ISSN: 2473-3857
v3 n4356

Motivating Students through Educational Software Development

Laura Felice
lfelice@exa.unicen.edu.ar

María Virginia Mauco
vmauco@exa.unicen.edu.ar

María Carmen Leonardi
cleonard@exa.unicen.edu.ar

Computer Science Department
Facultad Ciencias Exactas
Universidad Nacional del Centro de la Pcia. De Buenos Aires
Tandil, 7000, Argentina

Abstract

Educational software helps to motivate and improve the teaching / learning process. This is even more important when the software is developed by students for other students. By teaching students to combine in an efficient way the concepts learnt in Logic course and Algorithm Design course, we defined the ´Educational Tools Development´ project. The project has produced many innovative ideas and solutions improving academic performance of students. This paper presents a set of visual, interactive and didactic tools for an introductory logic course developed in the context of the mentioned project. The tools were developed by students of the 2nd year of a Computer Science career as final work of two courses, one that introduces concepts of Logic for Computer Science and another one that teaches the basis of algorithm design techniques, allowing them to integrate and deepen the contents of both.

Keywords: Educational Software, Propositional Logic, First Order Logic, Teaching Resources.

1. INTRODUCTION

In basic programming courses, students learn by providing solutions to traditional problems linked with simple data structure and simple logic. In addition, most of Computer Science curricula include basic courses of Logic. In these courses, students have to solve many exercises to gain practice in formalisms.

The use of didactic tools is a great contribution when it is involved in the teaching/learning process, with no much time for the understanding of the tools. The Undergraduate Degree Program in Systems Engineering includes an introductory course on Propositional Logic and First Order Logic (FOL) (http://ccomp2.alumnos.exa.unicen.edu.ar). At first, the students have to make their exercises manually, which makes it difficult to detect and correct errors. Thus, students consider these

ISSN: 2473-3857
v3 n4356

kinds of courses less attractive and more difficult than others. Using educational tools, the feedback is immediate, so the motivation can help to improve the teaching/learning process. For these reasons, it was considered to build and implement tools which can also help students see incompleteness and inconsistencies in their exercises. These tools must be adjusted to the notation and the methodology taught in the course. Also, the tools must be simple and intuitive in their use.

In addition, the curriculum includes a course of Algorithm Design (http://aydalgor.alumnos.exa.unicen.edu.ar) that gives the students the fundamental concepts of good programming practices. They are Recursive Programming, Computational Complexity, Abstract Data Types, and Algorithm Design Techniques. To make programming techniques more practical and useful to students once they complete their studies, several projects related to real-world problems are offered to them. Therefore, students learn by producing concrete and realistic solutions. These solutions involve the use of important concepts related to good practice of programming such as Algorithm Design techniques and Computational Complexity.

Students may attend courses on Logic and Algorithm Design in the second year of the curriculum. In this context, teachers in charge of the courses proposed the creation of projects developed by students that involve contents from both of the courses. Within this framework, several didactic, visual and interactive tools were developed in order to fit the contents, the notation and the methodology followed in the course of Logic, and that are easy and intuitive in its use (Cicconi & Fernández Cocirio, 2014), (Dahl & Fujii, 2016), (Ferrante, 2009), (Kiehr & Re Medina, 2012), (Maggiori & Gervasoni, 2012), (Ruau & Tosini, 2015), (Santillán Cooper, Horquin & Covelli, 2017). These tools are used as a support in the teaching / learning process, both by teachers and students of the Logic course.

In this paper, the developed tools are described briefly. Each tool allows to experiment with some of the contents learnt in the Logic course. Also, more than a tool was developed for the same topic in order to have a complementary and different approach.

This paper is structured as follows. Section 2 describes briefly the courses involved in this proposal. In Section 3, the team selection of the students is justified. Section 4 presents the description of the main tools. In Section 5 we briefly describe the use experience and the tools development, and finally, some conclusions are presented in Section 6.

2. COURSES INVOLVED

In this section, we briefly describe the main contents of two courses linked to the tools developed. Both courses are included in the first semester of the second year System Engineering curriculum career of Universidad Nacional del Centro in Argentina (www.exa.unicen.edu.ar).

