Educational Software for First Order Logic Semantics in Introductory Logic Courses

María Virginia Mauco
vmauco@exa.unicen.edu.ar
Facultad Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Tandil, Argentina

Enzo Ferrante
enzo.02@gmail.com
Center for Visual Computing, Ecole Centrale de Paris, France
Galen Team, INRIA Saclay, France

Laura Felice
lfelice@exa.unicen.edu.ar
Facultad Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Tandil, Argentina

Abstract

Basic courses on logic are common in most computer science curricula. Students often have difficulties in handling formalisms and getting familiar with them. Educational software helps to motivate and improve the teaching-learning processes. Therefore, incorporating these kinds of tools becomes important, because they contribute to gaining practice in dealing with formalisms. In particular, semantic analysis of first order logic formulas is an issue that presents several difficulties. For this reason, we developed two educational tools, *FOLST* and *LogicChess*, to support the teaching/learning process in first order logic semantics. Both tools are didactic, visual, and interactive. They allow users to experiment with first order logic formulas to determine their truth value. They are implemented in C++, and they have been released under a free software license. In this paper, we present *FOLST* and *LogicChess*, and we propose to design a framework based on the development and use of these two didactic tools.

Keywords: Educational Software, First Order Logic Semantics, Teaching Resources, Logical Concepts.

1. INTRODUCTION

Basic courses on logic are common in most computer science curricula. In this kind of course, students have to do a lot of individual work to solve exercises and to gain experience in working with formalisms. In this context, the use of didactic tools that support the learning process, without taking too much time to learn how to use them, becomes really useful.

The undergraduate degree program in Systems Engineering in our University has an introductory course on Propositional Logic and First Order

Logic (FOL) that is offered in the first semester of the second year.
Learning these subjects requires substantial individual work by students, because they have to solve logic exercises in order to obtain skills in handling formalism. At first, we used lectures and a pencil-paper problem solving approach to teach course content. From our teaching experiences, we noticed that students were not as motivated and interested as in programming courses, though they could understand the concepts. As educational tools can offer useful pedagogical possibilities that motivate and enhance teaching/learning processes, we have developed several didactic tools that follow the same logical notation and methodology used in the course and which are very easy and intuitive to use, with a fast speed of learning (Mauco & Ferrante, 2009; Mauco, Moauro & Felice, 2010). These tools are part of a project for developing educational software to support teaching/learning processes in introductory logic courses for computer science students. Advanced students developed them with teachers´ assistance, by applying design and programming methodologies learnt in other courses in their careers.

FOL is an extension of Propositional Logic, which includes predicates and functions (specified over a domain), to define a model or formula interpretation (Ben-Ari, 2003). Besides, quantifiers may also be used. The truth value of a formula depends on the model in which the formula is evaluated, making its determination more difficult than in Propositional Logic. There are some strategies, for example, that try to find contradictions in formulas. For this kind of problems, some didactic tools have been implemented (Huertas, 2011). However, FOL semantics (Ben-Ari, 2003; Harrison, 2009; Huth & Ryan, 2004) is one of the topics that turn out to be more difficult to understand for students, as it is possible to define infinite arbitrary models when determining the truth value for a given FOL formula. *Tarski's World* (Baker-Plummer, Barwise & Etchemendy, 2007) and *Moros y cristianos* (LLorens Largo, 2013) are tools developed as support in this topic.

To complement FOL semantics teaching/learning process, we have developed two didactic, visual, and interactive tools, *FOLST* (Mauco, Maggiori, Gervasoni & Felice, 2012) and *LogicChess* (Kiehr & Re Medina, 2012), which assist in the evaluation of FOL formula in models defined by the user in the domains provided by each tool. *FOLST* (First Order Logic Semantics Tutor) allows teachers and students to experiment with FOL formulas and determine their truth value. In order to achieve this goal, *FOLST* provides the users with the possibility of defining their own models working on two frames called Farm and World. Each of these frames provides the necessary elements to define a concrete model (a domain, functions, relations) in order to evaluate well-defined formulas in each of them. *LogicChess* allows users experiment with FOL formulas and find out their truth value working with models defined over a chessboard. Both tools give assistance when introducing formulas and show the corresponding truth value of each written formula in the model defined by the user.

