

Language Emulator, a helpful toolkit in the learning process of Computer Theory

Luiz Filipe M. Vieira, Marcos Augusto M. Vieira, Newton J. Vieira
Computer Science Departament
Federal University of Minas Gerais
{lfvieira, mmvieira, nvieira}@dcc.ufmg.br

Abstract

During a master degree class, we developed a toolkit to help students to understand the concepts of Automata Theory. It was denominated Language Emulator and has been used by undergraduate students to help them to learn the ideas of Automata Theory. Language emulator is a software developed in Java that allows the manipulation of regular expressions, regular grammar, deterministic finite automata, non-deterministic finite automata, non-deterministic finite automata with lambda transitions, Moore and Mealy machines. Using this software tool was of great acceptance among students, leading to over 95% of approval in a recent opinion research.

Categories & Subject Descriptors

K.3.2 *Computers and Education*: Computer and Information Science Education Computer science education.

General Terms

Experimentation, Languages, Theory.

Keywords:

Computer Theory, tool, teaching.

1 Introduction

The course of computer theory is traditionally taught without any software, being limited to tasks that do not require a computer. All the exercises involve the use of paper and pencil and those can be inconvenient on those course. One possible solution would be to have some exercises that required the use of a software toolkit. As shown by Rodger [8], there are two main reasons for students to have more difficulties with disciplines in Computing Theory. The first reason

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCSE'03, February 19-23, 2003, Reno, Nevada, USA.
Copyright 2003 ACM 1-58113-648-X/03/0002...\$5.00

is the fact that they require more background in mathematics than most of the computer science disciplines do. The second is the fact that students do not receive a feedback when they are working with paper and pencil exercises. Those effects can be reduced by the use of a toolkit, such as Language Emulator, that allows students to interact with Computer Theory and with automates. In this paper, we describe the related work in Section 2, the functionalities presented at Language Emulator in Section 3, the graphic interface in Section sec:interface. Section 5 describes how can this toolkit be used in a classroom and a case study. Finally, Section 6 brings conclusion and future work.

2 Related works

In Computer Science, for many disciplines and subjects, there are toolkits that help students in the learning process. Many tools, including [5], have been developed to create animations of algorithms and those can be used for teaching.

JFLAP [8] (Java Formal Languages and Automata Package) is an excellent tool that can be used in the discipline Fundamentals of Computer Theory, more specifically, in automata theory. However, it is not internationalized, it only accepts English, and it does not support transition among the various types of automata that it works with.

Deus Ex Machine [1] was developed by Nick Savoiu e it comprehend simulations of seven computation models. It was design to be used along with the book “Models of Computation and Formal Languages” de R. Gregory Taylor. Nowadays, there are only versions available for the Windows platform.

Turing’s World [3] is a graphical tool that allows to specify Turing machines by drawing the state diagram and their transitions. The user can visualize, graphically, the operations on the Turing Machine.

Webworks Laboratory [4] presents a set of projects in Fundamentals of Computer Theory. Its main highlight is the “Animating the Theory of Computing”, a tool developed to allow students to learning interacting with animations of finite automates.

Ganimal [2] is a project of software for compilers learning that generates animations of abstracts machines. A machine can be chosen according to the type of language: imperative, functional or a stack.

The Java Computability Toolkit [7] is a graphic environment to create and simulate deterministic finite automata(DFA)

and non-deterministic(NFA) finite automata and Turing Machines. It was developed in Java, and it allows to minimize the DFA and to convert a NFA into a DFA.

Table 1 summaries the related work. Language Emulator is the only work that supports internationalization and auto-correcting functions(as we explain in Section 5).

3 Tool

The Language Emulator was developed to work with the most usual forms of regular languages. The seven formats supported are:

- Regular Expression (RE).
- Regular Grammar (RG).
- Deterministic Finite Automata (DFA).
- Non-deterministic Finite Automata (NFA).
- Non-deterministic Finite Automata with lambda transitions (NFA λ).
- Moore Machine.
- Mealy Machine.