Logic course

This course covers introductory concepts to the Propositional Logic and First Order Logic (Ben-Ari, 2012). The contents of the course are oriented to provide the students the fundamental logic for computer science. They are: syntax, semantic, deduction and demonstration methods, correction results and completeness, satisfiability and validity, formalization of natural language specifications. In addition, the fundamentals for logic application in computer science, such as software verification, logic programming, database, are introduced. In each class, the work with the students is complemented by interactive illustrating with exercises developed by students and guide teachers. There is a tool for most of the topics. These tools were developed by other students that completed the course.

The classes are offered to approximately one hundred undergraduate students (second year) from Computer Science department. All of the students know some computing basics; they are learning programming techniques to improve the quality of programs.

Algorithm Design Course

To understand a problem in computational terms and try to write a program to solve it is the first obstacle students must confront when facing real-world problems. It is common for the students to find it difficult to separate the necessary and unnecessary details of the problem description. Therefore, the first obstacle the students must deal with is to obtain an abstract view of the problem.

The main objective of this course is focused on the introduction of main topics related to software development, such as abstract data

types, component libraries development, and the reusability of the software developed (Cormen, Lieserson & Rivestl, 2009), (Aho & Ullman, 1995).

To make this task efficient, it is necessary to learn good practices in the development of software. It is required that the programmers (students) work with basic design techniques like Divide and Conquer, Dynamic Programming and Greedy Methods and heuristics. For this kind of courses it is essential to create illustrative examples.

For the last years, teachers have promoted the creation of different types of students' work to support these objectives. They include not only solving real-world problems, but also the development of educational tools related with the Logic course. Several students are part of the project called ´Educational Tools Development´ whose main objective is the stimulation of the learning of Logic and the development of tools.

3. STUDENT TEAMS SELECTION

For the past years, Educational Software is being used by traditional courses in many careers of Computer Science. This was a motivation to design this proposal: to stimulate student teams to develop this kind of tools which should follow the Logic course methodology.

Beginning in 2009, student's projects were migrated from a traditional course project to the software development to create tools to assist the learning/teaching of Logic concepts using efficient techniques of Design Algorithm. Students learn the importance of a systematic approach in the process of developing robust tools, how to interact with mentor teachers and how to test systems. Emphasis is placed on developing skills and knowledge in technical areas like the programming language and programming environment. In addition to technical skills, students develop critical thinking, communication, and teamwork skills. By working on this kind of project, students learn the appropriate skills for filling meaningful roles in the whole development.

The students who develop these tools are selected according to their performance in the courses involved in this project. The development of these tools is the first large-scale project they carry out in the career. Each tool has been developed by two or three members in a group with a mentor teacher as a guide. The estimated time of tools development is approximately two months considering groups of two developers.

4. THE TOOLS

All the tools described in this section were developed by students finishing the second year of the career. Each tool involves topics from Logic and Algorithm Design courses. They are didactic, visual and interactive tools that use the same notation, methodology and contents of the Logic course. Thus, the use and the facility to work with them are easier and more natural.

The tools are open source with GNU GPL (GNU, 2017) license. C++ is the language for the implementation, following an Object Oriented design. Computational Complexity is a crucial aspect that students have to take into account. The implementation of each tool is based on this aspect, and different algorithms were analyzed in order to reach good time in the program execution. Qt framework (Blanchette & Summerfield, 2008) is the GUI used for the development of the graphics interface. The interfaces were first implemented in Spanish, as it is the student's mother tongue language, and some of them were later translated to English. The development of these tools was a hard challenge because students have to deal with frameworks they have never used before.

In the following sections we describe some selected tools and their functionality.

SAT and DP Solver

SAT (Boolean Satisfiability Problem) (Santillán Cooper et al, 2017) was developed to determine the satisfiability of a set of clauses of the Propositional Logic. Also, it was developed to show the facilities offered by Propositional Logic to model different problems. It is very important at this point of the career to know how the Propositional Logic can help to solve many problems. This tool allows the users to experiment with two real problems, Graph coloring and N-Queens problem, verifying that both can be solved like a Boolean satisfiability problem (Kelley, 1997). Figure 1 shows one interface window for N-Queens problem.