Though these tools allow users to experiment with FOL semantics in real situations, the domains, functions, and relations implemented by each tool limit models definition. FOL expressiveness power is based on the possibility of working with arbitrary domains, functions, and relations, defined considering the situation the user wants to formalize. In order to take advantage of teachers and students experience in the use of *FOLST* and *LogicChess*, we conclude the paper proposing the development of a framework to support FOL semantics teaching/learning process incorporating domains, relations, and functions defined by users.

The paper is structured as follows. In Section 2 main issues related to FOL semantics are briefly introduced. Section 3 presents the educational tools *FOLST* and *LogicChess*. Section 4 describes the evaluation of the use of both tools in a logic course. In Section 5 we present the proposal of development of a framework to support FOL semantics teaching/learning process. Finally, some conclusions and possible future work are mentioned in Section 6.

2. FIRST ORDER LOGIC SEMANTICS

As FOL extends propositional logic to a broader degree of expressiveness, determining truth and implication for FOL formulas is harder than in case of propositional logic. FOL languages need to be able to refer to the objects that appear in propositions, and they also need to describe relations between objects (Ben-Ari, 2003; Harrison, 2009; Huth & Ryan, 2004).

FOL formulas only make sense if a model or interpretation is defined, which involves specifying predicates, functions and a domain of discourse. A formula in FOL might be evaluated in different models, and even might have a different sense in each of them. That is why formulas are more difficult to analyze in FOL. Since the predicates and functions described by a FOL formula could be anything, and the particular state of the domain of discourse might also change their truth values, FOL is absolutely instance-dependent. That is the reason why it is not possible to determine the truth value of any FOL formula without considering the definition of the involved model. However, there are many algorithms that deal with, for example, finding contradictions in formulas (Ben-Ari, 2003; Harrison, 2009). In this way, if a contradiction is found, it can be well stated that no possible model would ever make that formula be true (but if a contradiction is not found, a model-dependent approach could be considered anyway in order to decide its truth value).
Clausula (Mauco & Ferrante, 2009) and *Clprover* (Mauco et al., 2010), the tools that had already been developed for our course, dealt with this kind of methods. The idea of *FOLST* and *LogicChess* was complementary to that: to let the user experiment FOL in realistic situations and evaluate formulas on them, in a totally instance-dependent way. This would let the user feel the meaning of FOL and understand its applications.

3. THE TOOLS: FOLST AND LOGICCHESS

FOLST (Mauco et al., 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. These tools were developed by third-year students in computer science careers as the final project for two courses that involve logic and algorithms analysis and design contents (both courses are taken in the first semester of the second year of the career). Both tools were implemented in C++ programming language, using Qt framework for the graphical interface (Blanchette & Summerfield, 2008). In addition, they are free software under GNU GPL v3.0 license (GNU, 2012). In the context of educational software, this is important mainly because of two reasons: regarding students, it allows them to explore, experiment and analyze concrete implementations of algorithms and thus, besides using them as support in their learning processes, it encourages them to participate in the development of their own tools. Regarding professors, it strengthens resource sharing between different universities and collaborative improvement of their courses.

As *FOLST* and *LogicChess* have been designed with the purpose of being a didactic tool, it is important to highlight the functionalities they include as regards FOL semantics teaching. During FOL semantics learning process, students may have to face the following issues:
- to determine if a FOL formula is syntactically correct;
- to evaluate a formula in a model and determine its truth value;
- to define new models to evaluate given formulas and observe changes in their truth value;
- to determine if a formula is logically valid (valid in every model), contradictory (false in every model), or just satisfiable (valid in one model but false in another one).

In all these cases both tools offer didactic support to verify exercises, giving confidence to students about the correctness of their results.

FOLST

The tool provides the implementation of two frames, Farm and World that allow the definition of different models. The Farm frame (Figure 1) consists in 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), and predicates allowing 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, given an animal, its closest one (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, the farthest one.

Formulas for a selected frame may be written in the editor window, which shows the logical connectives and quantifiers considered by the tool (Figure 1). The tool verifies if each formula is syntactically correct with respect to 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, 2012) 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. For example, Figure 1 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 to a particular frame (the use of an undefined predicate as happens 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 adding some cities if working with the World frame, and ask the tool to recalculate the formula truth value. Figure 2 presents a model that is an instance of the World frame. Five formulas were defined for evaluation 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 remark that *FOLST* allows saving/loading models and formulas.

