A WEB-BASED E-LEARNING TOOL FOR THEORETICAL COMPUTER SCIENCE

E. del Acebo, I. Boada, J.Poch, F. Prados, J. Soler
Departament d'Informàtica I Matemàtica Aplicada
Universitat de Girona, Spain.
esteve.acebo, imma.boada, jordi.poch, ferran.prados, josep.soler@udg.edu

Abstract

The paper presents a web-based tool designed to automatically pose and correct problems about finite state machines, concretely deterministic and non-deterministic finite automata. The tool is integrated into a more general e-learning environment, the ACME framework, a web-based problem-solving environment developed at the University of Girona. The tool only requires a web browser and is being used in our university with very promising results.

Keywords
e-learning, ACME, automata, finite state machines.

1. INTRODUCTION

While widely acknowledged as one of the basic and more important subjects taught in any Computer Science curriculum, Theoretical Computer Science has such a highly theoretical and conceptual nature that students often have difficulties in its assimilation. In this paper, we present a web-based tool designed to automatically pose and correct problems about the part of Theoretical Computer Science concerning finite state machines (FSM), concretely deterministic and non-deterministic finite automata. This FSM tool is integrated into a more general e-learning environment, the ACME framework, a web-based problem-solving environment developed at the University of Girona with the aim of reinforcing teaching and learning processes in technical/engineering degrees. ACME can assign personalized exercises to the students and allows them to try possible solutions, providing on-line feedback about their correctness. Additionally, ACME can also collect statistics, useful for the teacher in order to monitor the students' learning process. The finite state machines module presented in this paper has three parts, the first part allows the teacher to enter and edit problems and their corresponding correct solutions. The second part allows the students to design deterministic and not deterministic finite automata through a user-friendly graphical interface. Finally, the third part of the module assesses the correction of the proposed solutions and gives advice, when needed.

The structure of the article is the following: Section 2 presents some previous work about existing platforms, environments and utilities for Theoretical Computer Science e-learning, section 2 introduces ACME, the e-learning framework developed at the University of Girona. Section 3 gives a description of the new e-learning FSM tool and section 4 describes the experiences collected in the initial period of use of the tool. Finally section 5 presents the conclusions and examines the possible future work.

2. PREVIOUS WORK

There exist a considerable amount of systems and software packages aimed to the support of the teaching of Theoretical Computer Science, specially subjects as finite state machines, regular expressions and formal grammars. **AMORE** [4] is a program for the computation of finite automata, syntactic monoids of regular languages, and regular expressions. **JFLAP** [5, 10] is a Java tool for designing, and simulating several variations of finite automata, pushdown automata, and Turing machines. **Grail** [9] is a C++ toolkit for finite-state machines and regular expressions. **Turing's World** [1] allows one to experiment with many different forms of Turing machines in a graphical format. **Automata** [14] is a *Mathematica* based package that manipulates finite state machines and their syntactic semigroups. A number of operations on one-dimensional cellular automata are also available. **"Deus Ex Machina**" [15] allows the users to experiment with Turing machines, finite automata, push-down automata and other types of automata. ***FoLa*** [16] is an interactive Computer Algebra software for teaching and helping students in the study of the foundations of Computer Science. ***FoLa*** is distributed as a single *Maple* package. Besides supporting most of the basic operations on automata, grammars, regular expressions and languages, ***FoLa*** also supports two- and three-dimensional plotting and animations, which can be used for simulating the work of theoretical models such as finite state automata, parsing trees or the more complicated models such as two- and three-dimensional tape Turing machines. Finally, perhaps the most ambitious in scope, the **Exorciser** system [17] is an application framework that supports the implementation and distribution of interactive self-grading exercises on quiet a big number of subjects related to Theoretical Computer Science, within a visual framework that offers a consistent interface.