And additionally the following functionalities:

- Minimize DFA.
- Transform from any format supported by Language Emulator into an equivalent one.
- Give all the words of size $\leq n$ that belongs to a RG.
- Determine if a given word belong to the language of a RG.
- Determine if a word belong to the language correspondent to a DFA.
- Determine if a word belong to the language correspondent to a RE.
- Transform a Mealy machine into a Moore one.
- Transform a Moore machine into a Mealy one.
- Give the output of a Moore machine for a given input.
- Determine if a word belong to the a Mealy or Moore machine.
- Union, intersection and difference of two DFAs.
- Complement of a DFA.
- Union of the differences of two DFAs. $A_F = (A_1 - A_2) \cup (A_2 - A_1)$
- Save and Load a DFA.
- Export a DFA to a file in the format accepted by the GraphViz [6] tool. GraphViz receives a textual description of a graph and generates a visual representation of this.

Figure 1 shows the transformations supported by the Language Emulator. The toolkit shows also the intermediary steps of those algorithms, printing the values at each phase of execution.

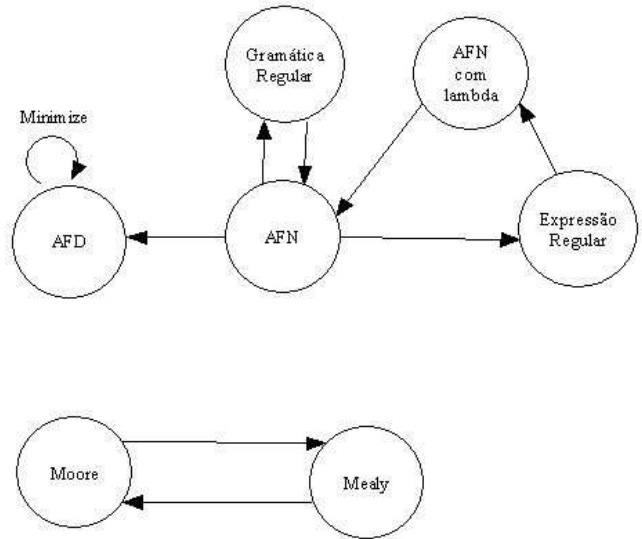


Figure 1: Transformation among the formats of regular language.

4 Interface

The programming language chosen to create the toolkit for automata theory was Java, just because it works in any platform that supports the Java Virtual Machine, what is common on educational environments. It was designed using Java Swing, the Interface standard in Java platform. Initially, the toolkit was designed in Portuguese, but it is possible to internationalize it, changing for the desired language, as, for example, English.

There are seven tabs that the user can chose to construct any of the seven formats supported by Language Emulator: DFA, NFA, NFA λ , RG, RE, Moore and Mealy machines. For each format exist an appropriate interface. This can be visualized when the user selects the tab corresponding to the format's name. It allows the user to access the functionalities presented in the format. For example, the DFAs can be minimized or generate the regular expression that recognizes the same language the DFA does. Those two functions can be accessed by the user throw the buttons Minimize and RE presented in the DFA tab, which is shown in Figure 2.

Figure 3 is a screen shot of Language Emulator that shows the presence of ToolTips. They are presented in the toolkit, what turns it more intuitive to use.

Figure 4 shows a regular grammar being constructed. The user introduces initially the non-terminals, the alphabet and then the initial non-terminal, agreeing with the definition of a regular grammar. By clicking on button OK, the user can initiates the process of completing the table where the rules are inserted. By clicking on Insert Rules, the grammar is built. It is possible to see if a given word belongs or not to the regular language equivalent to the language recognize by the RG, or to generates the equivalent NFA.

