Automatic Automata Grading System Using JFLAP

Melisa Biçer
Computer Engineering
Tarsus University
Mersin, Türkiye
melisabicer@tarsus.edu.tr

Ferhat Albayrak
Computer Engineering
Cukurova University
Adana, Türkiye
falbayrak@cu.edu.tr

Umut Orhan
Computer Engineering
Cukurova University
Adana, Türkiye
uorhan@cu.edu.tr

2023 Innovations in Intelligent Systems and Applications Conference (ASYU) | 979-8-3503-0659-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ASYU58738.2023.10296744

***Abstract*— This study focuses on the automatic grading of exams in the "theory of computation" course using JFLAP, an open source system. In the study, the source codes of JFLAP are arranged and two interfaces are presented, one for the student and the other for the evaluator. Practical exams with JFLAP support cover three topics: Finite Automata (FA), Pushdown Automata (PDA), Turing Machines (TM). In the student interface, files are sent to a server as soon as responses are recorded using traditional JFLAP. The evaluator adds an answer key to the same folder as the student files on the server and clicks the auto-grading button in the evaluator interface. The student file cannot get points unless more than half of the balanced distributed accept and reject samples in the answer key are correct. This approach reduces the time and subjective interpretation costs associated with traditional assessments, especially in large classrooms. The system's performance analysis shows increased exam participation and classroom interaction without affecting ultimate success. It is expected that the success rates of students who make more preparations in future studies will increase.**

Keywords*— *jflap, automatic grading system, finite automaton, turing machine, pushdown automaton

I. Introduction

With the advancement of technology, digital tools have found their place in every field. One of these fields is education. The digital tools used in education offer various benefits for both educators and students. The need for online educational tools [1]–[3] has become more evident with the Covid-19 pandemic. Efforts to make education more effective are being carried out under different headings, including self-learning [1]–[4], computer-assisted simulation [5], [6] computer-assisted instruction [7], [8] and computer-assisted assessment [9]–[12]. Regardless of the specific heading, all educational processes require the evaluation of students. The ability to automatically conduct this evaluation process through digital tools has significant advantages. For example, it can encourage educators to conduct more evaluations. On the other hand, conducting objective evaluations and promptly providing results will identify problems in the learning process in a timely manner. Additionally, the use of such a system in classrooms with a large number of students will be highly beneficial in reducing the workload on educators and standardizing grading.

Despite the advantages of automated assessment systems, they may not be able to comprehensively test students' learning skills due to the limitations of digital tools. However, intensive literature studies conducted under the headings of automatic short-answer grading [10] and automatic composition scoring [11]–[13] indicate that success is gradually increasing. Recognizing this limitation, a research group has developed a Java-based automated grading system to analyze students' processing behaviors and predict their academic performance [9]. In another study, the focus was on automatic grading methods in MOOC systems, specifically those related to 3D modeling online courses [14].

One of the computer-assisted educational systems, the open-source Java Formal Languages and Automata Package (JFLAP) software tool [5], can be used by students as a supportive tool in courses on computation theory and automata theory. While students can conduct experiments and evaluations on JFLAP on their own, this tool is not specifically designed for direct knowledge-based assessment by an evaluator. Nevertheless, some researchers have proposed a new approach to enhance the interactive quality of JFLAP by creating automatic application problems and providing quick and meaningful feedback to students regarding problem solutions [15]. In a study conducted by Shekhar et al. [16], the aim was to analyze students' progress while solving finite automaton problems using JFLAP, and to develop a simple and intuitive tool mechanism for students to solve the problem. In another study [17], researchers developed an application called OpenFLAP, which is similar to JFLAP but accessible through the web and supports automatic assessment by generating exercises. In yet another study [18], the semantic equivalence between student and instructor solutions is checked using Automatic Automata Checker (A2C), an open-source library, and feedback can be provided to improve students' understanding. These automatic systems allow for faster and more accurate evaluation of individuals in the field of education, while also significantly saving time and effort for educators in the assessment process.