3. THE ACME E-LEARNING PLATFORM

ACME is an e-learning platform to improve both teaching and learning at the technical/engineering degree programs. It has been under active development in the University of Girona since 1998. The platform was conceived to integrate new types of problems with minimal modifications [6]. Figure 1 illustrates the main modules of the ACME platform. Below we describe them and their functions:

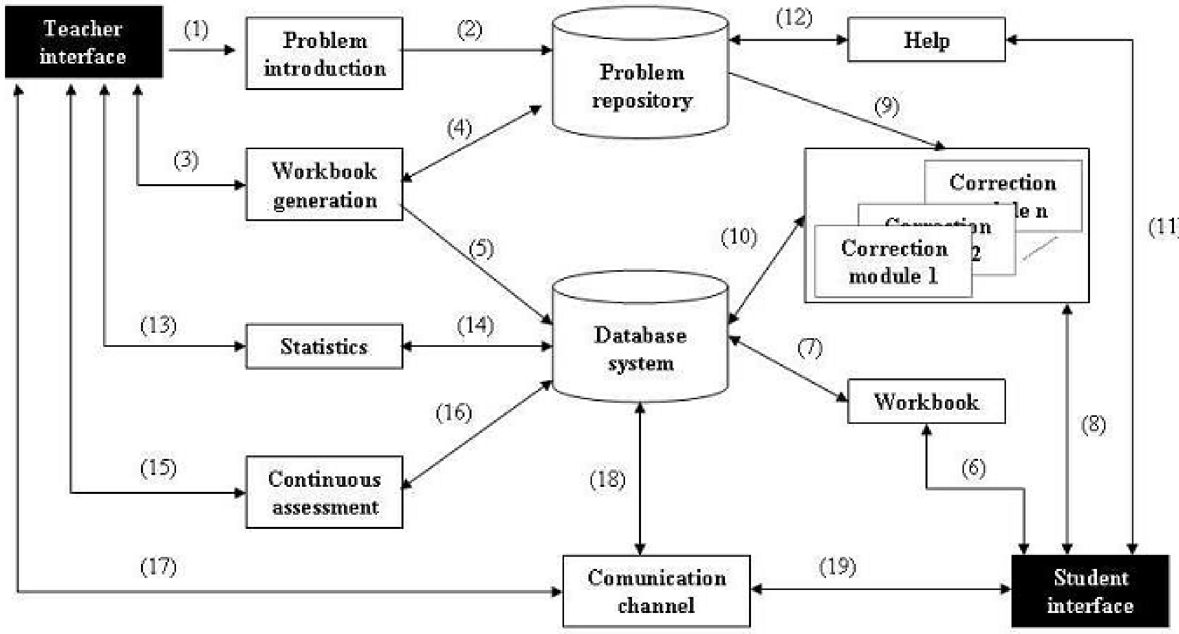


Fig. 1. Main modules of the ACME e-learning platform

- There is a *repository* to maintain all the problems teachers enter into the system (see (1) and (2) of Figure 1), allowing them to share sets of material.
- The teacher selects a set of problems related to each topic from the repository. The system uses these exercises and the *workbook generation module* to automatically generate a personalized workbook for each student. The workbook is composed of exercises grouped into topics, which can be added at any time (see (3), (4) and (5)).
- Students have to solve the problems, using the corresponding *interface*, and send the solutions to the system before a fixed deadline. Students access the workbook (see (6) and (7)) and send solutions to the problems (see (8)).
- The *correction module* has a specific component for each type of problem supported by the platform. Amongst them are mathematical, computer programming and database problems. For all these cases, this module provides an environment for on-line correction, giving immediate feedback and, in case of errors, advice about how to correct them. All the student solutions are stored in the *system database* (see (8), (9) and (10)).
- If students have difficulties, solving any of the problems, the *help module* provides a set of problems with the information required to obtain the correct solution (see (11) and (12)).
- All kinds of *statistics* can be obtained from the information recorded in the database system. For instance, the number of solutions sent, the number of problems solved, etc. (see (13) and (14)).

- From the database the *continuous assessment module* collects quantitative data about student work such as the number of errors and types of errors, the time taken to complete the problem, etc. Student progress through the personal exercise books provides the basis for the continuous assessment of skills and gives valuable information about the difficulties experienced. This information can be used to guide the students through the most difficult topics (see (15) and (16)).
- A virtual communication channel is established between the teacher of a group and all the students in the group. This channel can be used by the teacher as a virtual tutoring system (see (17), (18) and (19)).

The ACME environment supports different types of users: students, teachers and administrators. Interface windows are designed specifically for each user type and each subject.

Until the present day a number of tools have been developed for the ACME platform corresponding to as different subjects as introductory programming, basic maths, relational algebra, entity-relationship modelling, database normalization and SQL [2, 6, 7, 8, 11, 12, 13]. The tool presented in this article is the first attempt to incorporate a subject related to Theoretical Computer Science.

