are required for such simulators to be effective as an educational tool to assist the learning of the user.This being the case I however have found that many of these tools are presented with missing features that would allow for the user to develop a full understanding of how NFA work and what they can be used for.

Development

Overview

The objective of this project is to write a program that allows the users to visually see the workings of an automaton and how an input is passed through it. The primary aim of the simulator is education, meaning the complexity of the simulator should be such that anyone familiar with computers or simple programs. A graphical user interface will be used so that users can clearly operate the simulator with little to no programming knowledge required to operate it.

Description Of The Simulator

The simulator contains many methods for it to operate and allow for classification of inputs. A graph data structure is used to store the information about the different automata that are used in the simulation, this is partially useful as graphs share many similarities with automata and therefore can clearly simulate how automata work. The nodes of an automaton can be represented by the vertices of a graph and how to transfer between them as directed edges as the paths between them with a character key required to do so. The simulator marks the initial and final nodes that are required for a full traversal of an automaton to take place. The simulator takes a string input and one character at a time travels through the graph using these characters as the keys to move forward.

How Mine Hopes To Vary From Others

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. Regular expressions are expressions that use regular operations to describe a given language. Having this capability in my simulator will be useful as regular expressions are useful for many areas of computer science, with any application that uses text searching for strings that match a certain pattern can be very useful regular expressions provide a method for accomplishing this.

Requirement 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.

Functional Requirements

For the Simulator to be a successful tool for people to learn from there are many functional requirements that need to be implemented for the simulator to have minimal functionality for the users to use.

A primary objective for the project, to meet the required needs of the targeted group, the simulator needs to be able to allows for these users to select a language from a few given examples 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.

For this simulator as ideally the tool will be a web based graphical model of a NFA the use of React Node GUI will. 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. Also allow me to pick up a new skill that will be useful across many future projects.

Return where the NFA accepts the input language of the user or not.

Non-Functional Requirements

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 pre-made NFA ready for the user to gain understanding in how they work.

Further to these basic features the tool I will build will also allow for the user to be able to design and save certain NFA 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.

Ideally users will also be able to create a language of their own to then also take an input and run it with the language to inform the user if the given input passes or fails for their own language. Further to this 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.

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.

Met Requirements

The simulator that was built met many of the requirements set out initially. As was stated within the functional requirements many if not all of these requirements were met allowing for a functional program that allowed users to simulate some NFA.

The simulator of automata that was written allowed for all the functionality of a NFA that was stated in the definition at the start of this report. States or nodes were able to be created for the automata. A function that allows for transition between these states for the input symbol given. The ability to define initial and final states.

As seen in Figure 1, A view of the completed simulator.

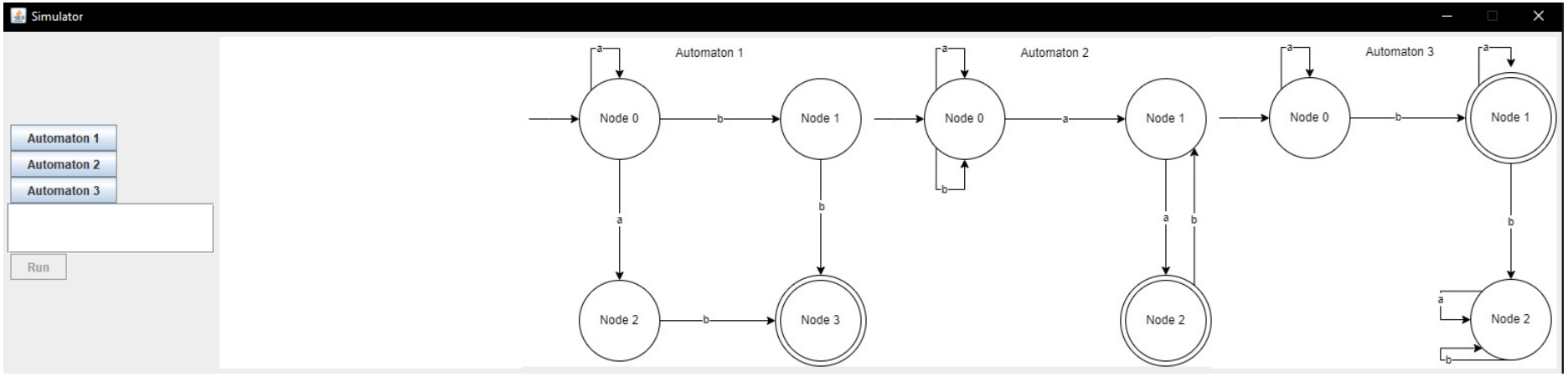
The simulator has the ability for the user to select a language for testing an input string against. Three sample languages are given to the user and three images representing the graph of the automata are shown to help the user with visualising the happening of the NFA there.

Passing an input string is also possible with the simulator. The user is able to input a string to be passed for any of the three example automata by selecting the specific automata they want and typing it into the input field and clicking the run button.

The simulator also has an output field where it clearly states whether input given by the user passes the automaton and is accepted by the language. Pass or fail it also shows the route taken by the algorithm for testing the input with the automata and can clearly show the user why an input either passes or fails.

Figure 1.