This study aims to transform the traditional problem-solving approach in computation theory exams into simulations using the JFLAP environment. Based on this idea, it has been determined that the JFLAP application, whose open-source codes were examined, can be transformed into a tool for automatic assessment in three topics of computation theory with some modifications: finite automaton, push-down automaton, and turing machine. Within this scope, two separate interfaces of the JFLAP application were developed—one for students and the other for evaluators—in order to enable automatic grading for these three topics in the computation theory course. The assessment method was determined, and some results regarding the exams administered to students were shared.

II. Material And Method

A. Automata Approaches

1) Finite automaton

A finite automaton is a machine or control mechanism designed to automatically follow or execute programmed instructions [19], [20]. It is commonly referred to in two types: deterministic and non-deterministic. A deterministic finite automaton (DFA) defines a unique and unambiguous state transition for any input symbol, while a non-deterministic finite automaton (NFA) may include uncertainties (arbitrary, zero, or multiple state transitions) for an input symbol. Both DFA and NFA are equivalent to each other and have the ability to solve Regular Grammar problems at the Type-3 level in the Chomsky hierarchy. The automaton editor in the

JFLAP application allows us to ask both DFA and NFA questions.

2) Pushdown automaton

A Pushdown Automaton (PDA) is a type of automaton that can recognize context-free languages. It consists of three components: an input tape, a control unit, and an infinitely large stack [19]. On the stack, it performs operations such as adding a new symbol to the stack, reading the symbol at the top of the stack, and removing symbols. It has the ability to solve Context-Free Grammar problems at the Type-2 level in the Chomsky hierarchy. The PDA editor in the JFLAP application allows us to work on non-deterministic CFL problems.

3) Turing machine

The Turing machine (TM), developed by Alan Turing, serves two fundamental purposes: deciding formal languages and solving mathematical functions [19]. Turing machines are mathematical models that can perform read and write operations on cells on an assumed infinite tape. Their main objective is to determine whether a string entered on the tape belongs to the language family of a given problem. They have the ability to solve Turing Machine problems at the Type-0 level in the Chomsky hierarchy. The JFLAP application is also highly effective in evaluating Turing machines in its TM tool.

B. Java Formal Languages and Automata Package

JFLAP, a free software developed by Duke University since 1993, is a popular open-source tool widely used in formal languages and automata courses [21]. JFLAP [5] enables the creation of automata, grammars, L-systems, and regular expressions, as well as the manipulation of these expressions. It can be categorized into main operation groups, including exploring the language of structures, converting between equivalent structures, and performing various analyses on structures. It is commonly used by undergraduate students and facilitates the visual and interactive understanding of theoretical concepts.

Although students can conduct individual experiments using JFLAP, the evaluator needs to open and test each student's file in the application, enter the words to be evaluated, and adjust the scoring. Such a method poses significant challenges in terms of workload and calculation accuracy, particularly when dealing with a large number of students. Therefore, the development of an automatic evaluation system would alleviate the evaluator's workload, enable students to receive feedback more quickly, and allow them to solve a greater variety of problems. Considering these advantages, in this study, the JFLAP application has been reorganized into two separate interfaces: one for students and another for evaluators.

1) Student interface

The process of saving files for students consists of two stages: saving the file and sending it to the evaluator's server. Students log into the system using their first name, last name, and student number. They then select one of three buttons: FA, PDA, or TM, to solve the problem provided by the evaluator. After designing their solution, when the student clicks the save button, their JFLAP file, along with their student number, name, surname, file name, and MAC address information, are encrypted and saved to the server. The files sent to the server are stored in separate folders based on the type of problem the student has answered. For example, when a student designs and saves an FA, a folder named FA is created on the server, and the student's solution file is saved inside that folder.