LogicChess

This tool 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 colour (black, white), type, and position. Figures 3 and 4 show the main elements to define a model in this tool: the chessboard and a piece. 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 is Pawn (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.

4. DISCUSSION AND ASSESMENTS

FOLST and *LogicChess* are being class-tested during this semester. They are being used in lectures, to introduce FOL semantics, and for homework assignments. Students are now reporting positive experiences with its use (we have approximately 100 students per year) indicating that both tools are easy and intuitive to use. They also appreciated the assistance provided by the tools to correct mistakes when writing FOL formulas thanks to the grammar checking; it helped them to understand the concept of well-formed formulas. As another advantage, they mentioned that working with simple-to-understand and natural models as farms, world maps or chessboards, allowed them to focus on the understanding of the formula semantics instead of putting attention in the complexity of the model itself. To interpret and express complex models in the context of FOL is a difficult task that necessarily requires having a previous thorough knowledge of FOL semantics; our tools help to gain this knowledge. Using *FOLST* and *LogicChess* enhance the student´s learning experience through engaging them in the formalization of realistic situations; this is particularly important because both tools were thought for beginner students without formalism handling experience.

5. TOWARDS A FRAMEWORK FOR FOL SEMANTICS

Although there are some tools for evaluating semantics of formulas in FOL in which users may define different models, all of these tools work on a predefined domain with relations and functions also predefined, limiting then the definition of models (Baker-Plummer et al., 2007; Kiehr & Ré Medina, 2012; Llorens Largo, 2013; Mauco et al., 2012).

The expressiveness of FOL lies precisely in the possibility of working with arbitrary domains, defined according to user needs. Users should also be able to specify the relations and necessary functions considering the situation that each one wants to formalize.

In this context, it is important to emphasize that the design of *FOLST*, for example, was thought to make the tool extensible. Thus, new domains can be easily added by simply setting their relations and functions, without having to work on parsing or semantic evaluation algorithms of formulas. However, it is necessary to have knowledge of the design and implementation of the tool in order to specify the appropriate classes in each case.

Though the experience using *FOLST* and *LogicChess* has been successful, we think it can be enhanced by offering a more general tool. For this reason, we conceptually conceived an interactive and visual tool, which through a friendly interface, allows specifying different domains with associated relations and functions to define specific models or interpretations, taking into account that the main users of this tool will be students from the early years of the career, with little experience in programming. In this way, abstract models are left out and users will be able to define real, tangible models, in which users can work to interpret each formula and determine its truth value. To the best of our knowledge, there is no educational tool with these features.

Considering this situation, we are working on the development of a framework that supports and assists the user in specifying different frames as a basis for defining models, where a frame is defined by a domain and a set of relations and associated functions. Because this tool is intended to become a didactic support tool for the teaching/learning process of FOL semantics, not only for students but also for teachers, the framework should provide support for:

- a simple and intuitive specification of frames;
- maintenance of library of frames;
- definition of multiple models and formulas for each frame;
- parsing of formulas;
- semantic evaluation of formulas in a model;
- error handling either in frames specification or in definition of models and formulas.

Moreover, as self-assessment contributes to students´ learning process, we consider it very important that the framework gives support to analyze the results of the students' practices and automatically returns information about their corrections. This functionality is not included by neither *FOLST* nor *LogicChess*.

Considering the pedagogical and academic profile that we want to provide to the tool, platform-independent technologies and free software will be used for the design and implementation. In addition, special emphasis will be given to the development of appropriate and complete documentation and examples with which allow users to understand how to use the framework. Besides, to design the tool interface, aspects of human-computer interaction that facilitate students' teaching-learning process will be analyzed in detail (HCI Bibliography, 2013).

6. CONCLUSIONS

In this paper we presented two concrete tools to support teaching/learning process in FOL semantics, and we set the basis for the development of a framework conceived as the abstraction of these tools. This proposal arises from the authors' experience with the development and use of *LogicChess* and *FOLST* tools, and from the fact that in the analysis of existing educational software we have done, no tools were found to work with arbitrary domains, relations and functions in FOL.

