OCaml-Flat on the Ocsigen Framework Extended Abstract*

Rita Macedo

NOVA School of Science and Technology, Lisbon, Portugal
rp.macedo@campus.fct.unl.pt

1 Introduction

Given the mathematical and formal character of the topics covered in subjects such as Languages and Automata, its teaching and learning processes are demanding and challenging. Several studies have confirmed these difficulties and evaluated the use of some applications [13, 18, 16, 14]. It is important to support students' autonomous work with interactive tools that allow them to experiment with examples and solve exercise. There are a lot of applications that aim to overcome the difficulties mentioned by using various solutions. While some are quite complete, they are *Desktop* applications and not available everywhere. Others can be used via a web *browser*, although they are underdeveloped.

The objective of this project is to develop an application serving as an experimental lab on Formal Languages and Automata Theory, FLAT, available through a browser. The idea is to create a tool that represents and animates graphically classical mechanisms and algorithms. It is also intended that this tool should be adapted to the course of Computational Theory [17] taught at NOVA School of Science and Technology and that it includes not only exercises evaluated automatically and providing feedback, but also allowing students to create their own exercises.

The application is being developed in *Ocsigen* Framework, which allows the creation of web application written in *OCaml*. By capitalizing of the features of *OCaml*, it is possible to obtain fully functional and less error-prone web pages. *Ocsigen* also facilitates the creation of web tools, as it allows to write client-side and server-side code in the same source file using the same language, thus facilitating system programming.

2 Related Work

The development of tools related to FLAT has been done since the beginning of the 60's [12]. The article *Fifty Years of Automata Simulation: A Review* [12] also argues that although there are already many tools, the scientific community continues to produce new ones because each one has its own principles and often new uses. In addition, each tool is influenced by the development tools currently available.

It is important to highlight the most significant applications: *JFlap* [7], because it is probably the most complete tool available; *Automaton Simulator*[3] for being a tool that allows the study of FA, verifying the acceptance of sentences; *FSM Simulator, Regular Expression Gym and FSM2Regex* [5, 9, 6] or also being web applications and allowing the study of FA and RE and their conversions; *Automata Tutor v2.0* [1] because it has an evaluation and feedback system;

*Work partially supported by the Tezos Foundation through the FACTOR project (http://www-ctp.di.fct.unl.pt/FACTOR/) and by national funds through FCT - Foundation for Science and Technology, I.P., within the scope of NOVA LINCS through the project UID/CEC/04516/2019.

and *Automate* [2] because it is a tool with similar objectives as this project and it is being developed at the same time.

Each application has their own focus and emphasizes different functionalities - most of them not allowing going backwards in the visualization of algorithms and giving more importance to automata at the expense of regular expressions - but most of them are still incomplete and excluding the *AutomataTutor* none of them allows the lecturers to create exercises in order to use the tool both in class and as a student assessment system.

3 Approach

Since the base library is written in *OCaml*, because it enables the algorithms to be the most similar to the mathematical definitions in the "classical" literature of FLAT [19, 15], we decided, as a proof of concept, to try to create a web tool completely in *OCaml*. The goal is to develop the application using the *Ocsigen* Framework. This framework allows the creation of interactive web systems, entirely written in *OCaml* [8, 10, 11]. One of its great advantages is to compile the client's *OCaml* code for *JavaScript*, which makes it possible to work together with this language and thus use a wide number of libraries available.

Since the *Ocsigen* Framework allows the joint work of *OCaml* with *JavaScript*, it was decided to use *Cytoscape.js* graph Library [4] for the graphical part of drawing automata and the syntax trees. The main reason was that while having a complete and easy to use library to draw the graphs we could focus on other more important issues of the projects, like how to animate, how to show and how to organize all the information and not about the representation of the graphs.

4 Presentation of the Application

Even though the application is still under development, it has already a running version that allows students to work with Finite Automata and Regular Expressions and to create simple exercises. It can be accessed in `http://ctp.di.fct.unl.pt/FACTOR/OFLAT` and the code can be seen in `https://gitlab.com/releaselab/factor/oflat`.

As far as the user interface is concerned, the *OFlat* application is a very simple and clean web page with a menu on the left side and a white box, where the graphics are drawn. In the page the user can import examples from filesystem or create new ones step-by-step (in the case of the Automata).

The automata are represented as is usual in the literature and after they are created we can make conversions, minimizations and generation of words. All of these operations representing the results side by side with the original, allowing the user to compare them and understand the changes made. It is also possible to visualize the acceptance of a word as an animation or step by step (with the possibility of going backward).