TABLE I. EXAMPLE OF ANSWER KEY

JFLAP					
FA		PDA		TM	
111101	A[1]	abbb	A[1]	abbaba	A[1]
10101	A[1]	bbb	A[1]	bababa	A[1]
0	R[2]	bbbbbb	A[1]	bbb	R[2]
10	R[2]	bbaab	R[2]	abaa	R[2]
000	R[2]	a	R[2]	bbaa	R[2]

1: A-Accept, 2: R-Reject

2) Evaluator interface

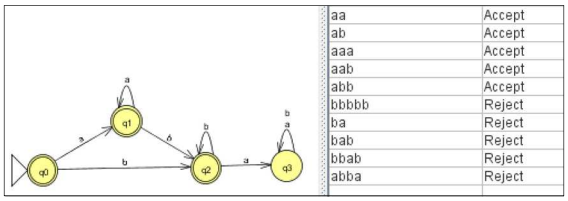
In the evaluator interface of JFLAP, the aim is to automatically evaluate student solution files. To enable automatic evaluation of the JFLAP files stored on the server in the evaluator section, a testing mechanism and the process of saving the scores obtained by students to a file have been implemented. When the evaluator runs the JFLAP application on the server, they examine the directory where the uploaded files of the students are located. The three types of answer files uploaded by students are scanned in separate folders (FA, PDA, and TM). The problem type is determined based on the names of these folders. Each folder is scanned individually, and the JFLAP files inside them are processed. To test the JFLAP files, the evaluator utilizes an answer key prepared by them, an example of which is given in Table 1.

The system is capable of performing the testing process when a separate answer key is prepared for each question. The evaluator creates an answer key by indicating the inputs they want to test in separate files for each problem type, and writing "A" for accepted inputs and "R" for rejected inputs. After simulating the designed solution using JFLAP, the evaluator compares the obtained results with the answer key. Then, they carry out the scoring process and write the student's name, ID number, number of correct answers, number of incorrect answers, and score into a file. The process continues until all the files in the developed JFLAP module are read and the students' responses are evaluated within each folder. The evaluator process is given as a flow diagram in Fig. 2.

III. RESULTS AND DISCUSSION

In this study, JFLAP, coded in Java language and with open-source code [21], was developed for the automatic evaluation and scoring of FA, PDA, and TM structures by the evaluator. The application was built with two separate interfaces: one for students and one for evaluators. The application was tested in the Theory of Computation course in the Department of Computer Engineering at Çukurova University during the fall semester of 2022-23, and students were administered the exam four times. For a balanced assessment, each question had a minimum of 5 Accept and 5 Reject responses defined, and for scoring, incremental increases starting from more than half were planned. For example, if there were 10 possible answers defined as 5A+5R, the scoring was performed by incrementing 20 points for each additional correct answer, starting from 6 correct answers (20

points for 6 correct answers, 40 points for 7 correct answers, and so on).



aa	Accept
ab	Accept
aaa	Accept
aab	Accept
abb	Accept
bbbbb	Reject
ba	Reject
bab	Reject
bbab	Reject
abba	Reject

Fig. 1. An example of finite automaton evaluation.

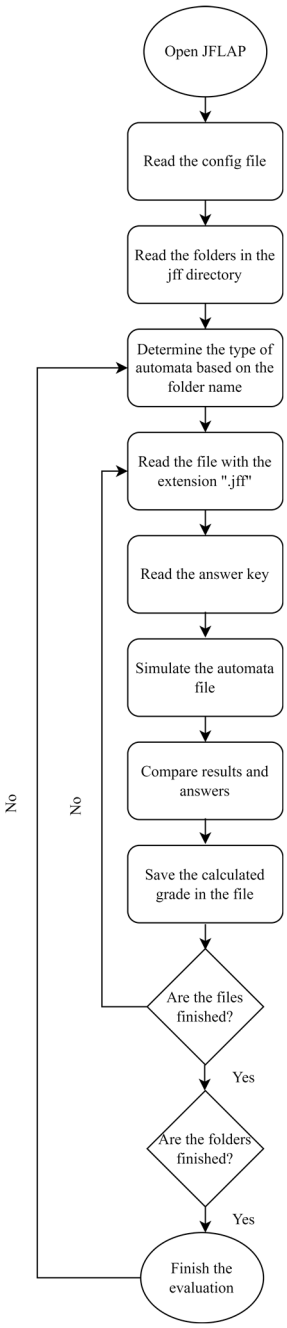


Fig. 2. Flow diagram of the evaluator.

