FITS: Formalization with an Intelligent Tutor System

José A. Alonso Jiménez*[,1], Gonzalo A. Aranda Corral[2] and Francisco J, Martín Mateos[1].

[1] Dpto. C. Computación e IA, Universidad de Sevilla, Avda Reina Mercedes, 41012 Sevilla, España
[2] Dpto I.E.S.I.A., Universidad de Huelva, Crta Palos de la Frontera, Palos de Frta (Huelva), España

Knowledge Representation & Reasoning is a fundamental topic in Artificial Intelligence. Different systems exist to help to teach this, usually focused on proofs, but anyone on formalization, which is the hardest process for students to understand in logic and AI courses. In this work we present FITS (Formalization with an Intelligent Tutor System), the first intelligent tutor to formalization teaching oriented and dedicated to both learners and teachers. At the moment the system embed three tools, the teacher's module to create exercise, the learner's module to solve them and the intelligent module which helps the teacher to design the exercises in the right way and to correct the learner’s exercises semantically. Automated reasoning systems have been used to develop the intelligent module. FITS is designed to have a web-based interface, and accessible from any browser, showing a simple attractive look.

Keywords Computed Assisted Instruction, Intelligent Tutor System, First Order Logic, Formalization, Knowledge Representation & Reasoning, Automated Reasoning.

1. Introduction

Logic plays an important role in teaching “Computer Engineering”. For instance, at the University of Sevilla logic is involved in many courses: “Computational Logic”, where the propositional and the first order logic are introduced; “Logic Programming”, where logic is used as programming language; “Artificial Intelligence Seminars”, where logic is used to represent knowledge; or “Automated Reasoning”, where logic is used to formalize problems in such a way that an automated reasoning system could manage them. In teaching these courses, different computer system are used as JAPE [3], a proof-editor for natural deduction; Tree Proof Generator [8], a system based on semantic tableaux; OTTER [5], an automated reasoning system based on resolution; and SWI Prolog [9], a logic programming language. Other systems for the tutorial teaching in logic are the “Logic Tutor” [2], the Logic-ITA [3], the “P-Logic Tutor” [4] and OLIVER [6]. Moreover, more tools for teaching Logic can be found on the web “Logic software and logic education” [1].

In spite of the amount of existing tools, there is an important subject on teaching logic for which these tools couldn't be used: *formalization*. Formalization is the process of translating arguments from natural language to logic. This is the initial step on the “Computational Logic” subject and the main one when logic is used to represent knowledge. Our experience shows us that this is the most difficult process for students, and the way to learn it is solving exercises.

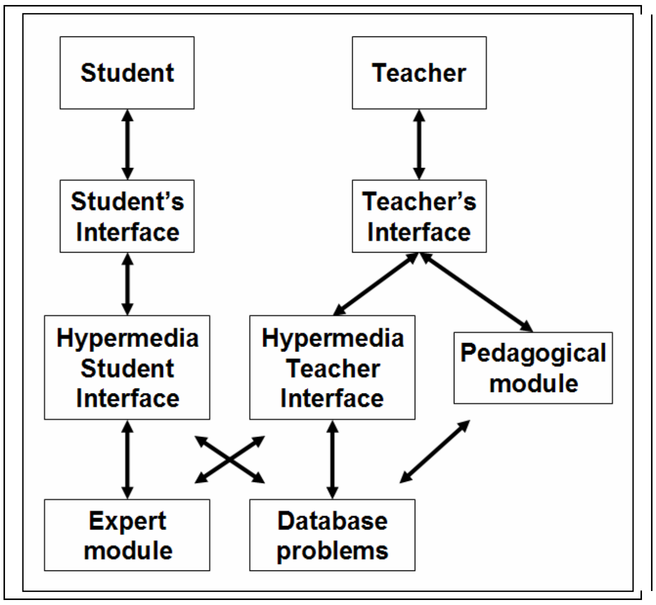
Until now, the only tools which help on teaching the formalization process are interactive forms as “Logic Tutor” by Huth and Ryan [4]. In these forms, the argumentations are presented and the student has to choose among the different proposals. As the forms have fixed answers, the possibilities of alternative equivalent formalizations are excluded.

In this work we present FITS, the first formalization teaching system which offers an opened answer style and the capacity of semantics correction. During the course 2006-07, we are going to integrate this system on the teaching of “Computational Logic” and other subjects.

* * Corresponding author: e-mail: jalonso@us.es, Phone: +34 954557955

2. Architecture and implementation

The architecture of FITS is showed in the following figure.



