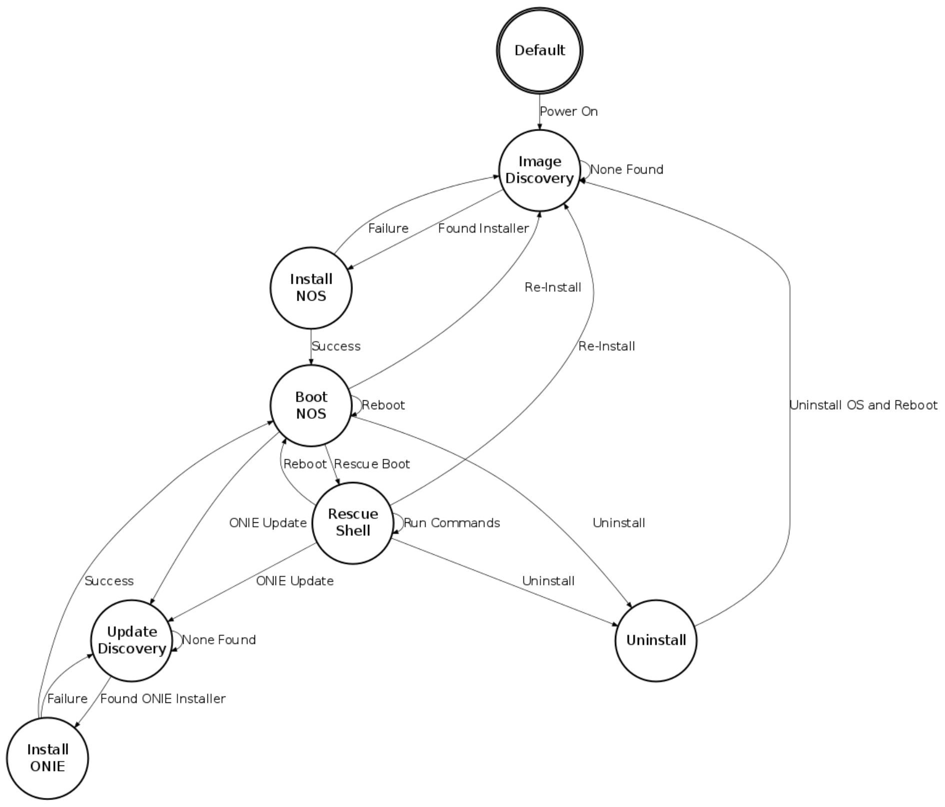
CS252A Finite State Machine Learning Module

A. Mudgal (12008), A. Bhutra (12043), A. Choudhary (12148),
C. Gupta (12218), D. Singal (12243)

November 27, 2014



Abstract

Automatas are widely used as introductory objects in Theory of Computation courses. They are best mastered with practice and experience. Students are often unclear on whether their Finite State Automatons are doing what they want them to do. This motivates us to create an extensible, independent, and user-friendly application that caters to a broad class of students, and is easy-to-use for the instructor.

1 Introduction

Our module shall work for the three types of automatas - Deterministic Finite State Automata, Non-deterministic Finite State Automata, and Push-Down Automata. Apart from the *play-with-the-ui* practice mode, the main aim of the project is to make a quiz interface for the automata. The two essential issues to be tackled are,

1. **The student** should be able to use the module for easy and efficient learning. This requires the module to be **natural** and **stable**, along with the possible availability of the module for **offline usage**.
2. **The professor** should be able to get across material and concepts to the student, through the provision of **examples** and **data**. Most importantly, there needs to be a **secure** quizzing module, that tests students in a **timed** manner, and broadly allows for most functions of the three types of automata.

As additional features, we created a module that allows for debugging of the automata. Essentially, on a given input, the entire sequence of states that the automata goes through can be seen with a very convenient U/I. If the automata is giving unexpected output for a string, this function can be used to create a correct automata.

We also plan to include **DFA Minimization**, conversion from **NFA to DFA** and vice-versa, identifying the **regular language** of a DFA/NFA (DFA/NFA $\rightarrow$ Regular expression). These are advanced features and still under construction.

2 Finite Automatas

Deterministic Finite State Automatas and Non-deterministic Finite State Automatas simulate a character string a set of states that we jump along depending on the next character in the string. There is a start state, and there are one or more accept states. Strings that reach an accept state when all their characters are scanned are *accepted* by the automata. A set of strings that an automata accepts, is called a regular set.
Push-Down Automatas have a stack along with a finite state automata. Symbols can be pushed and popped off a stack. The strecognize exactly the set of regular languages. A set of strings that a push-down automata accepts, is called a context-free language.

3 Interface

In this online automata builder, users login with a username and password into the server. They can switch between the types of finite state machines using appropriately placed buttons.