DPSolver was thought as a didactic and interactive tool that gives support to the learning of Propositional Logic. In particular, it was accomplished with the application of Davis Putnam algorithm to a set of clauses to

ISSN: 2473-3857
v3 n4356

determine its satisfiability. DPSolver allows computing the conjunctive normal form for any formula of the Propositional Logic. Also, it allows working with semantic entailment (Cicconi & Fernández Cocirio, 2014).

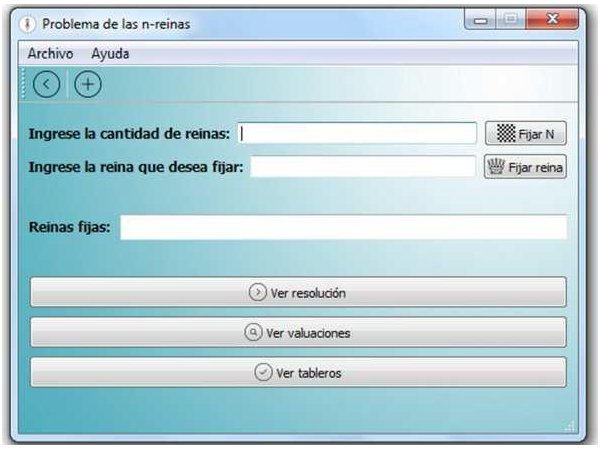

Figure 1. SAT for N-Queens problem

YAT3

YAT3 tool (Yet Another Truth Table Tool) (Ruau & Tosini, 2015) allows working with Refutation Trees, both in Propositional and First Order Logic. In a Logic course, it is common the tree construction from logic formulas to determine valid consequences and formula validity. Thus, the tree construction is made applying a set of rules until each branch of the tree can be classified as close or not.

The tool considers three ways to make the refutation tree: from the user given formula, the refutation tree is computed and shown; from the user given formula, refutation tree is computed one rule at a time; starting from the user given formula, the user itself decides the way to construct the tree, choosing the rule to apply in each step. Thus, this last way is similar to the work that students do in paper and pencil.

FOLST and LogicChess

FOLST (Mauco, Maggiori, Gervasoni, Ferrante & Felice, 2012) and LogicChess (Kiehr & Ré Medina, 2012) are didactic, visual and interactive tools that give support for syntactic and semantic evaluation of FOL formulas in user-defined models over the domains provided by each tool.

FOLST provides the implementation of two frames: Farm and World that allow the definition of different models. The Farm frame (Figure 2) consists of an image of a farm where different animals (pigs, ducks, cows, cocks) in different places (in the forest, on the grass, in the air, in the farmyard) may be added. Each animal has attributes (species, location, is sleeping the action is doing), and predicates allow the formalization of real information in this context. The frame provides eleven unary predicates, such as IsACow(x), IsOnTheGrass(x), and IsSleeping(x), and two binary ones, SamePlace(x, y) and SameSpecies(x, y). In addition, there is a function to return to the closest given animal (THECLOSEST(x)). The World frame (Figure 2) consists of a map divided into continents where cities (capital/non capital ones) may be located and connected. Six unary predicates are defined, such as IsCapitalCity(x), IsInAmerica(x), IsInAsia(x), and five binary ones, as SameContinent(x, y) and ThereIsAPath(x, y). The function THEFARTHEST(x) returns, for a given city, to the farthest one.

Figure 2. FOLST for Farm frame

Formulas for a selected frame may be written in the editor window which shows the logical connectives and quantifiers considered by the tool (Figure 2).

The tool verifies if each formula is syntactically correct in connection with a context-free grammar defined to recognize FOL well-defined formulas. This grammar was implemented using the free tools Flex, for lexical analysis (Paxson, 2012), and Bison, for syntactic analysis (Donelly & Stallman, 2017) In case of an error, FOLST reports the type of mistake the user has made so that s/he could detect and correct it easily. This is important from a didactic point of view since the users are not only warned about the error but they also get some clues to correct it.

ISSN: 2473-3857
v3 n4356

For example, Figure 2 shows three formulas written in a model based on the frame Farm.