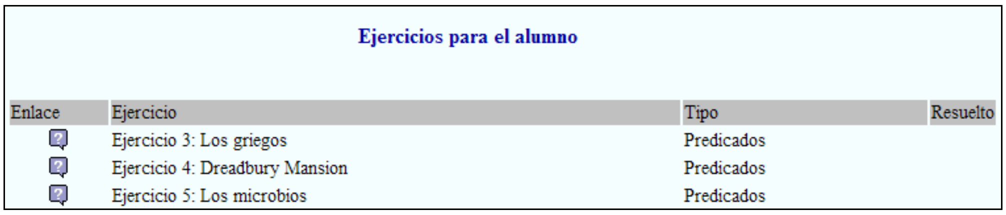
The problem database is stored on MySQL by the teacher. To introduce them, it disposes of the teacher hypermedia application that generates the forms HTML showed on the teacher interface.

The student interface is a set of HTML forms dynamically generated from the problem database. The student hypermedia application controls the communication with the expert module.

This expert module is the section where the system intelligence is located. This module is the responsible to check all the teacher argumentation and the student answers.

2.1. FITS from the student point of view

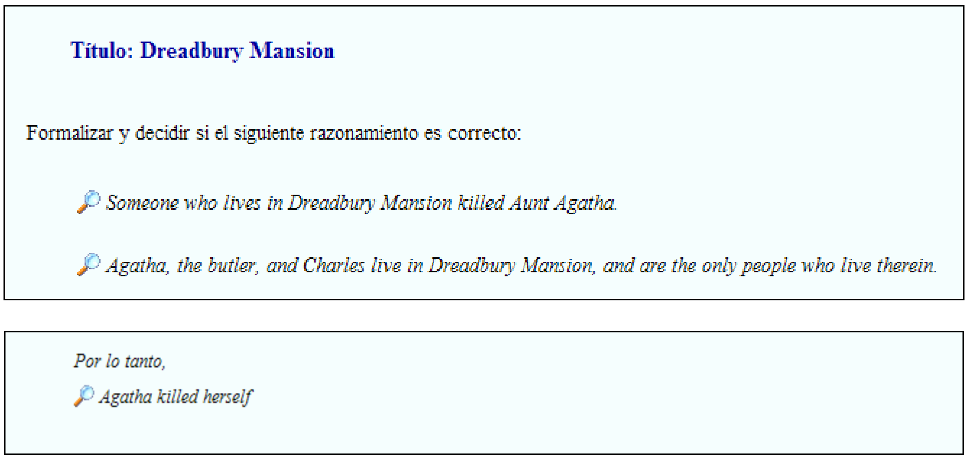
FITS is developed for being used on Internet. To enter it, the students must connect with the web page at http://www.cs.us.es/clg/FITS. After that, the system offers a collection of argumentation exercises (as it is shown in the following figure), and the student clicks on the one she wants to solve.



Ejercicios para el alumno

Enlace	Ejercicio	Tipo	Resuelto
?	Ejercicio 3: Los griegos	Predicados	
?	Ejercicio 4: Dreadbury Mansion	Predicados	
?	Ejercicio 5: Los microbios	Predicados	

Inside the problem, a list of hypothesis and conclusion is presented. If the student wants to solve the exercise, she must click, one by one, on all of them and write the answers.



Título: Dreadbury Mansion

Formalizar y decidir si el siguiente razonamiento es correcto:

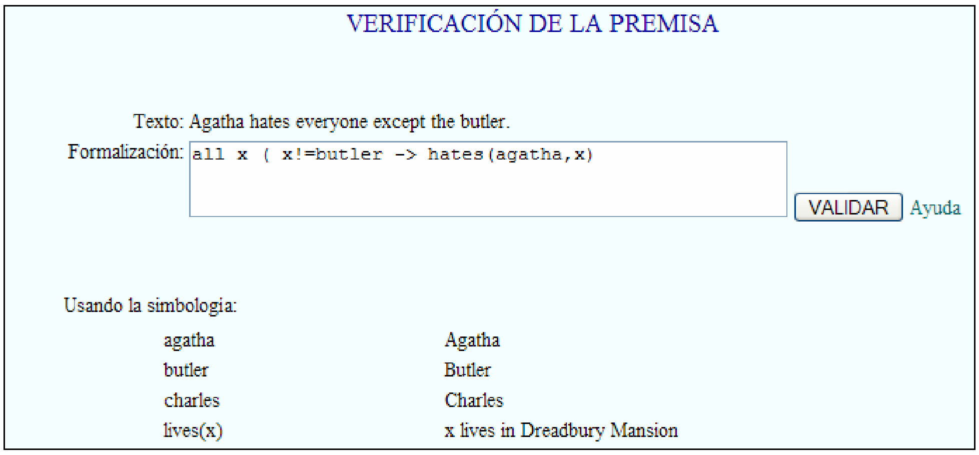
Someone who lives in Dreadbury Mansion killed Aunt Agatha.

Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein.

Por lo tanto,

Agatha killed herself

When an hypothesis is selected, the system shows the verification form. This form contains the argument and a blank box where the student can write the formalization using OTTER syntax.



