IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 10, Issue.1, January 2022 E-ISSN: 2347-2693

Simulation and Testing of Deterministic Finite Automata Machine

Kuldeep B. Vayadande[1*], Parth Sheth[2], Arvind Shelke[3], Vaishnavi Patil[4], Srushti Shevate[5], Chinmayee Sawakare[6]

[1,2,3,4,5,6]Dept. of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, Maharashtra, India.

**Corresponding Author: kuldeep.vayadande1@vit.edu*

DOI: https://doi.org/10.26438/ijcse/v10i1.1317 | Available online at: www.ijcseonline.org

Received: 22/Jan/2022, Accepted: 24/Jan/2022, Published: 31/Jan/2022

***Abstract*—** This article describes a JavaScript and GUI-based visualization tool for constructing, debugging, and testing DFA that can be utilized in the automata theory classroom. In automata, DFA is an important problem. What DFA is, DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. If the machine reads an input string one symbol at a time, the finite automata are termed deterministic finite automata. In DFA, there is only one path from the current state to the next state for specific input. The null move is not accepted by DFA, which means it cannot change the state without any input character. Multiple final states can be found in DFA. Like other automata visualization tools, users can edit and construct DFA by adding states and transitions and can observe transition execution by providing string input for testing. This DFA simulator allows users to construct DFA by adding states, marking any state as a final state, and also checking for string if it is valid for constructed DFA or not.

Keywords— HTML, CSS, jQuery, JavaScript, Bootstrap CSS, finite automata, visualization, simulator

I. INTRODUCTION

Many of the concepts and arguments covered in an automata theory course may be viewed and interacted with easily. The project includes drawing and simulating theoretical DFAs and showing the derivations and proving if the strings are accepted by the given DFA.

Construction type proofs, in which one representation of a language is changed to another representation, are among the proofs. In addition to the theoretical representation that is typically offered in textbooks, visualization gives pupils an alternative viewpoint [9][10]. Interaction also allows students to explore concepts and proofs while also receiving feedback. Studies in the field of algorithms demonstrate that pupils require a different visual representation with which to interact. It's usual for automata theory textbooks to start with the simplest principles and then skip through the more difficult ones.[8][11]

It becomes quite theoretical at this point. As a result, in the automata theory course, we developed a mechanism that allows us to switch the course from a lecture-only style with written exercises to a more interactive lecture format with interactive lab and homework activities. In this paper we describe a visualization tool for designing, debugging, and testing DFA which will allow users to construct DFA by adding states, marking any state as a final state, and also checking for string if it is valid for constructed DFA or not.

II. LITERATURE REVIEW

Morazán MT, Schappel JM, Mahashabde S [1] shows how to use FSM, a domain-specific language for automata theory, to create a visualization tool for constructing and debugging deterministic finite-state machines. Users can edit machines and view their operation, just like with previous automata visualization tools. The user is neither burdened nor distracted by rendering a machine as a graph, as is the case with other automata visualization tools.

Susan H. Rodger and Anna O. Bilska [2] provide a collection of new and improved tools for learning about formal languages and automata theory. The new Java-based tools include JFLAP for creating and simulating finite automata, pushdown automata, and Turing machines; Parser for parsing restricted and unrestricted grammars and converting context-free grammars to Chomsky Normal Form; and Pump Lemma for proving specific languages are not regular

.

Raffelt, Harald & Steffen, Bernhard & Tiziana, Margaria [4] describe dynamic testing, a method for routinely testing black-box systems using automata learning with absolutely no precondition. Our method analyses the system under test SUT sequentially based on interface descriptions while also extrapolating a behavioral model. This information is then used to guide the exploring process. Thanks to the applied learning strategy, our method is excellent in that the extrapolated models are the most concise in consistently conveying all of the knowledge gathered during the exploration.

Mr. Patil and Poornima Naik [5] propose a DFA parser for HQL/SOQL token processing. For the numerous HQL/SOQL tokens that we discovered, we generated a state table and state diagrams. State information is stored in a permanent database management system as a technique to improve efficiency and flexibility. Only a few commands are available in HQL/SOQL at the moment, but more will be introduced soon.

