Clickstream Data from a Formal Languages eTextbook*

Mostafa Mohammed, and Clifford A. Shaffer
Virginia Tech, Blacksburg VA, USA
{profmdn, shaffer}@vt.edu

ABSTRACT

When students interact with an eTextbook, it typically logs their interactions while engaged in activities like watching a visualization, attempting to solve an exercise, or refreshing the page. These event logs allow instructors and researchers to evaluate students' engagement level and approaches to using the artifacts. We predict that the way students use the book and the artifacts affects their performance on the exercises, their learning gains, and their performance in other aspects of the course.

In this paper, we describe a data set gathered from a complete semester course on Formal Languages. This includes all student interactions with the Formal Languages eTextbook. The book contains a set of auto-graded exercises and visualizations about Formal Languages course contents in the form of slideshows.

Keywords

OpenDSA, Formal Languages, auto-graded exercises, Interactions logs

1. INTRODUCTION

Formal Languages course is a theory course that contains a number of proofs and on-paper assignments. Formal Languages courses face a few challenges. They are often presented as fairly abstract and highly mathematical. This has the benefit of making students practice useful skills like proof writing, but might make it less appealing to students more used to the hands-on style of the typical CS programming course. A typical FLA class presents several models of computing (deterministic and non-deterministic finite state machines, regular expressions, push-down automata, context-free languages, Turing machines), with many proofs about their relationships and limitations. There are many algorithms associated with each model that students must learn to apply. Many instructors have their students use simulators to support this process, such as the state-of-the-art simulator Java Formal Languages and Automata Package (JFLAP) [7, 2]. JFLAP simulates most of the models used in Formal Languages courses, so it helps students by allowing them to watch different models, apply different algorithms on these models, or test these models with different input strings.

To increase student understanding and interaction with the course materials we implemented an eTextbook using the OpenDSA system. The OpenDSA project [1] is concerned with building complete eTextbooks for different topics in computer science like Data Structures and Algorithms, Computational Thinking, or Formal Languages. These eTextbooks are enhanced with various embedded artifacts such as visualizations, exercises with automated assessment, and slideshows to improve understanding. OpenDSA allows instructors to create instances of complete interactive eTextbooks that integrate interactive artifacts with the textual content. OpenDSA contains slideshows produced using the JSAV (JavaScript Algorithm Visualization) framework [3] to support various topics in undergraduate courses.

We used our eTextbook in a Formal Languages class of 60 students for an entire semester. We collected complete interaction log data detailing use with the book. The data includes detailed students interaction with various slideshows, and interactive exercises. The data set is a unique data set for the researchers where it includes senor students interactions with sophisticated book contents. Students deal with different exercises that ask students to apply different algorithms (i.e. Convert an NFA to DFA, or Minimize a DFA), build complex models (i.e. DFA, NFA, PDA, and TM), or write different grammars for languages. We are making this data-set available to researchers via DataShop. So researchers can get a complete data-set on senior-level students accessing a theory eTextbook course. This is different that usual data sets that contains data about students interactions with programming courses.

2. OPENFLAP

JFLAP is used extensively in FLA courses to help students visualize and observe the behavior of models and associated algorithms [8]. However, JFLAP has three disadvantages from the point of view of integrating material into an eText-

*(Does NOT produce the permission block, copyright information nor page numbering). For use with ACM_PROC_ARTICLE-SP.CLS. Supported by ACM.

book. First, it was written in Java and is a stand-alone application that runs on the student's machine. This does not allow it to easily tie to online tools like OpenDSA, or to an LMS [4, 6]. Second, JFLAP does not have any mechanism for auto-grading exercises. Students can use JFLAP to help solve many typical homework problems, such as creating a machine to recognize a given language. But they get little feedback from JFLAP about whether their answer is correct. Instead, they must wait until the homework is hand graded by instructional staff. In contrast, we have reached the state where many programming assignments can be done with immediate feedback from auto-graders, largely based on testing the program against unit tests.