4. THE FINITE STATE MACHINES TOOL

In this section we present the proposed finite state machines tool. First, we introduce the main design decisions. Then, we describe the tool from a technical perspective, with information about the main modules in it.

4.1 Design decisions

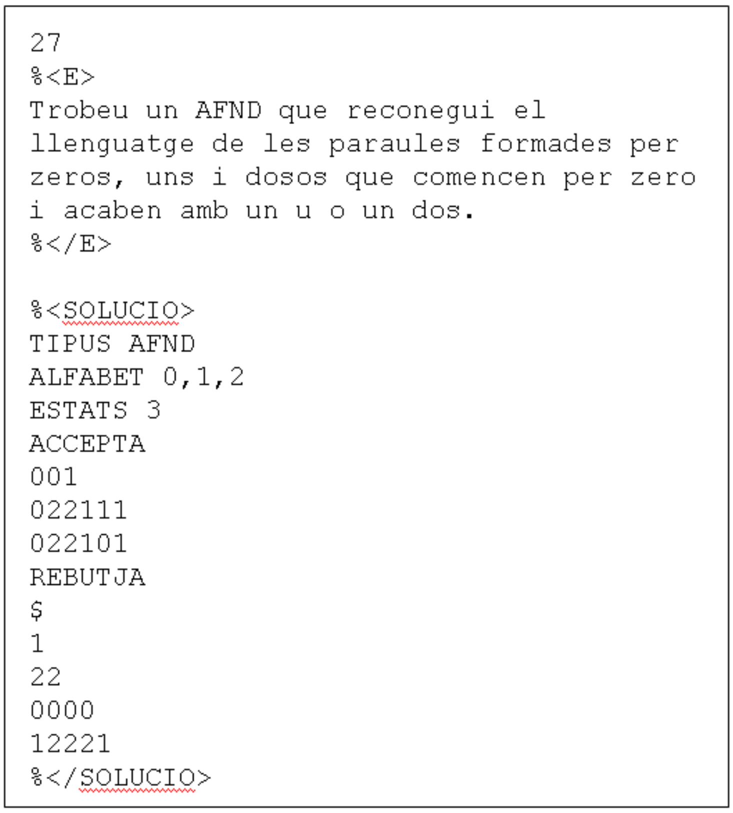
Our aim is to develop an environment that provides relational algebra learning and teaching support for both students and teachers. The main requirements and decisions we take into account for the design of this web environment are:

- *Easy access*. A first requirement of the tool is that it has to be easily accessible for teachers and students. No special installation is required, only a web browser.
- *Finite state machines training environment*. The system has to provide different types of finite state machines exercises and a graphical environment to facilitate students' work as well as the introduction of new exercises by the teacher.
- *Online correction*. When students enter solutions to exercises, the system has to provide immediate correction as well as information about errors. Students have to have the chance to enter new solutions.
- *Record of student work*. The system has to record all student work, i.e. all attempts entered for a query until the correct one is obtained.
- *Continuous assessment*. The system has to provide teachers with the necessary information and tools to carry out continuous assessment.
- *Enhance the communication between the student and the teacher*. The system has to provide a communication channel that enhances student-teacher contact, whether students ask about doubts or teachers give hints about how to solve queries.

Some of the desired features, such as easy access, record of student work, continuous assessment and the student-teacher communication channel, are provided by the ACME environment. Therefore, to reach our objective we only need to develop the modules that meet the requirements related with finite state machines. These modules are the problem generation strategy, the finite machines student and teacher interface and the correction module. They are all described in the next sections.

4.2 Problem definition

Differently from other tools in the ACME platform, problem definition protocol in the FSM tool is quite straightforward. Each problem is defined in a little text file structured in several parts separated by different tags. We can see an example in figure 2. In the text file we can find, first of all, a number codifying the type of exercise within ACME's problem base. Next to it appears the formulation of the exercise. Optionally, the path to one or more figures can be included in order to complete the problem formulation or facilitate its comprehension by the students. Finally, the text file includes information to be used by the correction module in the correction of the problem. This information includes the kind of automata to be built (deterministic or not deterministic), the characters that can be accepted as symbols of the input alphabet, the number of states the automaton has to have in order to be minimal (optionally) and two lists of strings: a list of strings that must be accepted by the automaton and a list of strings that must be rejected.



```
27
%<E>
Trobeu un AFND que reconegui el
llenguatge de les paraules formades per
zeros, uns i dosos que comencen per zero
i acaben amb un u o un dos.
%</E>

%<SOLUCIO>
TIPUS AFND
ALFABET 0,1,2
ESTATS 3
ACCEPTA
001
022111
022101
REBUTJA
$
1
22
0000
12221
%</SOLUCIO>
```