Ficara, Domenico & Procissi, Gregorio & Vitucci, Fabio & Antichi, Gianni & Pietro, Andrea [6] proposed that Delta Finite Automata (FA) is a new representation for deterministic finite automata (orthogonal to prior solutions). It drastically reduces states and transitions and only requires one transition per character, allowing for quick matching. In addition, a unique state encoding approach is created, and the entire programme is tested in the packet categorization domain.

Kuldeep Vayadande, Ritesh Pokarne, Mahalaxmi Phaldesai, Tanushri Bhuruk, Tanmai Patil, Prachi Kumar[9] Demonstrate the use of experimental computer simulations and propose large-scale extensions using novel experimental testing. We used the game to study the complexities of symbio poiesis and evo-devo, as well as the abstract hypothesis: that similarities exist at many levels, including cells, animals, natural communities, and so on, as a result of comparable interactions between both as a health-based game.

K. B. Vayadande, N. D. Karande, and S. Yadav[10] describes many strategies for identifying moving objects against a static or dynamic background. The issue with dynamic backgrounds is that if they are not static, correct foreground objects cannot be detected accurately (precision and recall). The majority of the work reviewed in this paper focuses on object detection in a static environment.

Varad Ingale, Kuldeep Vayadande, Vivek Verma, Abhishek Yeole, Sahil Zawar, Zoya Jamadar [11] this papers goal is to the operation of a lexical analyzer in the most direct way in order to provide a thorough comprehension of the lexical phase of the analyser.

Kuldeep Vayadande, Harshwardhan More, Omkar More, Shubham Mulay, Atharva Pathak [12] says that because of the widespread use of computer learning games in institutions, game development has dropped significantly. Playing computer games increases student engagement. For example, many studies have demonstrated that using versions of computer games shows improved focus, stimulates students' motivation to study, hence the Pacman game is presented. Such games have been demonstrated to be effective in the classroom through experimental research.

III. METHODOLOGY

The project consists of three major steps first being constructing DFA using tools built using JavaScript to describe transitions by creating states and assigning input for each transition altogether forming a functional DFA. The second step is to connect the transitions by connectors describing transition status regarding input value and connected state node. And the third step is to ensure and be able to test the DFA by inputting a string and getting an acknowledgment whether DFA accepts the string or not.

Building DFA

We used JavaScript to represent properties of the states and transitions like code snippet below-

```
let states = [
 {
  id: 0,
  connectedNodes: [],
  selected: false,
  final: false,
 },
];
```

We created a list of states and included properties of each node like id for naming the state node, connected Node's list to store all the connected states which will be used in step 2 to generate connecting links for the visual representation of the DFA, and selected and final flags for recognizing active state and final state.

The next task is to collect information about states and transition from user forms by assigning event listeners to the buttons provided to the user and appending it to the state's list, also adding connected states to the connected Nodes list.

Connecting the states

After having the connected Nodes list, we draw a line to represent connected nodes by generating a transparent rectangular block which will be a div in CSS; between connected states and highlight the bottom-line using the border-bottom property of CSS. This will generate the effect of connecting the two nodes and the user will be able to visualize connected states and observe transitions written in each state node, the overall result imitating simulation of DFA.

Testing DFA

This step consists of establishing logic for testing if all the transitions of DFA are valid by checking the valid input against each transition also checking with the states and then being able to test a given string for the user-generated DFA by simulating input given to each transition possible and keeping a record of the final state. If the input string passes all the checks above then output a line of text mentioning the status of acceptance of input string and if yes, mentioning final state.

A. Proposed System

There are very few tools aimed at teaching automata theory, properties, and applications, the great majority of them do not use real-time visualisation and are more analogous to the traditional text and reading approach. This

project assists in the visualisation and real-time testing of user-customizable DFA. The user can test the DFA by providing an input string. This kind of system can also be used in theory lectures to enable the students to understand the topic and interpret how DFA works with specified states and transitions, which is absolutely essential because topics like automata theory can sometimes be difficult to understand. Our project aims to address the above problem by providing a real-time DFA builder and simulator web app that will help learners in having better understanding of the concepts.