In the exam implementation, a plan was devised with tight time constraints, allowing students to conduct self-tests for each question and then submit their answers. Fig. 1 presents the input strings and their corresponding result values. If the input is suitable for the designed solution, an "accept" result is obtained; otherwise, a "reject" result is obtained. In the student application, when the save function is executed, the designed solution is transferred to the evaluator's server. Students can submit multiple solutions they have designed separately, and the solution with the highest score is considered valid for evaluation. In the evaluator application, student files are tested with the prepared answer keys, resulting in a scoring file. The scoring file contains the student's name, surname, file name, number of correct and incorrect answers, and the grade they received.

During the period in which the application was used, a total of 175 students took the course (including evening classes). Four separate exam sessions were conducted, including two midterm exams, a final exam, and a makeup exam. Depending on the available laboratory facilities, some exams were completed in two sessions, while others required three sessions. Different sets of questions were given to groups in the exams conducted over three sessions. As a result of these exams conducted within the scope of this study, 79 students passed the course with varying grades, and the success rate was calculated to be around 45%. Considering that the success rate for the same course with traditional exams before the pandemic was between 40% and 54%, it was determined that the practical exam method with JFLAP did not significantly affect the success rate. On the other hand, in traditional exams, the number of students taking the makeup exam ranged from 12 to 33 during the three-year period before the pandemic, and the success rate remained between 10% and 17%. In the applied makeup exam, however, the participation increased to 59%, and the success rate reached 39%. Considering that the method encourages participation, does not decrease the success rate, and facilitates exam evaluation, it can be concluded that the method is effective and successful in achieving its objectives.

IV. Conclusion

This study describes the work conducted on the automatic evaluation of exams for the computational theory course offered in computer science undergraduate education, along with the resulting system. The open-source system known as JFLAP was modified by making changes to its code to develop two interfaces: one for students and the other for evaluators. Within this framework, exams were conducted focusing on the topics of FA, PDA, and TM, which are also supported by JFLAP. In the student interface, after designing the solution to the problem provided by the evaluator, the student's information and the designed answer file are transferred to the server where the evaluator is located. The evaluator also uploads the answer keys prepared for each problem test to the corresponding folders on the same server. Subsequently, when the JFLAP evaluator interface is opened, the evaluation mechanism is executed separately for each folder available on the server.

The field of computational theory, in which JFLAP is used, inherently involves complex and comprehensive problems. Consequently, the traditional approach of preparing and evaluating solutions on paper can result in significant costs in terms of time and subjective interpretation. This study aims to reduce the costs associated with evaluation in traditional exams, especially in cases of large class sizes. When evaluating the performance of the system, it was observed that the system increased exam participation and in-class interaction without affecting the final achievement. Although this study is new for the relevant experimental group and did not show a significant change in achievement, it is

anticipated that in future implementations, students will improve their success rates in the course by engaging in more practice opportunities.

REFERENCES

[1] "MOOC." https://www.mooc.org/ (accessed May 05, 2023).

[2] "coursera." https://www.coursera.org/ (accessed May 05, 2023).

[3] "udemy." https://www.udemy.com/ (accessed May 05, 2023).

[4] "edX." https://www.edx.org/ (accessed May 05, 2023).

[5] "JFLAP." Accessed: May 05, 2023. [Online]. Available: https://www.jflap.org/

[6] "Logisim." Accessed: May 05, 2023. [Online]. Available: http://www.cburch.com/logisim/

[7] H. Chen, Y. Wen, and J. Jin, "Computer-aided teaching and learning of basic elementary functions," *Heliyon*, p. e15987, May 2023, doi: 10.1016/J.HELIYON.2023.E15987.