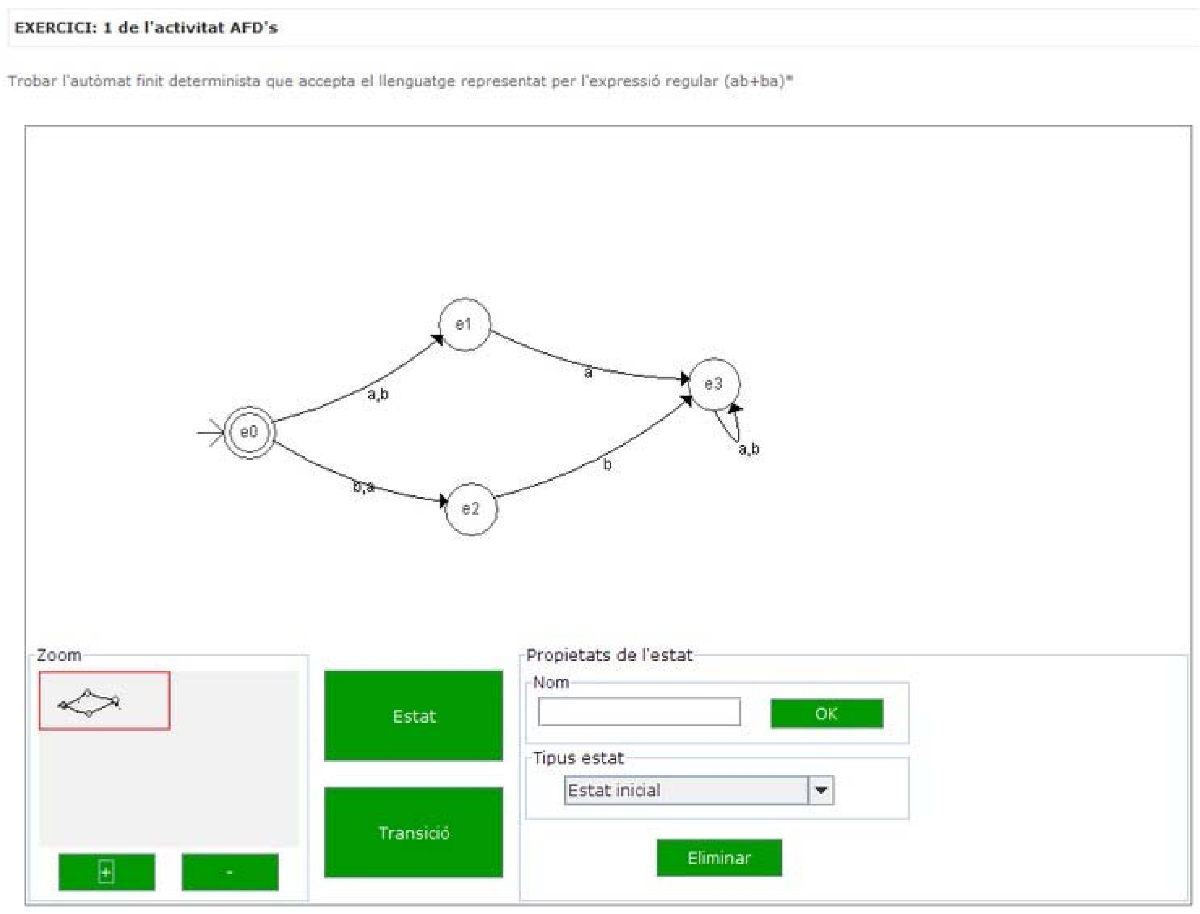
Fig. 2. Example of text file defining an exercise in the finite state machines tool for the ACME platform.

4.3 Students interface

Students enter the ACME environment by using their password. Once in the system a list of subjects appears, amongst them finite state machines. Students select it to access the list of finite state machine exercises assigned to their workbooks and, by selecting one of them, visualize all the information required to solve it.