B. Flowchart/ Algorithm

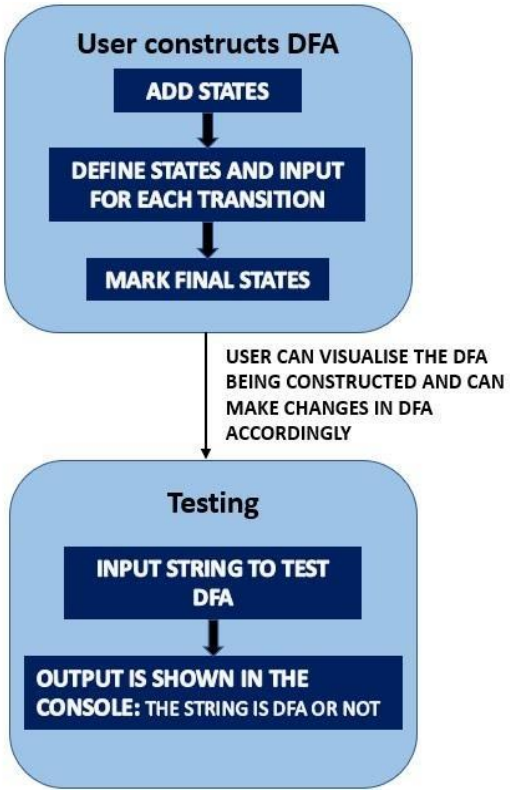

Fig 1: Flowchart

Algorithm

Step 1: - Adding new states
First, the user needs to add as many states as he wants, to construct a DFA.
Step 2: - Assign transitions between the states.
After adding states, the user needs to assign transitions between the states as per his/her wish.
Step 3: - Marking a state as a final state.
After the user has finished assigning the transitions, he/she will have to assign any state as the final state for further testing.
Step 4: - Testing a string.
At last, after finishing all the above steps, the simulator will ask the user to input a string and will check if the constructed DFA reaches the final state using the input string.

IV. Results and Discussion

On opening the simulator, it will show a basic home page (consider Fig 2).
First, the user needs to add as many states as he wants, to construct a DFA (consider Fig 3 and Fig 5).
After adding states, the user needs to assign transitions between the states as per his/her wish (consider Fig 4).
After the user has finished assigning the transitions, he/she will have to assign any state as the final state for further testing (consider Fig 5).
The simulator asks the user to input a string and will check if the constructed DFA reaches the final state using the input string (consider Fig 6).
The simulator will give the results if the given string is accepted by the constructed DFA or not (consider Fig 7).

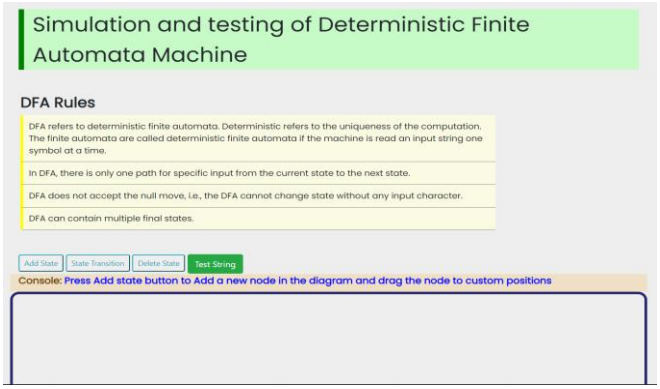

Fig 2: Basic home page.

On opening the simulator, it will show a basic home page mentioning rules for DFA, console to show state and logs, buttons to build DFA and window below to interact with it. (Consider Fig 2).