The regular expressions are graphically shown not only as a sentence but also as a syntax tree. There is also the option of testing the acceptance of a word by the regular expression, which is, generally, not found in applications but is an important functionality. It is also possible to make conversion into an Automaton and generation of words in the regular expression.

An "exercise" corresponds to the statement of a problem where the user is asked to define a language by means of a finite automaton or a regular expression. The statement is accompanied by a set of unit tests that are used to verify the solution. After an exercise is imported the user can create an automata or regular expression to answer it.

The application, thanks to the framework used has a centralized client-server code and is also structured and extensible which makes it easy to keep adding new mechanisms. It is also prepared to integrate the *Learn OCaml* system[1], thanks to the way the code is organized and the fact that both of the applications are written in *OCaml* and making use of the *Js_of_ocaml* (component of the *Ocsigen Framework*) to create the web application. It will allow teachers to create classroom and assessment exercises.

The web page about the project that includes a video demonstration can be accessed in https://release.di.ubi.pt/factor/oflatocsigen.html.

5 Conclusions

In this project it was proposed to develop a Web application in *OCaml* that allows the students to study different topics of Formal Languages and Automata Theory.

Although a *JavaScript* library was used to display the graphs, we have achieved the proof of concept by showing that it is possible to create a web page almost completely in *OCaml*. We were able to have a complete interaction with the support library, and to maintain all the core programming, like decisions and verifications, in the *OCaml* side of the code, using the *cytoscape.js* library only to show what in *OCaml* is decided to be shown.

In the end, and despite the fact that the application does not cover all the mechanisms taught in the course of Computational Theory, we have built a version which is finalized and being used in class and is also prepared to easily add new mechanisms and improvements in the design.

References

[1] Automata tutor v2.0. http://automatatutor.com/index. Accessed: 2019-02-11.

[2] Automate. https://idea.nguyen.vg/~leon/automata_experiments/index.html. Accessed: 2019-02-11.

[3] Automaton simulator. http://automatonsimulator.com/. Accessed: 2019-02-11.

[4] Cytoscape.js. http://js.cytoscape.org/. Accessed: 2019-06-04.

[5] Fsm simulator. http://ivanzuzak.info/noam/webapps/fsm_simulator/. Accessed: 2019-02-11.

[6] Fsm2regex. http://ivanzuzak.info/noam/webapps/fsm2regex/. Accessed: 2019-02-14.

[7] Jflap. http://www.jflap.org/. Accessed: 2019-02-11.

[8] Ocaml. https://ocaml.org/. Accessed: 2019-02-11.

[9] Regular expressions gym. http://ivanzuzak.info/noam/webapps/regex_simplifier/. Accessed: 2019-02-14.

[10] Vincent Balat. Ocsigen: Typing web interaction with objective caml. *Proceedings of the ACM SIGPLAN 2006 Workshop on ML*, pages 84–94, 09 2006.

[11] Vincent Balat, Jérôme Vouillon, and Boris Yakobowski. Experience report: Ocsigen, a web programming framework. *Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ICFP*, 44:311–316, 09 2009.

[12] Pinaki Chakraborty, P C. Saxena, and C.P. Katti. Fifty years of automata simulation: A review. *ACM Inroads*, 2, 12 2011.

[13] Joshua Cogliati, Frances W. Goosey, Michael T. Grinder, Bradley A. Pascoe, Rockford Ross, and Cheston J. Williams. Realizing the promise of visualization in the theory of computing. *ACM Journal of Educational Resources in Computing*, 5:1–17, 06 2005.

[1] http://learn-ocaml.hackojo.org/

[14] Loris D'antoni, Dileep Kini, Rajeev Alur, Sumit Gulwani, Mahesh Viswanathan, and Björn Hartmann. How can automatic feedback help students construct automata? *ACM Trans. Comput.-Hum. Interact.*, 22(2):9:1–9:24, mar 2015.

[15] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the theory of computation, 2nd Edition*. Prentice Hall, 1998.

[16] N Pillay. Learning difficulties experienced by students in a course on formal languages and automata theory. *SIGCSE Bulletin*, 41:48–52, 01 2009.

[17] António Ravara. Lecture notes in computational theory, March 2019. The notes are currently unavailable to the general public, only for students with authentication.

[18] Ian Sanders, Colin Pilkington, and Wynand Van Staden. Errors made by students when designing finite automata. *SACLA'15 - Southern African Computer Lecturers' Association*, 2015.

[19] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.