[8] C. C. Cingi, "Computer Aided Education," *Procedia Soc Behav Sci*, vol. 103, pp. 220–229, Nov. 2013, doi: 10.1016/J.SBSPRO.2013.10.329.

[9] C. S. Koong, H. Y. Tsai, Y. Y. Hsu, and Y. C. Chen, "The Learning Effectiveness Analysis of JAVA Programming with Automatic Grading System," *Proceedings - International Computer Software and Applications Conference*, vol. 2, pp. 99–104, Jun. 2018, doi: 10.1109/COMPSAC.2018.10210.

[10] Z. H. Amur, Y. K. Hooi, and G. M. Soomro, "Automatic Short Answer Grading (ASAG) using Attention-Based Deep Learning MODEL," *2022 International Conference on Digital Transformation and Intelligence, ICDI 2022 - Proceedings*, pp. 1–7, 2022, doi: 10.1109/ICDI57181.2022.10007187.

[11] A. N. Oktaviani, M. Z. Alief, L. Santiar, P. D. Purnamasari, and A. A. P. Ratna, "Automatic Essay Grading System for Japanese Language Exam using CNN-LSTM," *17th International Conference on Quality in Research, QIR 2021: International Symposium on Electrical and Computer Engineering*, pp. 164–169, 2021, doi: 10.1109/QIR54354.2021.9716165.

[12] Z. Wang, J. Liu, and R. Dong, "Intelligent Auto-grading System," *Proceedings of 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2018*, pp. 430–435, Apr. 2019, doi: 10.1109/CCIS.2018.8691244.

[13] S. S. Ibrahim, E. F. Elfakharany, and E. Hamed, "Improved Automated Essay Grading System Via Natural Language Processing and Deep Learning," *8th International Conference on Engineering and Emerging Technologies, ICEET 2022*, 2022, doi: 10.1109/ICEET56468.2022.10007407.

[14] S. Mehta, C. Raman, N. Ayer, and S. Sahasrabudhe, "Auto-Grading for 3D Modeling Assignments in MOOCs," *Proceedings - IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, pp. 51–53, Aug. 2018, doi: 10.1109/ICALT.2018.00012.

[15] V. S. Shekhar, A. Agarwalla, A. Agarwal, B. Nitish, and V. Kumar, "Enhancing JFLAP with automata construction problems and automated feedback," *2014 7th International Conference on Contemporary Computing, IC3 2014*, pp. 19–23, 2014, doi: 10.1109/IC3.2014.6897141.

[16] V. Shekhar, A. Prabhu, K. Puranik, L. Antin, and V. Kumar, "JFLAP extensions for instructors and students," *Proceedings - IEEE 6th International Conference on Technology for Education, T4E 2014*, pp. 140–143, Jan. 2014, doi: 10.1109/T4E.2014.22.

[17] M. Mohammed, C. A. Shaffer, and S. H. Rodger, "Teaching Formal Languages with Visualizations and Auto-Graded Exercises," *SIGCSE 2021 - Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pp. 569–575, Mar. 2021, doi: 10.1145/3408877.3432398.

[18] A. Kumar, A. Walter, and P. Manolios, "Automated Grading of Automata with ACL2s," *Electron Proc Theor Comput Sci*, vol. 375, pp. 77–91, Mar. 2023, doi: 10.4204/EPTCS.375.7.

[19] Anil Maheshwari and Michiel Smid, *Introduction to Theory of Computation*. Ottawa, 2019.

[20] M. Sipser, "Introduction to the Theory of Computation," *ACM SIGACT News*, vol. 27, no. 1, 1996, doi: 10.1145/230514.571645.

[21] "JFLAP Source Code." https://github.com/citiususc/jflap-lib (accessed May 05, 2023).