Advantages and disadvantages that students have expressed during their practices using the mentioned tools have greatly influenced our framework proposal. After experimenting the advantages that provides an interactive

graphical tool to solve exercises in a formal language as FOL, students suggested the need of new domains, relations and functions to model different situations.

An important aspect of *FOLST* and *LogicChess* tools is that they 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, which makes it even more interesting since these tools have been developed under the same technologies students are learning at the time. But above all, full potential of educational computer programs is exploited. Given the positive experience with these tools, the framework will be developed using Free Software.

FOLST and *LogicChess* were developed by students to be used by other students, which makes it motivating in the teaching/learning process. They are currently being used as a complementary tool throughout the course, and will be used for sure in the future too. The tools focused strongly on the appearance of formulas, making them as similar as they were in class, and it is geared toward letting the user work with multiple models and formulas at the same time. Experiences from both projects were presented and published at student's symposiums (Maggiori & Gervasoni, 2012; Kiehr & Ré Medina, 2012).

The aim of this kind of tools is to bring Logic down to earth, a science absolutely abstract by itself, and let the student experiment its scopes and limitations under a daily-life situation. This understanding is highly appreciated before and while facing FOL formula solving algorithms and techniques.

7. REFERENCES

Baker-Plummer, D., Barwise, J., & Etchemendy, J. (2007). Tarski´s World. University of Chicago Press.

Ben-Ari, M. (2003). Mathematical Logic for Computer Science. Prentice Hall. Series in Computer Science.

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

Donnelly, C. & Stallman, R. (2012) Bison - Version 1.25: The YACC-compatible Parser Generator. Retrieved April 2012 from http://dinosaur.compilertools.net/bison/index.html.

GNU General Public License, version 3 (2012). Retrieved April 6, 2012. from http://www.gnu.org/licenses/gpl-2.0.html.

Harrison, J. (2009) Handbook of Practical Logic and Automated Reasoning. Cambridge University Press.

HCI Bibliography: Human – Computer Interaction Resources. (2012). Retrieved June 20, 2013 from: http://hcibib.org/.

Huertas, A. (2011). Ten Years of Computer-based Tutors for Teaching. Logic 2000-2010: Lessons Learned. Lecture Notes in Computer Science Volume 6680, 131-140.

Huth, M., & Ryan, M. (2004). Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press.

Kiehr, A., & Ré Medina, M. (2012). LogicChess: Herramienta Didáctica para la Ejercitación en Lógica de Predicados de Primer Orden. Concurso de Trabajos Estudiantiles, 41 JAIIO, Argentina, 394-404.

Llorens Largo, F. (2013). Herramienta didáctica para el aprendizaje práctico de la Lógica de Primer Orden: Moros y Cristianos. Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante, Retrieved June 20, 2013 from: http://www.dccia.ua.es/logica/MyC/index.htm.

Maggiori, E., & Gervasoni, L. (2012). FOLST: Una Herramienta Didáctica para la Lógica de Predicados de Primer Orden. Concurso de Trabajos Estudiantiles, 41 JAIIO, Argentina, 405-415.

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.

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.