With this NFA, it is possible to discover if a word is recognize by the automata, modify it, inserting new symbols on its alphabet or changing its initials and/or finals states. It is also possible to generate a regular expression that represents the equivalent language accepted by the automata

Software	Int.	Aut.	DFA	NFA	NFA λ	RG	RE	Mealy	Moore	Pushd.	Tur. M.	Plat.	Av.
Language E.	✓	✓	✓	✓	✓	✓	✓	✓	✓			all	free
JFLAP			✓	✓	✓	✓				✓	✓	all	free
Deus Ex-M.			✓	✓	✓					✓		windows	free
Turing's W.											✓	macintosh	com.
Webworks			✓	✓	✓	✓						all	free
Ganival			✓									all	free
Java C. Kit			✓	✓	✓						✓	all	free

Table 1: Comparative Table.

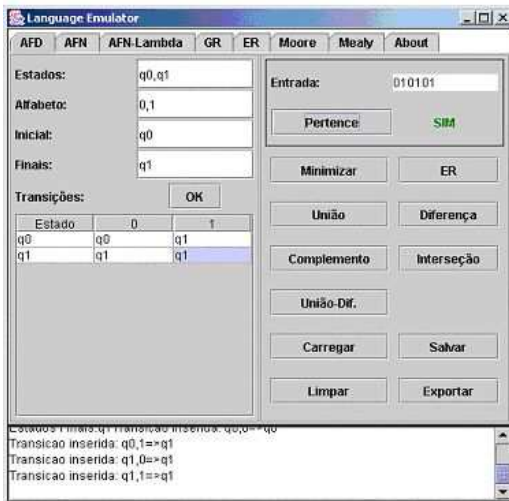


Figure 2: Deterministic Finite Automata at Language Emulator.

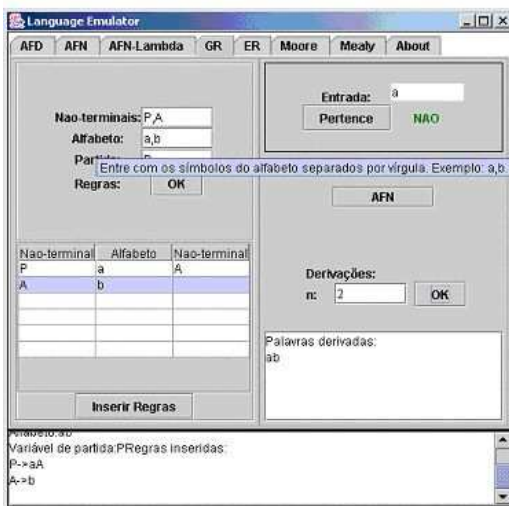


Figure 3: ToolTips are presented in Language Emulator

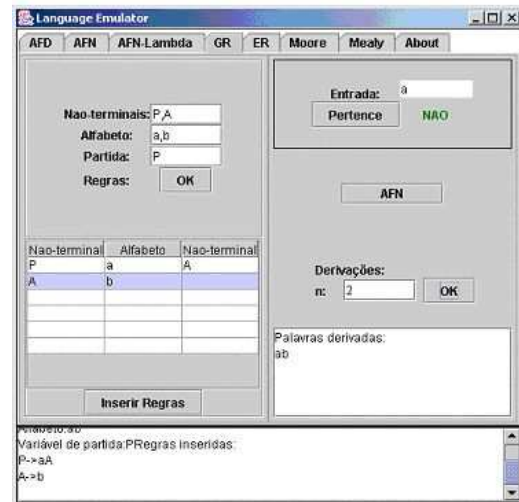


Figure 4: A Regular Grammar in Language Emulator

and, inclusive, generates a grammar or even a DFA.

In Language Emulator, the functionalities that allow the manipulation of a DFA are describes in the Section 3. It is possible to modify any automata in any of its components (states, alphabet, initial state, final state and transitions). The DFA can be minimized or it can generate the equivalent regular expression.

The Language Emulator can be used to verify if a word belong or not to a language equivalent to a regular expression and/or generate the NFA. Both of them can be edited by the user.

In the toolkit, a representation in the format of a NFA λ presents almost the same functionalities that a representation in the DFA format. It is possible to discover if a word is recognize by the automata, modify it and it can be generated the equivalent NFA.