Student:
The student will be redirected on to a page that contains options to take a quiz, or go to the practice arena. On clicking any option, appropriate instructions are displayed.

Professor:
An admin interface for the professor is also available. Here, the professor can look at students' marks, begin/end/modify a quiz. A quiz is created by giving a question, and a set of test cases for the accept and reject column.

4 Implementation Details

First we go over some basic details, and then look at a few details.

- Languages used:
  - HTML
  - PHP
  - Javascript (JS)
  - JS libraries: jQuery, JS Plumb for U/I
- Simple and beautidul U/I creating using CSS. Basic Cognitive Science principles were applied:
  - Instruction screen before quizzes and practice tests
  - Easy to use boxes at home screen for selecting quiz/practice
  - Well positioned buttons to switch from DFA-NFA-PDA
  - Separate functions for creating states, marking a state as final state, changing positions
  - Very natural U/I for debugging part, so that students can very easily realize if there is any fault in their automata.
  - Negative(to be rectified): The box of the automaton shall be positioned better so that both the functions and the automaton are visible in the same screen.
- Server security:
  - Protected from basic attacks like sql injection, code injection, etc.
  - Captcha
  - Directory traversal disabled
  - Quiz-data fetched through backend php script on sql server., not visible to students
- Testing correctness of Automata: For checking purposes, we keep a set of testing cases corresponding to each question on the server (the data is stored on a mysql database). If all the test cases are passed, we assume that the user has answered the question correctly, and his score is updated. The professor is expected to insert sufficiently complicated test cases.
- Novel scoring mechanism: Every quiz contains 3 questions, each of equal weightage. The maximum marks in each quiz is 30. The starting time is noted as soon as a student starts attempting the quiz. Score is a value that it reflects the number of questions successfully solved in the quiz. The score is multiplied by 10, to give a value from 0 to 30. Now, if the total attempt duration is more than the optimal duration given by the admin, the student receives $1/3^{rd}$ of his/her original score.
  However, if the attempt duration is less than the optimal time, we allot the score as:
  $FinalScore = Score - (Score * (2/3) * (attemptduration/optimalduration))$
  Hence, score is a continuous function of the attempt duration, with the minimum value being $(correctanswers * 10)/3$ and the maximum value being $(correctanswers * 10)$.

4.1 Database

The database stored on the server is an SQL database named secure_login. The tables are:

- members - It maintains a list of the users alongwith username, email and password.
- questions - It stores the quiz no., question no., accepted or rejected fields.
- statement - It stores the quiz no., question no. and statement.

- quiz_time - It stores the optimal time for the quiz. populated by admin.
- score - It stores the score of a particular student in a quiz alongwith the starting time. An entry with score 0 is made as soon as the student starts attempting the quiz. This is because the student should not be able to give the quiz multiple times.

4.2 Admin

The admin has the ability to create or edit quiz and also add questions. He gives the problem statement for the 3 questions, and corresponding accept and reject testcases for all the questions. Also, the admin gives the optimal time for the quiz, which is used in scoring the students.
Thus, the whole control of the system is given to this admin panel whose username and password is different from normal users. This admin panel is very easy to use, and some instructions can be added to make it even better.

4.3 U/I features

The students can select the quiz based on the topics on which they want to practice questions. For example, if I choose DFA as the option then I can get a question like: Given a regular expression 0^*1^*, construct a NFA that accepts it. The users can create states like start state, accept state or any intermediary state, put arrows from one to another and change their positions as per requirement. After clicking on the submit button, the user can check his score which is displayed as per number of questions answered correctly. For better understanding, some examples can be given to the user.

- Any state of the automata can be created using Shift+Click.
- To make a state as final, the user has to double click on the state.
- The states can be connected by arrows and it asks for values for the transition.

4.4 Secure Login Form

The user registers with a username, email-id and password. We have kept captcha for extra security. After logging in, the user is provided with a list of the quizes. He can select the button corresponding to the quiz he wants to participate in.

5 Future work

The concept of the automaton builder and simulator, as we have created can be modified into a multiplayer game where instead of having a fixed time, students could compete to complete the automaton before the other person. This would require some socket programming.
Another scope of this project is that the database can be modified to store questions for quizes and exams, and then it can be used to conduct examinations in the course Theory of Computation for future batches. Furthermore, questions related to Turing Machines can be added in the quiz.
As we have already mentioned in this report, some difficult features such as DFA Minimization, conversion from NFA to DFA, etc. are also in the pipeline. We already have some code for DFA minimization, which could not be integrated with the U/I since it required some complicated modelling of the JSON object associated with every automata. We will try to implement these features before we roll out the application for future batches.

6 Acknowledgements

We thank Professor Bhattacharya for guiding us through this project. We also acknowledge our inspiration from the site http://madebyevan.com/fsm/ while doing this project.