Increasing Student Interaction with an eTextbook using Programmed Instruction*

Mostafa Mohammed[1] [0000−0002−0652−2817] and Clifford A. Shaffer[2] [0000−0003−0001−0295]

Virginia Tech, Blacksburg VA, USA {profmdn, shaffer}@vt.edu

Abstract. Textbooks for theory courses in CS tend to be heavy on prose and mathematics. We find that students do not engage such material, and skip or rush through it without understanding. To increase students level of engagement, we developed support within the OpenDSA eTextbook system support for creating materials based on the Programmed Instruction pedagogical paradigm. This requires near-constant activity by the student, who must read a little, ideally a sentence or a paragraph, and then answer a question or complete an exercise related to that information. Based on the question response, students are permitted to continue, or must retry to solve the exercise. Versions of the eTextbook have been used to teach the senior-level Formal Languages course at Virginia Tech for two semesters. In this demonstration, we show how students interact with material developed using the Programmed Instruction approach.

النثر

Keywords: Programmed Instruction · Formal Languages · Engagement · Book Interactions · OpenDSA eTextbook.

1 Introduction

Programmed Instruction is an instructional approach first championed by B. F. Skinner when he noticed some flaws in the teaching process in schools [3]. Skinner found that instructors give little attention to individual students in class, books provide no immediate evaluation of student solutions, and since not all students have the same prior knowledge, some of them are not prepared to acquire knowledge from the textbook. These points made Skinner think about a different teaching technique called the Programmed Instruction (PI) machine [5]. The PI machine works by presenting a small piece of information (a sentence

* Supported by NSF under grant DUE-1432008. 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.

or short paragraph, called a *frame*) followed by a question that students must answer to be able to see the next frame. If the student fails to answer the question, the student must stay in the current frame and repeat the question. In Skinner's model of teaching, the machine acts like a personal tutor to the student by giving them immediate feedback for each response. Students see the PI machine like a game that gives them checkpoints after solving a question and allows them to proceed in the book further after each successful answer. Skinner [5] claimed that students would be able to learn twice as much with the same time and effort as compared to the traditional teaching method.

After Skinner created his machine, similar teaching machines started to appear during and after the 1950s. During this period, PI research revolved around addressing instructional effectiveness, learner pacing, reinforcement strategies, and long-term effects [4]. Although there was considerable interest in PI due to Skinner's *Teaching Machines* and *Technology of Teaching* [6], PI was superseded by Computer-Assisted Instruction (CAI), and the Keller Plan in the 1970s [3, 2].

The Personalized System of Instruction (PSI, or Keller Plan) became widely used during the 1970s and 1980s [2]. Under the Keller Plan, students work on the given materials at their own pace. Students can study the modules at any time and as much as they want without the need to wait for the instructor to explain the topic. Instructors then can focus on students that struggle with the material. In the Keller Plan, the instructor's role is minimized. Instructors should decide what content students have to master, guide students through their studying, and give tests and exams to students. Class time under the Keller Plan is just a place for students to study their materials and take tests.

The OpenDSA project [1] is concerned with providing visualizations, content, and interactive exercises for topics in computer science like Data Structures and Algorithms, Computational Thinking, or Formal Languages. These eTextbooks are enhanced with embedded artifacts such as visualizations, exercises with automated assessment, and slideshows to improve understanding. OpenDSA has many features found in the Keller Plan. OpenDSA has modules, and throughout each module, there are exercises that students should solve to prove their mastery of the module. OpenDSA does not prevent students from moving further if they are failed to solve these exercise. But it does follow the mastery approach in that students can repeat exercises as many times as they wish, eventually receiving credit whenever they demonstrate mastery.

2 Programmed Instruction eTextbook for Formal Languages

Programmed Instruction frames limit the student's ability to skip content by enforcing them to satisfy a particular framed-satisfaction criterion, typically answering the frame question. Whenever possible, we design the questions to present different examples every time the student reloads the page. Suppose a student wants to study for the final exam. If he/she sees the same question or the

same example that he already solved before then this will not helping him/her to practice effectively. The Frames system randomly selects a different example from a pool of available questions and auto generate frame questions.

2.1 Frame question types

Currently, the Frames system supports a number of different types of questions

- MCQ questions. The majority of Frame questions are MCQ. The idea for PI is to answer the question to master the current information. So, having MCQ question allows students to find a way to escape from the question if they could not find the correct answer. This may case some students to pass some questions without understanding the related information to the given question. To overcome this problem, we added support to Frames questions to give students a message to tell them why their answer is correct / not correct. This message can give students hints in cases that they do not answer the question correctly, see Figure 1(a). On the other hand, this message gives students an enforcing statement to make the information more clear to students in case they answer the question correctly, see Figure 1(b). More examples for Frames questions are shown in Figure 2
- Embed an exercise IFrame. In general, instructors can embed any type of HTML exercises in an IFrame inside the frameset.