The tool informs in each case which is the error in the definition of the formula; errors could be independent of the frame used to instantiate the model (a parenthesis missing as in Formula 1, or the presence of a free variable as in Formula 2) or they could be specific for a particular frame (the use of an undefined predicate as in Formula 3). This figure also shows that the tool gives users the possibility to work with many different models simultaneously. For each formula in the editor window, FOLST computes its truth value in a model when the user selects the option Verify formula. The possible results are Valid, in case the formula is true in the considered model, and False otherwise. The user may change the model, for example by adding some cities, if working with the World frame, and may ask the tool to recalculate the formula truth value. Five formulas were defined to be evaluated in this model. As all of them are syntactically correct, the tool shows for each one the corresponding truth value. In addition, it is important to emphasize that FOLST allows saving/loading models and formulas.

LogicChess (Kiehr & Re Medina, 2012) allows the user to write formulas in the editor window checking them to determine if they are syntactically correct. Correct formulas may be evaluated in user-defined models. Each model represents a chessboard composed by chess pieces (rook, knight, queen, king, etc) which have attributes such as color (black, white), type, and position. Users may define a finite set of models in an easy way by adding, deleting or modifying model components.

The tool provides fifteen predicates classified in: identification predicates such as isPawn(piece), isKnight(piece), etc.; position predicates as for example sameRow(piece1, piece2), isInL(piece1, piece2); and distance predicates such as distance(piece1, piece2, number), freePath(piece1, piece2) Using the elements presented in the previous paragraph, in an analogous way to what FOLST does with farms and world maps, LogicChess allows students to introduce FOL formulas and perform on the fly modifications of the model. After modifying the chessboard, the truth value of the formulas is updated. The same happens when a formula is modified. In that way, users can modify both model and formulas having an instant verification of its satisfiability.

CLAUSULA and CLPROVER

Clausula (Ferrante, 2009) (Mauco & Ferrante, 2009) allows students and teachers to experiment with arbitrary FOL sets of clauses in order to determine their (in)satisfiability. Clausula implements classic and fundamental methods of FOL such as the Resolution method for clauses (Ben Ari, 2012). Also, the tool gives the possibility to calculate the most general unifier of a pair of literals, using the Unification Algorithm proposed by Robinson. Regarding the (in)satisfiability determination of a set of clauses, the tool detects if the set corresponds to a Program Logic or it is an arbitrary set, so as to apply the adequate strategy.

Clprover (Mauco, Moauro & Felice, 2010) is the other tool developed to determine the (in)satisfiability of FOL clauses, with the possibility to work from arbitrary formula (Figure 3). Clprover provides the factorization and subsunction of clauses.

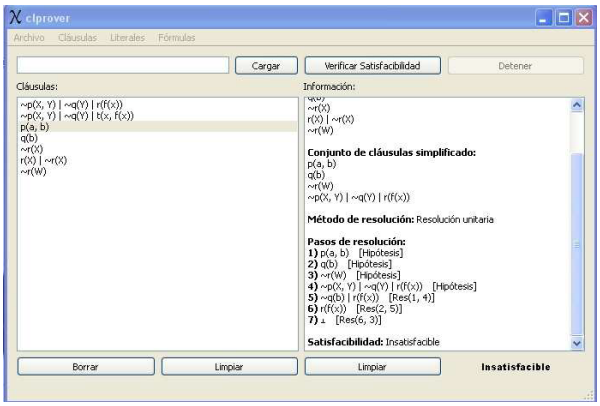

Figure 3. Clprover. Resolution Method

TheM

The THeM tool (Teaching Herbrand Models) (Dahl & Fujii, 2016) allows to determine the (in)satisfiability of a set of FOL clauses working with Herbrand Models. An algorithm that shows the procedure made manually by students was implemented as part of the didactic tool used to solve this kind of problems. First, a set of defined clauses is generated. Then, the tool searches the existence of Herbrand Models for the set (Figure 4).