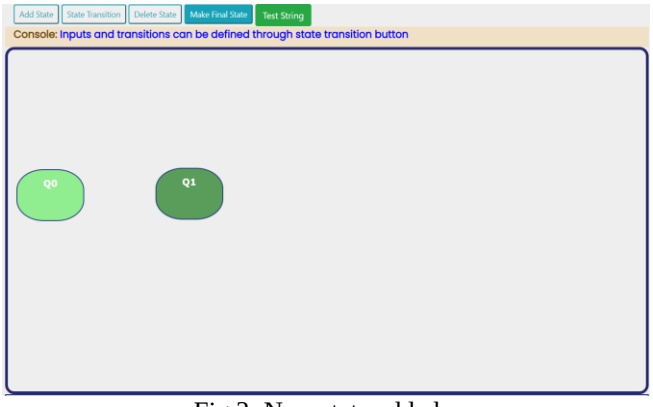

Fig 3: New state added.

There is an Add State button using which the user can add a state to the DFA he's trying to build and this is the first step to construct the DFA. Users also have access to the Delete State button in order to delete a state if required. (Consider Fig 3 and Fig 5).

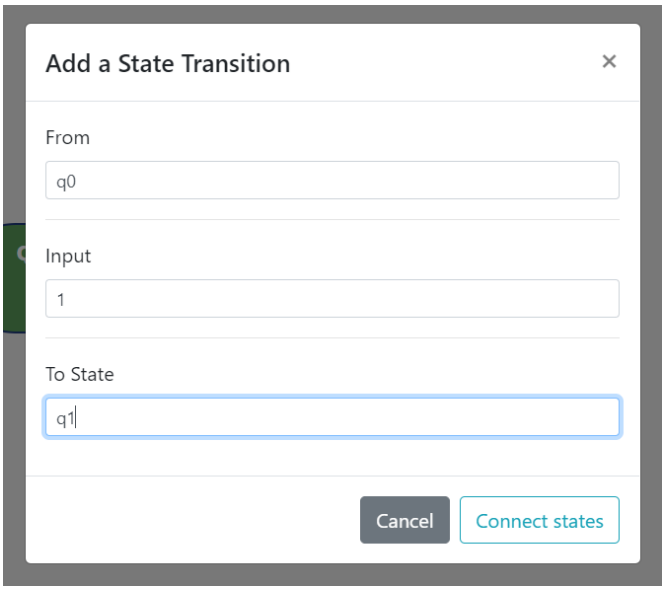

Fig 4. Assigned transition between two states.

After adding states, the user needs to assign transitions between the states and properly construct DFA following rules defined for DFA. If rules are not followed correctly, validations are put in place to notify the user about it. (Consider Fig 4).

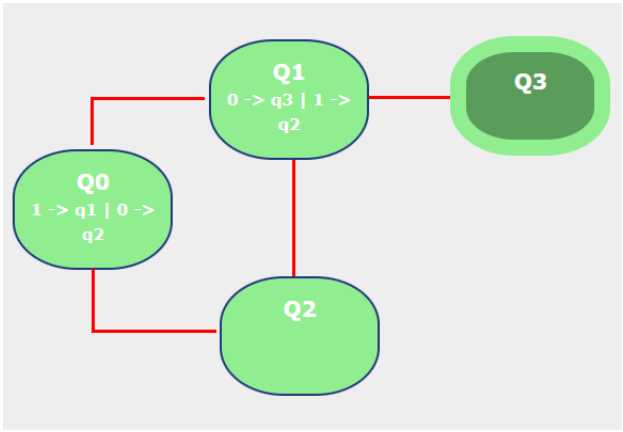

Fig 5: Assigned Q3 as the final state.

After the user has finished assigning the transitions, can proceed to mark final states as per requirements using the Mark Final State button as DFA needs at least one final state before moving on to testing. (Consider Fig 5).

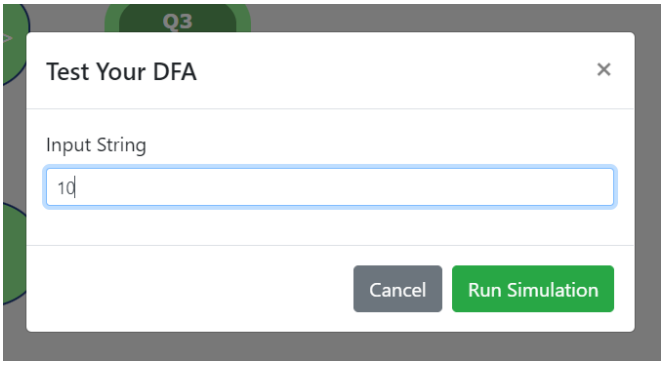

Fig 6: Giving input string to test the constructed DFA.