2.2 Frames checkpoints and mastery points

Students are required to successfully solve frames questions to be able to proceed in the frameset. Every time students solve a question the Frames system stores their progress. This way students can retrieve their progress if they closed their book or refreshed the module page, saving their time and effort to redo all the question they already solved. Once students reach the end of the frameset, the Frame system awards them mastery credit for completing the frameset.

2.3 Demonstration

In our demonstration we will present our eTextbook for the Formal Languages course. The demo will show examples of framesets that help students to study Formal Languages topics. We used this material to teach the course for two semesters, Fall 2020 and Spring 2021. We will demonstrate our infrastructure that allows instructors to create frame questions and provide useful hints that can guide students to answer the frame question.

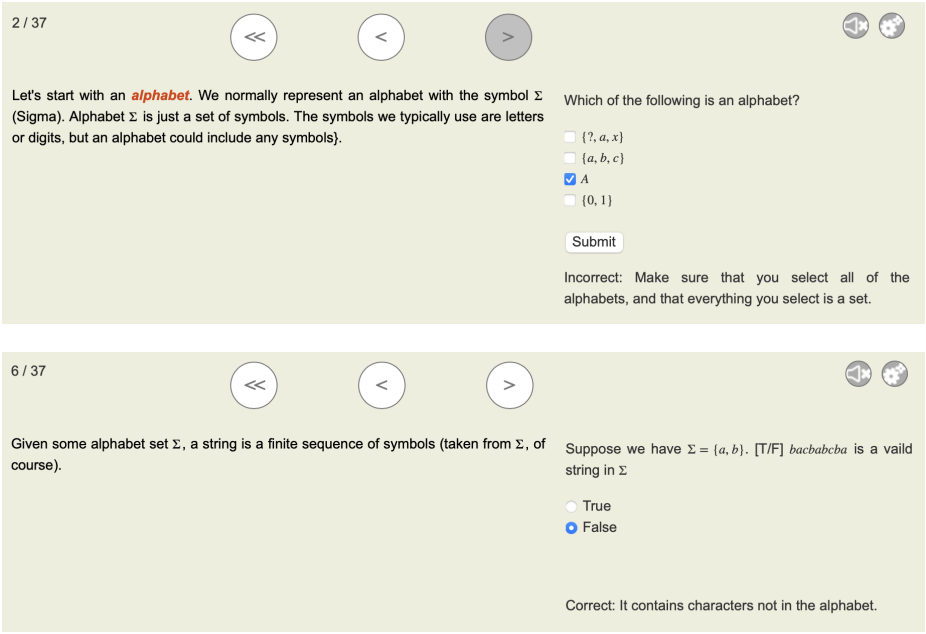

Fig. 1: PI Frames hints for correct/incorrect students answers

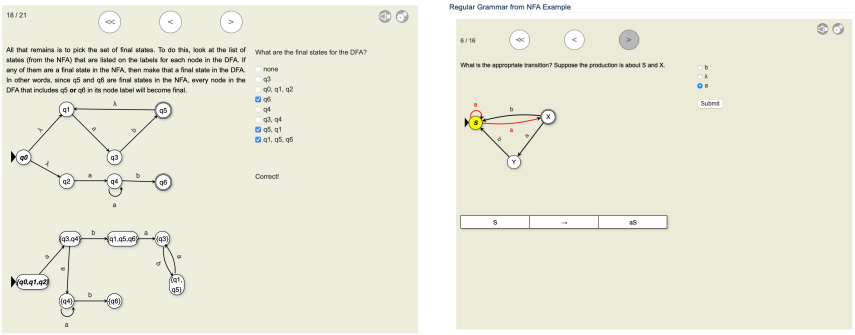

Fig. 2: PI Frames examples

References

1. Eric Fouh, Ville Karavirta, Daniel A Breakiron, Sally Hamouda, Simin Hall, Thomas L Naps, and Clifford A Shaffer. Design and Architecture of an Interactive ETextbook–The OpenDSA System. *Science of Computer Programming*, 88:22–40, 2014.
2. Fred S Keller. "GOOD-BYE, TEACHER..." 1. *Journal of applied behavior analysis*, 1(1):79–89, 1968.
3. Barbara Lockee, David Moore, and John Burton. Foundations of Programmed Instruction. *Handbook of research on educational communications and technology*, pages 545–569, 2004.
4. Michael Molenda. When Effectiveness Mattered. *TechTrends*, 52(2):53, 2008.
5. BF Skinner. Programmed Instruction Revisited. *Phi Delta Kappan*, 68(2):103–10, 1986.
6. Burrhus Frederic Skinner. Teaching Machines. *Science*, 128(3330):969–977, 1958.