ISSN: 2473-3857
v3 n4356

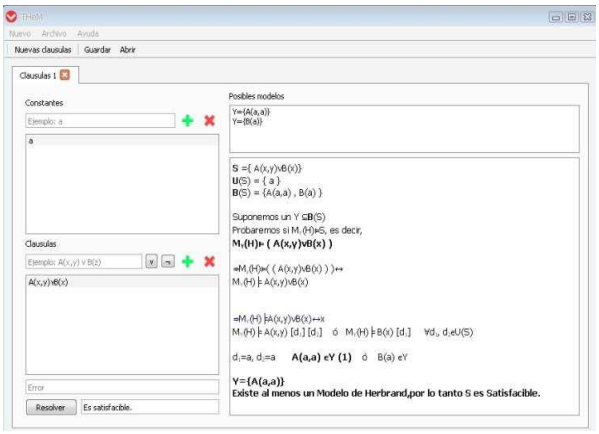

Figure 4. TheM. Herbrand Models

5. THE USE EXPERIENCE AND THE TOOLS DEVELOPMENT

The tools presented in the previous section are used in the Logic course as support software in class and to complete homework. They are used by teachers and students. The students have reported positive experiences in the use of the tools. About one hundred students per year attend Logic course. They manifested the easiness in the use, and make the verification of the exercises results. Thereby, teachers make sure that the student participation in class is more active and more productive, having good evaluation results. Nowadays, there is a tool for each fundamental concept addressed in the course.

Advantages and disadvantages that students have expressed during their practices using the mentioned tools have greatly influenced to continuously work to enhance the proposal. It has been reported an improvement in the performance of the students during the course. In addition to the improvement in the academic performance, students involved themselves as active users and testers which allow them to provide new ideas to enhance the tools.

These are some of the tools developed by students that were described in this paper.

- DPSolver and SAT for Propositional Logic and their applications,
- YAT3 for refutation trees in Propositional Logic and First Order Logic,
- FOLST and LogicChess for semantic in First Order Logic,
- THeM for Herbrand Models,
- Clausula and ClProver for resolution in First Order Logic.

All the tools were implemented in C++ programming language using the framework QT for the graphic interfaces. The designed solution and the graphic interface are independent, therefore encouraging reuse and simplifying maintenance.

The tools are free software and they use platform-independent technologies, so that versions for other platforms could be released. When using free software in the teaching/learning process, students have the possibility of using and sharing the resources it offers, and they are encouraged to have a look at the code, what makes it even more interesting since these tools have been developed under the same technologies they are learning at the time.

6. CONCLUSIONS

This paper presents a project aimed at students in the second year of a Computer Science career. The main objective of ´Educational Tools Development´ project is the design and implementation of didactic tools for an introductory Logic course used as support in the teaching/learning process. The developed tools use the same notation and the methodology that Logic course follows to introduce the course contents.

A formal and systematic evaluation is being carried out by teachers and the first results are very encouraging, so, there is much enthusiasm for further development of tools.

For the students who developed these tools, the contribution involved three aspects:

- a first experience in the development of complete software for a real user, from the requirements stage to the implementation,
- learning new technologies and putting into practice the knowledge and the methodology acquired in the Design Algorithm course,
- the integration and more deep comprehension of the logic content involved in the tools functionality.

Another contribution for these students was that they presented the tool during a class of the Logic course. In this way, they had a first experience of oral presentation to a complete course, allowing an informal discussion with the students who were motivated to see that

ISSN: 2473-3857
v3 n4356

students only slightly more advanced than them were able to develop a complete tool.

In addition, teachers of both courses encouraged students to present the tools in national competitions for students and in national and international conferences. This allowed students to have a first experience in preparing and presenting a conference paper.

It is considered that the development of a transversal project for two courses, such as the one presented in this paper, allows students to integrate and deepen the contents of both. Following the completion of this project, another important observation is that students visualize the courses as part of a process that incrementally will form them in their career.

7. ACKNOWLEDGEMENTS

The authors give thanks to the students and teachers involved in the ´Educational Tools Development´ project for their dedication and their hard work. In addition, the authors acknowledge the students that are using the tools providing a feedback to improve the teaching/learning process.

8. REFERENCES

Aho, A., Ullman, J (1995). Foundations of Computer Science. C Edition. Computer Science Press.