After completion of construction, user will be able to test the DFA by using the Test String button also providing an input a string in order to check if the DFA accepts the string and validate the result in the console (Consider Fig 6 and Fig 7).

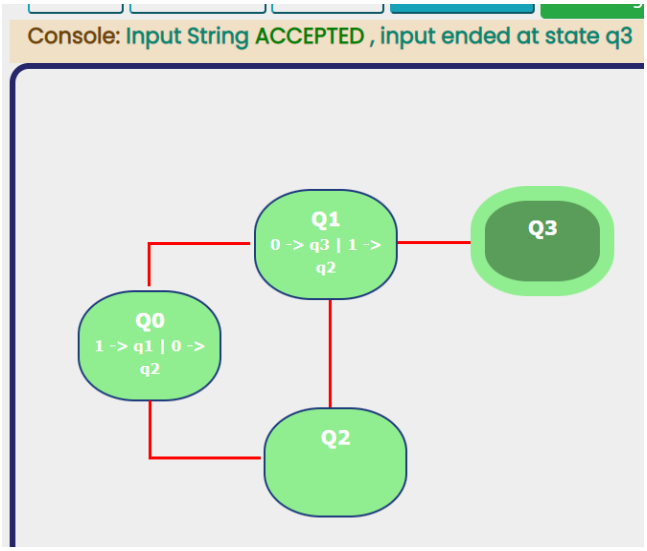

Fig 7: Given input string accepted by the DFA.

After completion of construction, the user is able to test the DFA by the use of the Test String button also providing an input of a string in order to check if the DFA accepts the string and validate the result in the console (Consider Fig 6 and Fig 7).

V. Conclusion and Future Scope

In this research, we describe a visualization tool for designing, debugging, and testing Deterministic Finite Automata (DFA) which will allow users to construct DFA by adding states, marking any state as a final state, and also checking for string if it is valid for constructed DFA or not. This project makes the study of DFA practically possible and not theoretical. But there are few limitations of this project. Those are as follows:

1. Our simulator is limited to DFA only.
2. The simulator cannot perform operations on NFA
3. This simulator is limited to only one operation of DFA

In the future, we can add an option to construct NFA. We can add a simulator to perform operations on NFA like testing a string on NFA, converting NFA epsilon to NFA. We can add a converter in which it will convert NFA to DFA or NFA epsilon to DFA.

Acknowledgment

I wish to express my sincere gratitude to Dr. Kuldeep Vayadande, Asst Professor Vishwakarma Institute of Technology for providing us an opportunity to do the course project on this topic and also thank him for supporting us till the end. We also thank the International Journal of Computer Science and Engineering for accepting our project paper and giving us an opportunity to publish it.

REFERENCES

[1] M. T. Morazán, J. M. Schappel, and S. Mahashabde, "Visual designing and debugging of deterministic finite-state machines in FSM," Electronic Proceedings in Theoretical Computer Science, **vol. 321, pp. 55–77, 2020.**

[2] S. H. Rodger, A. O. Bilska, K. H. Leider, M. Procopiuc, O. Procopiuc, J. R. Salemme, and E. Tsang, "A collection of tools for making automata theory and formal languages come alive," Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education - SIGCSE '97, **1997.**

[3] H. Raffelt, M. Merten, B. Steffen, and T. Margaria, "Dynamic testing via Automata Learning," International Journal on Software Tools for Technology Transfer, **vol. 11, no. 4, pp. 307–324, 2009.**

[4] P. G. Naik, S. G. Patil, and G. R. Naik, "Natural language interface for querying hardware and software configuration of a local area network," International Journal of Computer Sciences and Engineering, **vol. 7, no. 2, pp. 949–963, 2019.**

[5] D. Ficara, S. Giordano, G. Procissi, F. Vitucci, G. Antichi, and A. Di Pietro, "An improved DFA for fast regular expression matching," ACM SIGCOMM Computer Communication Review, **vol. 38, no. 5, pp. 29–40, 2008.**

[6] Jiwei Xue, Yonggao Li and Bo Nan, "Application research of finite automaton in distance education," 2010 4th International Conference on Distance Learning and Education, 2010, pp. 129-133, doi: 10.1109/ICDLE.2010.5606024.