5 Teaching

Along with the theory, some practical exercises can be assigned. As each part of the theory is taught, such as DFA, a task could be assigned involving the use of the toolkit. This model turns the subject easier to learn.

5.1 Case Study - UFMG

During the first semester of 2002, in the discipline of Fundamentals Computer Theory at UFMG (Federal University of Minas Gerais) the toolkit was used. Following the suggested model above, after had been taught NFA λ , DFA and RE,

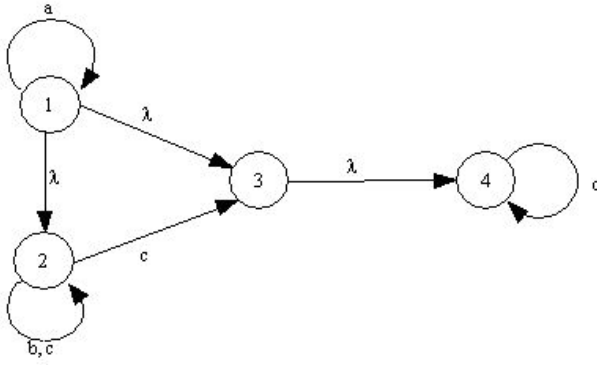


Figure 5: Automata's Exercise

it was proposed the following exercise:

Example of exercise

Considering the NFA λ in Figure 5:

- Construct, using paper, the transition table.
- Construct, using the toolkit, the NFA λ .
- Construct, using paper, the equivalent NFA.
- Using the toolkit, convert the NFA λ into a NFA.
- Construct, using paper, the equivalent DFA.
- Using the toolkit, convert the NFA into a DFA.
- Minimize, using paper, the DFA.
- Using the toolkit, minimize the DFA.
- Construct, using paper, the regular expression that denotes the language recognize by the automata.
- Using the toolkit, convert the DFA into a RE.

Applying the exercise as homework, the students were able to correct their own exercises and learn where they were making their mistakes.

5.2 Helping the teacher and students

The toolkit beyond being used by the students, could also be used by the teacher. Let L_1 be the language recognized by automata A_1 described by a student. And let L_2 be the language recognized by the correct automata A_2 .

The toolkit allows the computation of the resulting regular language from a union, intersection or difference between two regular languages. Furthermore, it was embedded into the toolkit the operation *union of the difference* of two regular languages. This operation is very useful, conform we show next.

Using the proper toolkit, when we select the button Union of the differences in the tab of DFAs, we can calculate $A_3 = (A_1 - A_2) \cup (A_2 - A_1)$.

The language recognized by automata A_3 will be empty if and only if the automata A_1 is equal A_2 , in other words, if it

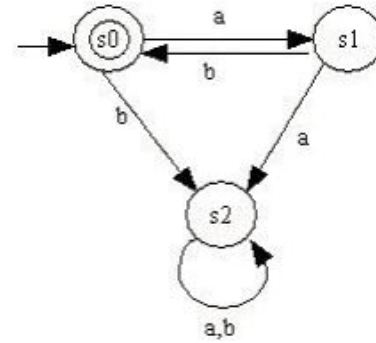


Figure 6: Correct DFA that recognize L_2

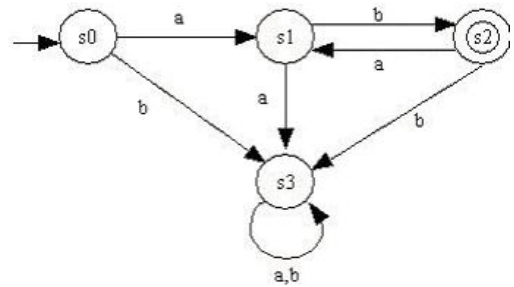


Figure 7: Student's DFA

is correct. And, generating a word from A_3 it is possible to find an example of a word that shows that the students automata is incorrect. Example: Let the language L_2 be equal $\{ab\}^*$. Figure 6 shows the correct automata A_2 .