Non-Deterministic Finite State Automata Simulator



Requirements Not Met

Ideally the simulator that was built would be a web based app with GUI functionality. Unfortunately due to poor research and understanding of web based GUIs and applications this was not able to be implemented within the deadline of the simulator app. Much time was wasted trying to implement a working example of a web based GUI using React Node GUI without clear understanding of how exactly to achieve this due to the poor research made. A further attempt was made using another development platform to build web apps with UX using Vaadin, after failing to gain a proper understanding of the working of React I attempted to instead implement using this second tool. However I had left myself with too little time to learn a completely new web development app and apply it to my simulator resulting in the GUI of the simulator to just be a desktop app for users to run themselves and not a web app.

Many of the non-functional requirements were not met. Although within the code the functionality for an automaton to be constructed by the user is there it wasn't implemented graphically for the user to do this with the user interface. The code allows for automata to be made with an edge list that is a set of nodes and their keys to pass between but no user interface was implemented to allow the user to actually do this resulting in the feature not being implemented for use.

The ability to watch a step by step traversal of the NFA on a given input string using a visual indicator wasn't met. The visual aspect of the automata was made possible by using images of the pre-built automata themselves and not built graphically using the GUI, this meant that it wasn't possible to visually update the shown automata to show the step by step traversal of the NFA on a given input string. Instead a text implementation was used to show the path of the NFA as text.

Due to the constraints of the GUI that was built users were unable to create a language of their own to then also take an input and run it with the language to inform the user if the given input passes or fails for their own language. This also meant that the saving of these constructed languages was not implemented as the ability to make the languages in the first place wasn’t.

Tools Used

The builder and simulator of the nondeterministic finite automaton was written using java as a base. I chose to use java as it was the language I had the most familiarity with and believed this would help be the best fit due to this. However looking back this may have been a mistake and writing the code with javascript or another language may have allowed for the development of it as a web app to progress more smoothly and maybe actually end up as one.
Although plans were made to use web development tools such as React Node and Vaadin these never came to fruition and the learning curve for a web app to be produced for my simulator proved too much for it to be used during the time frame of the project. Therefore the decision to fall back to java swing was made and in the end the GUI of the builder and simulator was made using java swing.

Conclusion

Struggles

Throughout the project the largest struggle of the whole thing was time and self management. As someone who suffers from chronic fatigue syndrome I often found that the schedule that I had aimed to keep was unable to be so due to fluctuations with my health. Better time management and planning was needed to keep on top of the workload I set for myself.
I was too ambitious with what I would be able to learn and implement as well as accomplish for the project resulting in something produced that was less than what I had envisioned.
Better planning and research was needed to compile a more complete to brief simulator as I started the construction of the simulator before fully understanding all of the tools I would need to use. This led into struggles later own where my development slowed and some features had to be scrapped to produce a working simulator within the time limit.

Areas Of Incomplete Knowledge

As someone with limited experience with graphical user interface and web development I was too ambitious in being able to acquire the skills required to implement the simulator and builder for both at the same time for this one project. It would have been better to focus on one of these aspects, mainly the user interface allowing for a more complete simulator to be produced instead of trying to learn both at the same time and not gain mastery over either of them.

Areas for Improvement

From this project there is much I have learned about both app development as well as my own work capabilities. The number one thing that I need to improve on is my understanding of my own knowledge and what I can accomplish. The initial brief I set for myself was a bit too ambitious for my current ability and the stress of trying to accomplish all I set out initially resulted in producing a product I am not fully satisfied with.

What Went Well

The initial coding of the project went very well and I was able to produce working code that allowed for the construction of graphs to represent different automata. Then, although a struggle I then produced methods that would traverse these graphs with an input string and return a pass or fail for them, This meant I was able to produce at the base level a very simple simulator that allowed automata to be built and simulated meeting the requirements of the project set out initially.

Future Considerations And What I Learned

There are many things that I learn about making software throughout the development of this NFA builder and simulator. With the struggles I ran into later into the project I came to realise how important having a clear and effective plan is. Clear plans with complete research allows for smooth sailing through the development of software and limits the issues that could be run into later or sets up clear plans for issues if they occur. I would have benefited greatly in the project with better planning and research before I started on the construction of the simulator as the problems I ran into later slowed the progress of the project. Overall I have learn a vast deal about the development of software and the importance of planning and self management, I further believe that I am not suited for working solo on a project and believe for the future I will focus more to working with a team while trying to better myself maybe allowing for me to make something on my own in the future.

How The Simulator Functions As a Learning Aid

To conclude this report on the project Visual Nondeterministic-Finite-Automaton Builder And Simulator. I will look at the tool and see how well it functions as a learning aid for the targeted users. The simulator has many functions that are useful for the user to help gain an understanding of NFA the finished product allows for:

- Example automata reflected as a graph
- Returning whether a stack of characters passes with an automata
- Passing an input string and testing with various automata
- Giving an output of the path taken through an automata for a given input
- Giving a visual representation of various automata

As a learning tool with these features implemented a basic tool is produced with limited but useful features that allows it to be used as an introduction to NFA. However as many features were not realised for the builder and simulator it is not up to par as a learning tool for developing full understanding of NFA as there is no ability for the user to construct their own automata.

The Visual Nondeterministic-Finite-Automaton Builder And Simulator that was built for this project although functional is not practical as a learning aid, better simulators are out there that would be more suited for the user to use.

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.

Shams, L. and Seitz, A., 2008. Benefits of multisensory learning. *Trends in Cognitive Sciences*, 12(11), pp.411-417.