Appendix

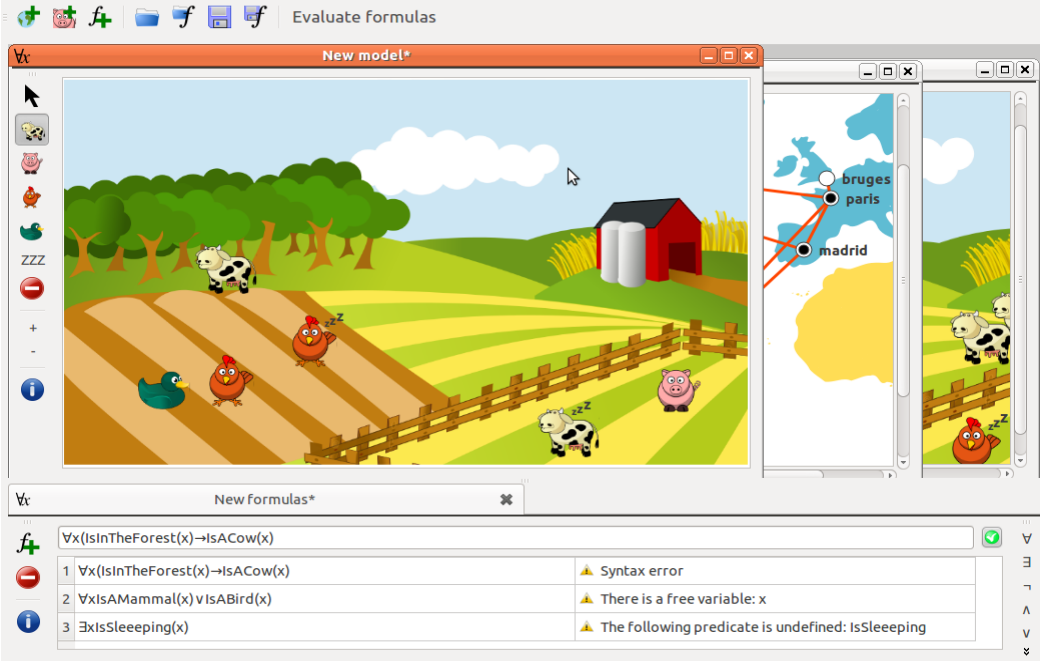

Figure 1. Model defined by the user in the Farm frame

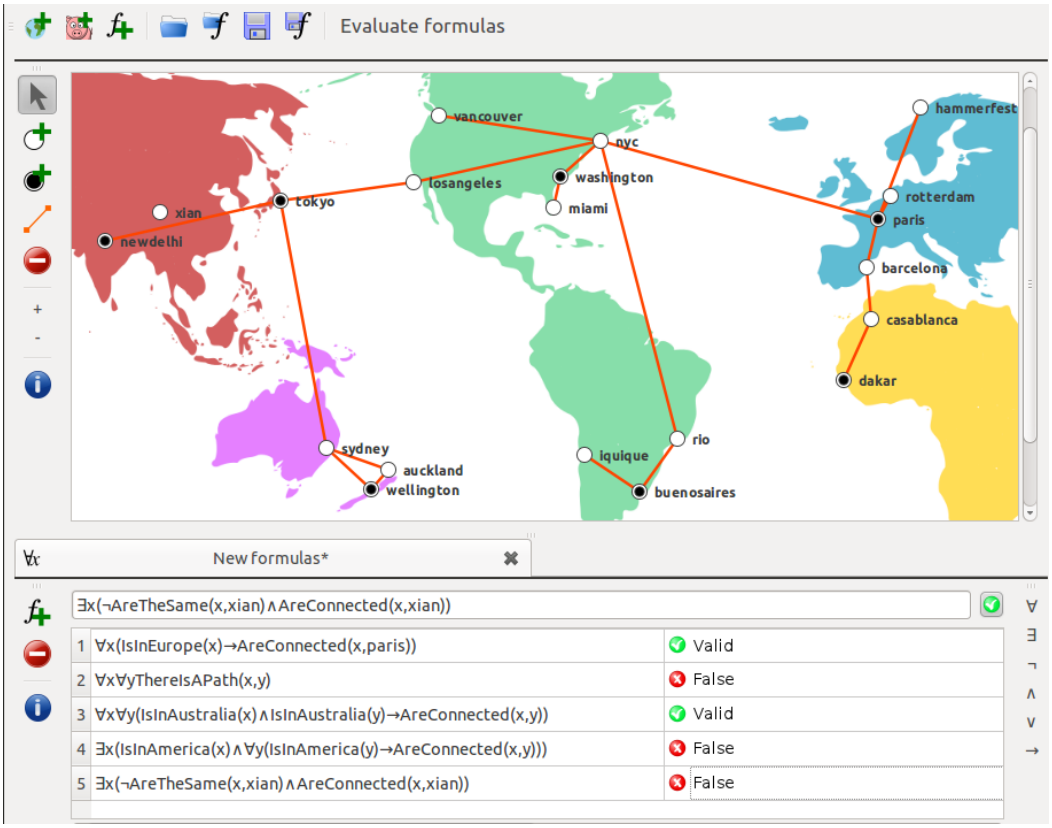

Fig. 2. FOLST evaluating truth values of formulas under user-defined World model

Figure 3. LogicChess evaluating a formula under user-defined model

Figure 4. LogicChess: a piece and its attributes