These drawbacks inspired us to develop an open-access, web-based version of JFLAP with enhanced support for auto-graded exercises. We have largely re-implemented JFLAP functionality within the OpenDSA framework. We refer to it as OpenFLAP. OpenFLAP is implemented using the JSAV library. OpenFLAP also allows us to create exercises, auto-assess them, and report the result to an LMS through OpenDSA's standard framework [5].

3. OPENFLAP EXERCISES

OpenFLAP allows us to create two types of exercises.

- Auto-Graded exercises Auto-graded exercises ask students to build different models, i.e., Deterministic Finite Automata (DFA) or writing a Context-Free grammar. These exercises are similar to programming exercises. To test students' solutions, instructors can assign some test cases. That can be used to test the correctness of students model/grammar.
- Proficiency exercises Proficiency exercises ask students to apply an algorithm to a given model like convert an NFA to a DFA. OpenFLAP allows instructors to create proficiency exercises where students need to apply algorithm steps on a given model. OpenFLAP checks the correctness of every student step and shows a message to the student to prompt them to retry the incorrect step before moving forward to the next steps

4. STUDENTS INTERACTION DATA-SET

When students work with our eTextbook, the OpenDSA system collects data about students interactions with the book components. The book contains several slideshows, exercises, khan-academy exercises, and traditional text with some images. Students need to answer all exercises to earn credit and they can freely skip looking at the slideshows or read the text. Our Formal Languages eTextbook includes:

- Prose and images. Traditional text about the algorithms and proofs for different Formal Languages models. We added some images that can help to understand the text.
- Slideshows A series of slides is often used to describe a topic to students. Slideshows include four buttons that allow students to navigate in the slide show. These buttons are a) next slide, b) previous slide, c) first slide, and d) last slide. Figure 1 shows an example for NFA to DFA slideshow.

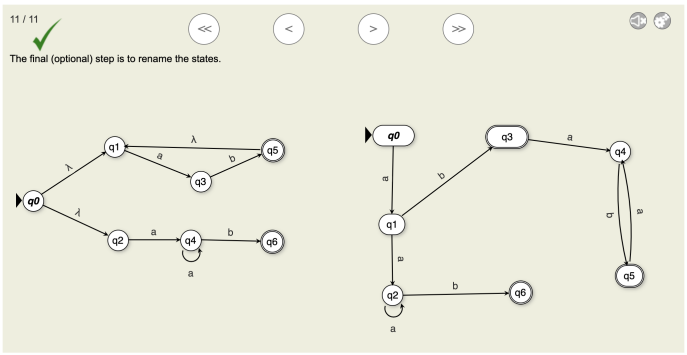

Figure 1: Slideshow example for NFA to DFA algorithm.

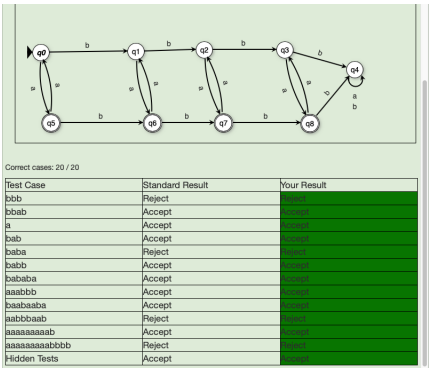

Figure 2: Auto-graded exercises to create a DFA.

- Auto-graded exercises and Proficiency exercises. A large number of exercises are available, related to building various example machines. Exercises are included that require students to build Deterministic and Non-Deterministic Finite Automata, Push Down Automata, or Turning Machines. Some exercises are about writing Grammars for a given language, or converting a model to another model. All exercises require students to score 100% correctness to get the credit for the exercise. Students can repeat the exercise as necessary to achieve credit. Figures 2 and 3 shows examples for Auto-graded exercises and proficiency exercises.
- Multiple Choice, T/F, Fill-in-the-blank. OpenDSA includes many questions in standard simple question formats, implemented using the Khan Academy Exercises Framework. Figure 4 shows an example for a Khan Academy exercise.

Every primitive user interaction (button clicks, page loads, window focus and blur events) is captured and stored in the database. Table 1 lists specific events from the data set along with their meaning.