Suppose the automata A_1 from the student is the one shown in Figure 7:

The automata A_1 recognizes $ab, abab, ababab, \dots, (ab)^n$. However, it is not correct. One fast way to show this is to find a counter-example. In our case, the automata A_1 does not recognize λ and therefore, does not recognize the language L_2 . Using the function difference among two DFAs from the toolkit, we calculate $A_2 - A_1$ to find the words that are not correctly recognized by automata A_1 .

The automata in Figure 8 recognizes only λ . We also calculated $A_1 - A_2$ to find the words that are incorrectly recognize by automata A_1 .

The automata in Figure 9 recognizes the empty set. Ob-

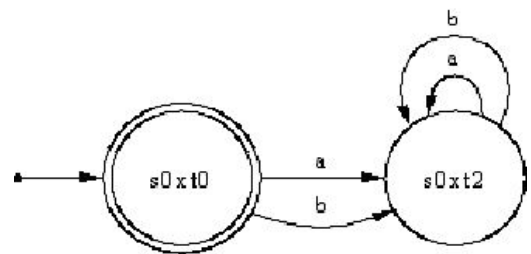


Figure 8: Resulting automata from $A_2 - A_1$

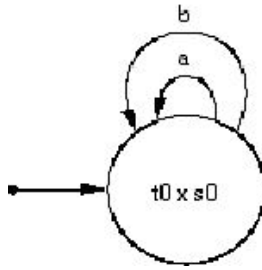


Figure 9: Resulting automata from $A_1 - A_2$

serve that there is no final state. Figure 8 and Figure 9 were generated with GraphViz [6] after using the export function from Language Emulator. Another way to verify the correctness of the student's automata would be to generate the equivalent regular expression and verify if it represents the same language that the correct automata does. In our example, the Language Emulator returned $ER_1 = (ab(ab)^*)$ for the student's automata. Observe that λ does not belong to the language equivalent to ER_1 .

5.3 Toolkit Evaluation

As it was the first time the Language Emulator was being used in a classroom, the students of the discipline participate in the process of evaluation of the toolkit.

Almost 95% of the students answered that the use of Language Emulator helped them in the process of learning Automata Theory. They also give some suggestions as future work that can be used to improved the toolkit and its capacity to help in the teaching process. One of the main advantages cited by the students was the fact that they could solve the exercise using paper and pencil and after that, use the Language Emulator to verify if they did it right.

6 Conclusion

The Language Emulator has been used in the process of teaching students the automata theory. It is possible for students to interact with the formats of regular languages and receive an immediate feedback. The use of a toolkit in a computer science course had a high level of approving, agreeing with the finality of this toolkit. The students liked the fact that they could verify if their work done in paper was correct.

The toolkit is in an earlier stage and it can be improved. Future works include the drawing of states and transitions, using a graphical input, and the research of new techniques that facilitate the understanding of the automata theory by the student. One such topic of research is to show, using the least number of modifications, how to transform the student's automata from incorrect to correct.

Acknowledgment

This work was partially supported by CNPq.

References

- [1] Deus ex machine, <http://www.ics.uci.edu/savoiu/dem/>. Web site.
- [2] Ganimal, <http://rw4.cs.uni-sb.de/ganimal/>. Web site.
- [3] Turing's world, <http://www-csli.stanford.edu/hp/logic-software.html>. Web site.
- [4] Webworks laboratory, <http://www.cs.montana.edu/webworks>. Web site.
- [5] Gloor", P. Aace - algorithm animation for computer science education. In *Workshop on Visual Languages* (1992), pp. 25–31.
- [6] Koutsofios, E., and North.", S. C. Drawing graphs with dot. In *"AT&T Bell Laboratories"* (Murray Hill NJ, U.S.A., October 1993).
- [7] "Robinson, M.B., H. J. N. J. D. A. A java-based tool for reasoning about models of computation through simulating finite automata and turing machines. In *30th Annual ACM SIGCSE Symposium (Special Interest Group on Computer Science Education)* (New Orleans, Louisiana, March 1999).
- [8] Rodger, S. Using JFLAP to interact with theorems in automata theory. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)* 31 (1999), 336–340.