In figure 3 we can see a capture of part of the window containing the graphical interface used by the students to respond to the exercises drawing the appropriate finite automata. The interface appears after the exercise is selected from the list of exercises available to each student (not every exercise is available to all the students, in general. The ACME platform allows the teacher to restrict one student the access to a class of exercises until she has solved all of the exercises of a simpler class, for example). The wording of the exercise is found in the upper part of the screen and the remaining space is occupied by the drawing area, at the top, and the drawing controls, at the bottom. The interface is quite simple. Students can create, delete and drag-and-drop states and transitions. States can be named and marked as initials or not initials and finals or not finals. It is also a zoom tool in case the automaton grows bigger than the drawing area.

Once the student is satisfied with her proposed solution, she clicks a button and her solution is sent to correction. All the information related to the correction process is shown in a new window. In case of errors, students have the chance to send a new solution. These steps can be repeated as many times as necessary until a correct solution is entered.


EXERCICI: 1 de l'activitat AFD's

Trobar l'autòmat finit determinista que accepta el llenguatge representat per l'expressió regular (ab+ba)*

Fig. 3 Screenshot of the graphical interface of the FSM tool for the ACME framework

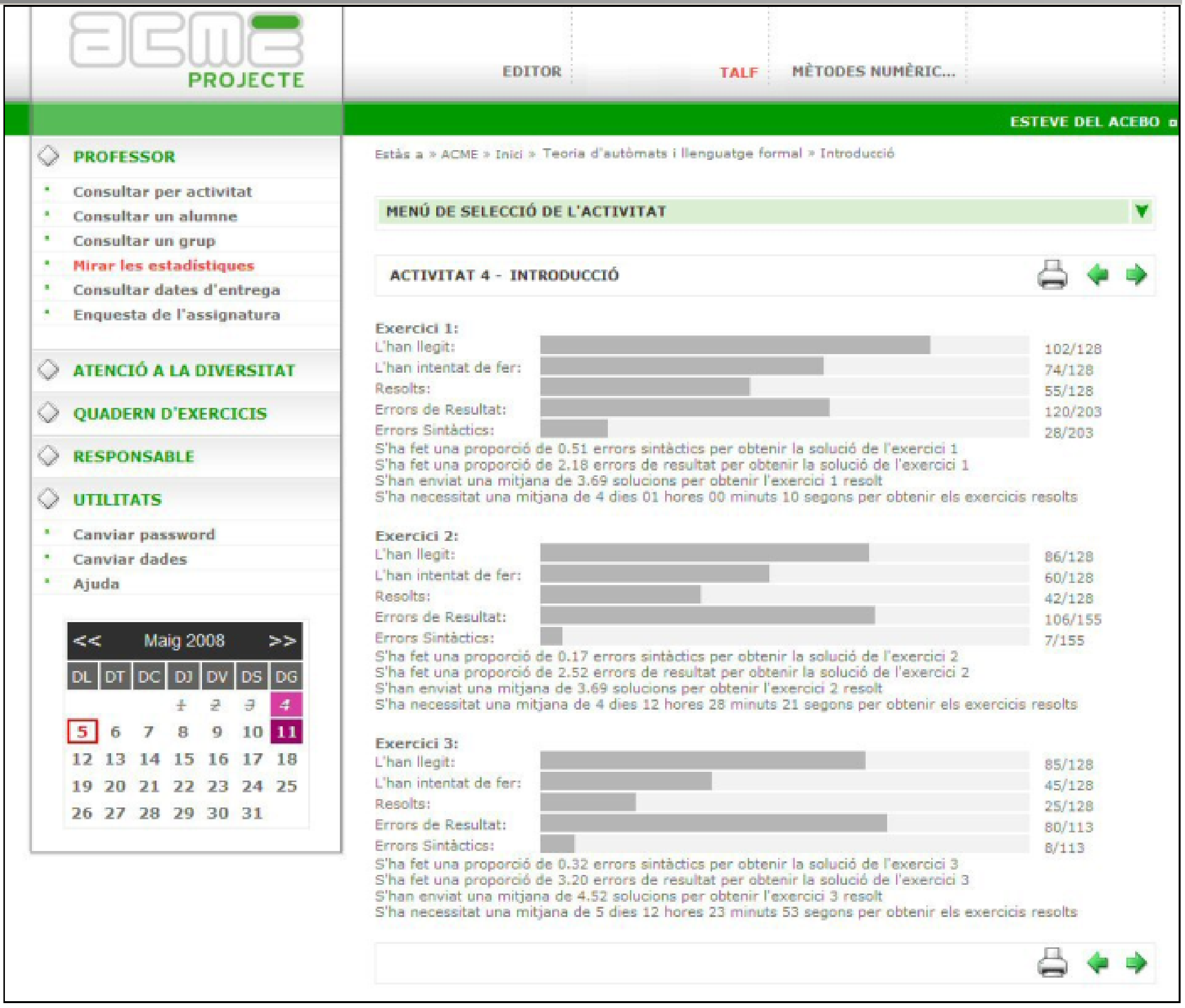
4.4 Correction module

The process of correction of the answers given by the students to the exercises is quite simple and basically involves ensuring the following conditions to be fulfilled:

- The input alphabet is correct.
- The type of automaton (deterministic or not deterministic) is correct.
- The number of states of the automaton is correct (in the case the exercise asked for a minimal automaton).
- The automaton accepts all the strings specified as valid in the text file containing the description of the exercise and rejects the strings all the strings specified as no valid.

The correction module checks these points sequentially, emitting warnings in the form of text messages whenever it detects any error. Examples of such messages are: “Automaton is not deterministic”, “Automaton has too many states” and “Automaton should accept/reject string *w*”.

The correction module also tracks the activity of the students, maintaining a record of, amongst many other data, what exercises each student has attempted to solve, how many tries has she needed to solve it correctly, what kind of exercises have the students more troubles solving them or what kind of errors are the most frequents. This tracking permits the elaboration of statistics (see figure 4) and the personalized tutoring of the students.



Estàs a » ACME » Inici » Teoria d'autòmats i llenguatge formal » Introducció

MENÚ DE SELECCIÓ DE L'ACTIVITAT

ACTIVITAT 4 - INTRODUCCIÓ

Exercici 1:

L'han llegit:	102/128
L'han intentat de fer:	74/128
Resolts:	55/128
Errors de Resultat:	120/203
Errors Sintàctics:	28/203

S'ha fet una proporció de 0.51 errors sintàctics per obtenir la solució de l'exercici 1
S'ha fet una proporció de 2.18 errors de resultat per obtenir la solució de l'exercici 1
S'han enviat una mitjana de 3.69 solucions per obtenir l'exercici 1 resolt
S'ha necessitat una mitjana de 4 dies 01 hores 00 minuts 10 segons per obtenir els exercicis resolts

Exercici 2:

L'han llegit:	86/128
L'han intentat de fer:	60/128
Resolts:	42/128
Errors de Resultat:	106/155
Errors Sintàctics:	7/155

S'ha fet una proporció de 0.17 errors sintàctics per obtenir la solució de l'exercici 2
S'ha fet una proporció de 2.52 errors de resultat per obtenir la solució de l'exercici 2
S'han enviat una mitjana de 3.69 solucions per obtenir l'exercici 2 resolt
S'ha necessitat una mitjana de 4 dies 12 hores 28 minuts 21 segons per obtenir els exercicis resolts

Exercici 3:

L'han llegit:	85/128
L'han intentat de fer:	45/128
Resolts:	25/128
Errors de Resultat:	80/113
Errors Sintàctics:	8/113

S'ha fet una proporció de 0.32 errors sintàctics per obtenir la solució de l'exercici 3
S'ha fet una proporció de 3.20 errors de resultat per obtenir la solució de l'exercici 3
S'han enviat una mitjana de 4.52 solucions per obtenir l'exercici 3 resolt
S'ha necessitat una mitjana de 5 dies 12 hores 23 minuts 53 segons per obtenir els exercicis resolts

Fig. 4 A web page of statistics about the difficulty of several exercises on finite state machines provided by the ACME platform. Several links to other components of the platform can be found to the left.

5. EXPERIMENTAL RESULTS

The proposed FSM tool has been integrated in the ACME environment. It has been evaluated in an introductory Computer Theory course at the University of Girona. During sessions students solve exercises by using the tool. At the end of these sessions they are given a feedback questionnaire. According to this survey, students consider the main advantage of the system to be the immediate feedback provided about the solution proposed to a given exercise. They feel themselves motivated to solve the problems. The possibility of correcting a problem in real time encourages them to work until a correct solution is found. They also consider hat error messages offer valuable information towards the solution to the problem.

Teachers' impressions were very positive. The tool improves upon a classical teaching methodology since it provides a system for the continuous assessment of the student's progress, retrieving statistics on different aspects of the problems, for instance the number of attempts required to solve a problem, the main errors, the main difficulties detected, etc. We can track student work and send messages to them through the communication channel, enhancing the student-teacher relationship at the same time.