[7] Raza, Mir Adil, Kuldeep Baban Vayadande, and H. D. Preetham. *"DJANGO MANAGEMENT OF MEDICAL STORE."*, International Research Journal of Modernization in Engineering Technology and Science, **Volume:02 Issue:11 November -2020**

[8] K.B. Vayadande, Nikhil D. Karande," *Automatic Detection and Correction of Software Faults: A Review Paper*", International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653, **Volume 8 Issue IV Apr 2020.**

[9] Kuldeep Vayadande, Ritesh Pokarne, Mahalaxmi Phaldesai, Tanushri Bhuruk, Tanmai Patil, Prachi Kumar, "*SIMULATION OF CONWAY'S GAME OF LIFE USING CELLULAR AUTOMATA"* International Research Journal of Engineering and Technology (IRJET), **Volume: 09 Issue: 01 | Jan 2022**, e-ISSN: 2395-0056, p-ISSN: 2395-0072

[10] K. B. Vayadande, N. D. Karande, and S. Yadav, "A review paper on detection of moving object in dynamic background," International Journal of Computer Sciences and Engineering, **vol. 6, no. 9, pp. 877–880, 2018.**

[11] Varad Ingale, Kuldeep Vayadande, Vivek Verma, Abhishek Yeole, Sahil Zawar, Zoya Jamadar. "Lexical analyzer using DFA", International Journal of Advance Research, Ideas and Innovations in Technology, www.IJARIIT.com.

[12] Kuldeep Vayadande, Harshwardhan More, Omkar More, Shubham Mulay, Atharva Pathak, Vishwam Talnikar, " Pac Man: Game Development using PDA and OOP", International Research Journal of Engineering and Technology (IRJET), **Volume: 09 Issue: 01 | Jan 2022**, e-ISSN: 2395-0056, p-ISSN: 2395-0072

[13] Rohit Gurav, Sakshi Suryawanshi, Parth Narkhede, Sankalp Patil, Sejal Hukare, Kuldeep Vayadande," Universal Turing machine simulator", International Journal of Advance Research, Ideas and Innovations in Technology, (**Volume 8, Issue 1** - V8I1-1268, ISSN: 2454-132X

AUTHORS PROFILE

Kuldeep B Vayadande completed Bachelor of Engineering in Computer Science & Engineering from Shivaji University, Kolhapur in 2009. He completed his M.Tech in Computer Science & Technology from Shivaji University in 2014. He is currently pursuing Ph.D. in Computer Science & Engineering from Career Point University, Kota, Rajasthan.

Parth Sheth is currently studying BTech in Artificial intelligence and data science at Vishwakarma Institute of Technology, Pune. He completed his schooling from Deccan Education Society, Pune. He completed his 11th and 12th from Ashok Vidyalaya, Pune.

Arvind Shelke is currently studying BTech in Artificial intelligence and data science at Vishwakarma Institute of Technology, Pune. He completed his schooling from Gurukul English school, Beed.

Vaishnavi Vishwas Patil is currently studying BTech in Artificial Intelligence and Data Science in Vishwakarma Institute of Technology, Pune. She completed her schooling till 10th from Kendriya Vidyalaya CRPF, Talegaon Dhabhade, Pune and then went to Kendriya Vidyalaya no. 2 Dehu Road ,Pune with a science field.

Srushti Shevate is currently studying BTech in Artificial intelligence and data science at Vishwakarma Institute of Technology, Pune. She completed her schooling from Mount Carmel Convent High School, Pune

Chinmayee Sawakare is currently studying BTech in Artificial Intelligence and Data Science in Vishwakarma Institute of Technology, Pune. She completed her schooling from Podar International School, Nashik