Ben-Ari, M. (2012) Mathematical Logic for Computer Science. *Springer Verlag London.*

Blanchette, J., & Summerfield, M. (2008). C++GUI Programming with Qt 4 2nd Edition.Prentice Hall Open Source Software Development Series.

Cicconi, D., Fernández Cocirio, M. (2014). DPSolver: Una herramienta interactiva para la implementación del algoritmo de Davis Putnam. Simposio de Trabajos Estudiantiles, 2º CoNaIISI, Argentina.

Cormen, T.; Lieserson, C.; Rivest, R. (2009). Introduction to Algorithms. Ed. The MIT Press.

Dahl, J., Fujii, D. (2016). THeM: Una Herramienta Didáctica para Modelos de Herbrand. Simposio de Trabajos Estudiantiles, 45ºJAIIO, Argentina. 232-141. Retrieved June 2017 from http://45jaiio.sadio.org.ar/sites/default/files/EST-1635.pdf.

Donnelly, C. & Stallman, R. (2017) Bison Version 1.25: The YACC-compatible Parser Generator. Retrieved June 2017 from http://dinosaur.compilertools.net/bison/index.html.

Ferrante, E. (2009). Clausula: Herramienta Didáctica para la Enseñanza de Lógica de Predicados de Primer Orden. Simposio de Trabajos Estudiantiles, 38ºJAIIO, Argentina.

GNU General Public License (GPL). Retrieved June 2017 from http://www.gnu.org/licenses/gpl.html

Kelley, J. (1997). The Essence of Logic, Prentice Hall.

Kiehr, A., Re Medina, M. (2012). LogicChess: Herramienta Didáctica para la Ejercitación en Lógica de Predicados de Primer Orden. Simposio de Trabajos Estudiantiles, 41ºJAIIO, Argentina. 394-404. Retrieved June 2017 from http://41jaiio.sadio.org.ar/sites/default/files/19_EST_2012.pdf.

Maggiori, E., Gervasoni, L. (2012). FOLST: Una Herramienta Didáctica para la Lógica de Predicados de Primer Orden. Primer Premio en el Concurso de Trabajos Estudiantiles, 41ºJAIIO, Argentina. 405-415. Retrieved June 2017 from http://41jaiio.sadio.org.ar/sites/default/files/20_EST_2012.pdf

Mauco, M.V., & Ferrante, E. (2009). Clausula: A Didactic Tool to Teach First Order Logic. Publishing in ISECON 2009, Information Systems Education Conference, Washington DC. USA.vol 26: §4142.

Mauco, M.V., Maggiori, E., Gervasoni, L., Ferrante, E., & Felice, L. (2012). FOLST: A Didactic Tool to Support First Order Logic Semantics Learning. Publishing in Proceedings of International Conference on Future Computers in Education. Shangai. China. Lecture Notes in Information Technology, Vols.23-24, 302-307.

ISSN: 2473-3857
v3 n4356

Mauco, M.V., Moauro, L & Felice, L. (2010). Una Herramienta Didáctica para la Enseñanza de Lógica de Predicados de Primer Orden. Publishing in Congreso Iberoamericano de Educación Superior en Computación (CIESC 2010).

Paxson, V. (2012) Flex - Version 2.5: A Fast Scanner Generator. Retrieved April 2012 from http://dinosaur.compilertools.net/flex/flex.ps

Qt:Documentación oficial sobre el entorno gráfico. http://doc.qt.io

Ruau, K., Tosini, J.M. (2015). Yet.anotherTruth.Tree.Tool: Una herramienta didáctica sobre Árboles de Refutación. Simposio de Trabajos Estudiantiles, 44ºJAIIO, Argentina. 63-71. Retrieved June 2017 from http://44jaiio.sadio.org.ar/sites/default/files/est63-71.pdf.

Santillán Cooper, M., Horquin, E., Covelli, T (2017). SAT: Una Herramienta Didáctica para el problema de la satisfacibilidad. Segundo Premio en el Concurso de Trabajos Estudiantiles, 46ºJAIIO, Argentina. 62.71. Retrieved September 2017 from http://www.clei2017-46jaiio.sadio.org.ar/sites/default/files/Mem/EST/est-06.pdf