6. CONCLUSIONS AND FUTURE WORK

We have presented the FSM tool for the ACME framework. With this tool, the ACME platform adds to its capabilities the ability to support learning and teaching of finite state machines theory and practice. The platform presents several distinctive traits:

- It supports both the learning and the teaching process. The collection of data about the interactions of the students with the tool makes possible to provide students with a personalized learning experience.
- The system has a graphical interface that eases the problem solving process as well as the interaction with the platform.
- The framework provides the student with immediate feedback about her mistakes, difficulties and abilities.
- No other software than a web browser is necessary to use the tool.

The tool has been in use during the past semester in an introductory course on Theoretical Computer Science with very promising results.

We are very conscious of the limitations of the tool in its current state, the main one being the small number of problem types it considers (in fact, only problems whose answer can be given as a finite automaton). Our immediate plans include the expansion of the FSM tool in order to add the capability to pose and correct problems about other diverse subjects as pushdown automata, Turing machines, formal grammars, and regular expressions.

References

[1] Barwise, J. and Etchemedy, J. *Turing's world*, Cambridge University Press, 1993

[2] Boada I., Prados F., Soler J. and Poch J. A teaching/learning support tool for introductory programming courses. *International Conference on Information Technology Based Higher Education and Training*, ITHET (2004) pp. 604-609.

[3] Buchberger, B. *Mathematica: A system for doing mathematics by computer?*,Advances in the Design of Symbolic Computation Systems, Springer, Vienna - NewYork, 1997, pp. 2-20.

[4] Jansen, V., Potthoff, A., *Thomas*, W. and Wermuth, U. A short guide to the AMORE system, *Lehrstuhl für Informatik II*, Universitat Aachen, Aachen, Germany, 1990.

[5] LoSacco, M. and Rodger, S. Flap: A tool for drawing and simulating automata, *EDMEDIA 93, World Conference on Educational Multimedia and Hypermedia*, 1993,pp. 310-317.

[6] Prados F., Boada I., Soler J., Poch J. Automatic Generation and Correction of Technical Exercises. *International Conference on Engineering and Computer Education*: ICECE (2005).

[7] Prados F., Boada I., Soler J., Poch J. An Automatic Correction Tool for relational Database Schemas. *International Conference on Information Technology based higher Education and Training*, ITHET (2005).

[8] Prados F., Boada I., Soler J., Poch J. A Web Based-Tool for Entity-Relationship Modelling. *International Conference on Computational Science and its Applications* ICCSA (2006). LNCS 3980, pp. 364{372.

[9] Raymond, D. and Wood, D. Grail: A c++ library for automata and expressions, *J. Symbolic Computation 11* (1995).

[10] Roger, S., Bressler, B., Finley, T. and Reading, S. Turning automata theory into a hands-on course, *Proceedings of SIGCSE'06*, ACM, March 2006.

[11] Soler J., Boada I., Prados F., Poch J. A web-based problem-solving environment for Database Normalization*. Simposio Internacional de Informatica Educativa*. SIIE (2006).

[12] Soler J., Prados F., Boada I., Poch J. A Web-based tool for teaching and learning SQL. *International Conference on Information Technology Based Higher Education and Training*, ITHET (2006).

[13] Soler J., Poch J., Barrabes E., Juher D. and Ripoll J. A tool for the continuous assessment and improvement of the student's skills in a mathematics course. *Technology of Information and Communication in Education for Engineering and Industry*. TICE (2002), pp. 105{110.

[14] Sutner, K. *Implementing finite state machines, Computational support for discrete mathematics* (N. Dean and G. Shannon, eds.), DIMACS series in discrete mathematics and theoretical computer science, vol. 15, American Mathematical Society, 1992, pp. 347-363.

[15] Taylor, R. *Models of computation and formal languages*, Oxford University Press,New York, 1998.

[16] Tran, Q. 2006. Interactive computer algebra software for teaching and helping students to study foundations of computer science. *J. Comput. Small Coll*. 22, 1 (Oct. 2006), 131-143.

[17] Tscherter V., Lamprecht R., Nievergelt J. Exorciser: Automatic Generation and Interactive Grading of Exercises in theTheory of Computation. Proc. *Fourth International Conf. on New Educational Environments*, 3.1.47-50, Lugano, Switzerland. May 2002.