5. DATA FORMAT

The data comes in the form of a CSV file with 262205 rows, where each row is an event that is made by a student. Each event row includes the interaction_ID, user_ID, event name, event description, event time, browser name, Operating System name, Device used, and chapter id.

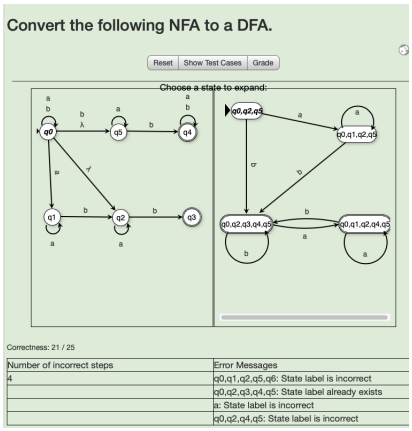

Figure 3: Proficiency exercise to convert an NFA to DFA.

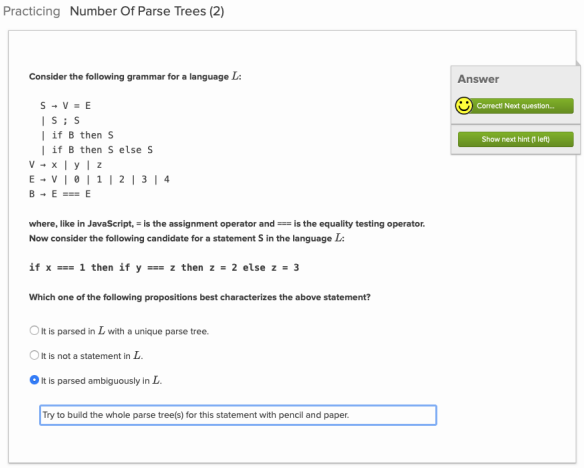

Figure 4: Khan Academy exercise.

event	description
Window-load	loaded a book module
Window-focus	Window focus
jsav-forward	Slide show forward button
jsav-exercise-reset	Exercise reset button
jsav-exercise-grade	Request exercise grade
jsav-matrix-click	Click in cell for grammar production
jsav-node-click	Select a graph node
submit-deleteButton	Deleted a graph node
submit-edgeButton	Button click: enter add-an-edge state
window-unload	Closed the module

Table 1: Some events types from the data set.

The data set can be found at `https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=3427`.

6. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grants DUE-1139861, DUE-1431667 and IIS-1258471. The Egyptian Ministry of Higher Education funded Mostafa Mohammed during his PhD. We are grateful to the many, many students who have worked on OpenDSA, OpenFLAP, and the FLA eTextbook over the years.

7. REFERENCES

[1] E. Fouh, V. Karavirta, D. A. Breakiron, S. Hamouda, S. Hall, T. L. Naps, and C. A. Shaffer. Design and Architecture of an Interactive ETextbook–The OpenDSA System. *Science of Computer Programming*, 88:22–40, 2014.

[2] JFLAP website. `http://jflap.org`, 2020.

[3] V. Karavirta and C. A. Shaffer. JSAV: the JavaScript Algorithm Visualization Library. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, pages 159–164. ACM, 2013.

[4] M. Mohammed, S. Rodger, and C. A. Shaffer. Using programmed instruction to help students engage with etextbook content. *The First Workshop on Intelligent Textbooks*, 2019.

[5] M. Mohammed, C. A. Shaffer, and S. H. Rodger. Teaching formal languages with visualizations and auto-graded exercises. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 569–575, 2021.

[6] M. K. O. Mohammed. Teaching formal languages through visualizations, simulators, auto-graded exercises, and programmed instruction. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 1429, New York, NY, USA, 2020. Association for Computing Machinery.

[7] S. H. Rodger and E. Gramond. JFLAP: An aid to studying theorems in automata theory. *Integrating Technology into Computer Science Education*, 30(3):302, 1998.

[8] S. H. Rodger, E. Wiebe, K. M. Lee, C. Morgan, K. Omar, and J. Su. Increasing Engagement in Automata Theory with JFLAP. In *ACM SIGCSE Bulletin*, volume 41, pages 403–407. ACM, 2009.