4TB3 Proposal: Checking Answers of Question on Compilers for Correctness

Joseph Seger, Fawaz Ismail

March 2014

1 Overview

We will take the approach of making an informative and interactive webpage designed to teach the basic concepts behind finite state machines to those with little or no previous experience with the topic.

The format of the webpage will be chapter by chapter and will progress from the first chapters covering the more basic concepts on to more and more challenging topics. There will be approximately five to seven chapters.

Each chapter will contain a title of the topic to be covered, followed by an in-depth explanation. In order to provide the best possible learning experience, the explanation will be simple and easy to follow. This will also include diagrams to demonstrate each section. Upon completion of the chapter, the user will be prompted to complete an optional challenging exercise to test their comprehension and understanding. The user's answer will be checked for correctness and feedback will be provided. If the user completes two unsuccessful attempts to the exercise, they will be provided with the option to view a possible answer.

The lessons that will be taught via the webpage are based on material from our 4F03 course notes[4] and the Compilers textbook[2]. The content will be related to Chapter 3 from the textbook, as well as section 3 of the slides. Each of which covers Regular Languages and Finite State Machines. The W3 Schools website[1] will serve as a resource of information for effective methods of checking answer correctness in JavaScript, and aid in the development process.

Finally, since the purpose of the webpage is to be a teaching tool, we will reference Javier Bargas-Avila and Glenn Oberholzer's research into form validation errors[3] to ensure user feedback is delivered in an effective way,

2 Chapter Topics

Topics that will be discussed in the ebook will all relate to finite state machines in some way. The following topics will be included:

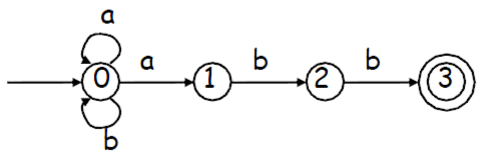
- Regular Grammars - What they are and how they relate to finite state machines.
  Exercise: Design a grammar which produces the given language.

  1. All strings of a's and b's with an even number of a's and an odd number of b's.
  2. Canadian telephone numbers

  Exercise: Describe the language the given grammar would produce.

  1. [a](b c){d}
  2. { [0] 1 }

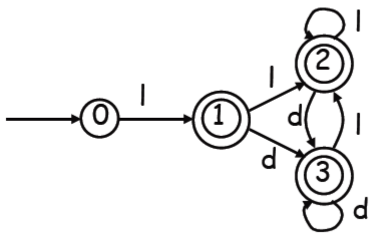
- Finite State Machines - Construction and mechanics.
  Exercise: Translate a given finite state machine into a regular grammar.

  1. 

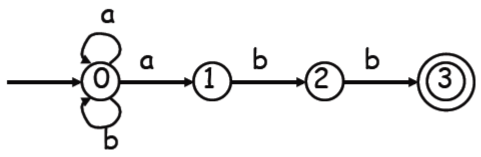
  Exercise: Construct a finite state machine given the following regular grammar.

  1. Finite State Machine for each exercise of Regular Expressions Section.

- State minimization - Concept behind "redundant" states.
  Exercise: Minimize the given finite state machine.

  1. 

- Determinism - Concept behind determinism vs nondeterminism.
  Exercise: Create an equivalent deterministic finite state machine given the following nondeterministic finite state machine.

  1. 

3 Approach

The real difficult aspect of the assignment will be creating code that checks the correctness of the reader's exercise answers. The code for each exercise will be written in Javascript and will be robust enough to check the correctness of many different problems of the same type (e.g. multiple state minimization problems all based on different finite state machines). Each type of problem will have its own unique correctness checker.

3.1 Regular Expressions

For the Regular Expression questions, we will use a method similar to that of JavaScript Form Validation [1]. The user will be provided with a textbox to input their answer to the problem. Upon clicking the submit answer button, the program will parse through the input, comparing to the expect result. Finaly outputting either Success for the correct answer, or prompting to redo question for the incorrect answer.

3.2 Finite Automata

Validating the finite state machines will be a more difficult task. Initially we plan on allowing the user to construct Finite Automata using a drag an drop method on the website. The ussue lies in validtating the final answer of the user. This will require more research into JavaScript, as well as brainstorming to create an intuitive solution to validating the state machines.

4 Timeline

March 7th Project Proposal Due

March 13th Website Skeleton Completed and Method For Finite State Diagram Validation Decided

March 20th Regular Grammars Section Completed (Lesson and Excercises)

March 27th Finite State Machines Section Completed (Lesson and Excercises)

April 2nd State Minimization and Determinism Sections Completed (Lesson and Excercises)

April 4th Report and Implementation Due

April 7th Poster and Presentation Due

References

[1] Javascript form validation. `http://www.w3schools.com/js/js_form_validation.asp`. Accessed: 2014-03-06.

[2] Ravi Sethi Jeffrey D. Ullman Alfred V. Aho, Monica S. Lam. *Compilers: Principles, Techniques, and Tools*. Pearson Education Inc., second edition.

[3] Glenn Oberholzer Javier Bargas-Avila. Online form validation: Don't show errors right away. *Human Computer Interaction*, pages 848 – 851.

[4] Emil Sekerinski. 4f03 course slides. `http://www.cas.mcmaster.ca/~cs4tb3/2013_14/Downloads_files/Slides.pdf`.