VERIFICACIÓN DE LA PREMISA

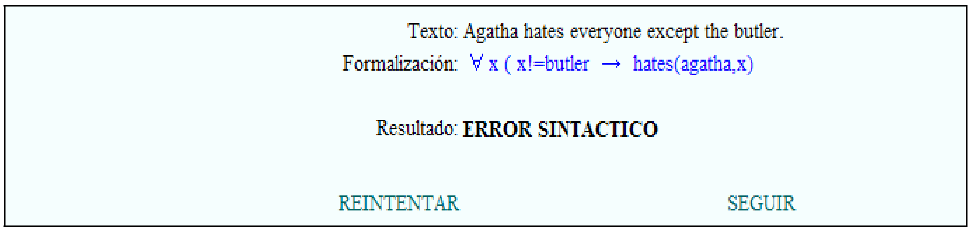
Texto: Agatha hates everyone except the butler.

Formalización:
```
all x ( x!=butler -> hates(agatha,x)
```

Usando la simbologia:

agatha	Agatha
butler	Butler
charles	Charles
lives(x)	x lives in Dreadbury Mansion

To check the formalization, the "VALIDAR" button should be pressed and a new window appears with the result. The system reports syntax errors in the proposed formalization:

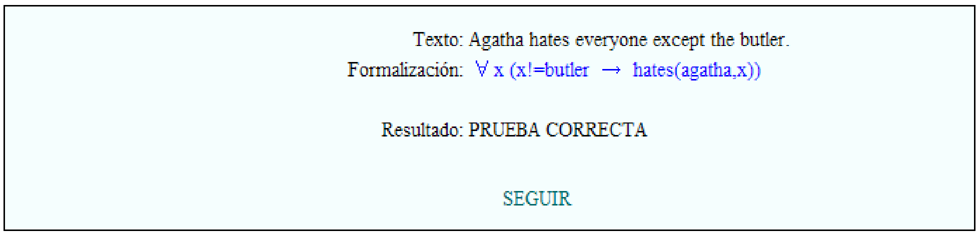


Texto: Agatha hates everyone except the butler.

Formalización: $\forall$ x (x!=butler $\rightarrow$ hates(agatha,x)

Resultado: **ERROR SINTACTICO**

as well as successful formalization modulo semantic equivalence:



Texto: Agatha hates everyone except the butler.
Formalización: ∀ x (x!=butler → hates(agatha,x))

Resultado: PRUEBA CORRECTA

SEGUIR

We note that the student formalization expression is showed in mathematical notation translating OTTER syntax. This will be used whenever the system shows the equations.

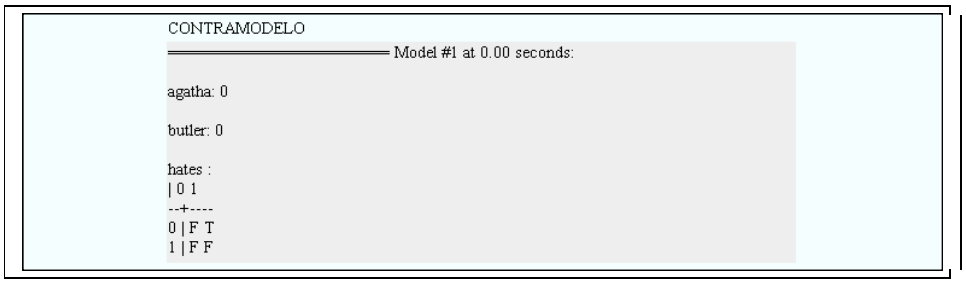
The newest aspect of FITS is its semantic nature; that is, the system accepts as right answer any formal equation which is logically equivalent to the solution, using the problem symbolization. In this example, the student could formalize the conclusion using the following formula

$$\forall x \ [x = \text{butler} \vee \ \text{hates(agatha,x)}]$$

or any other equivalent. But if the conclusion is formalized as

$$\forall x \ [x = \text{butler} \rightarrow \ \text{hates(agatha,x)}]$$

the system indicates that it is an incorrect formalization and shows a counter-model.



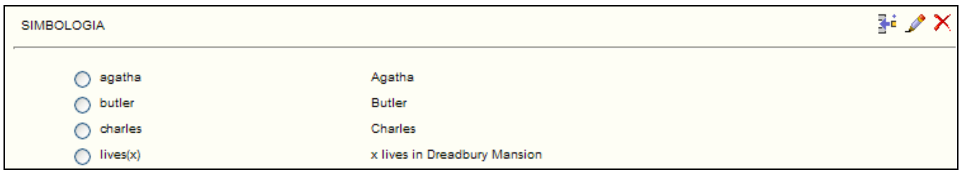
CONTRAMODELO

```
======================== Model #1 at 0.00 seconds:

agatha: 0

butler: 0

hates :
|0 1
--+----
0 |F T
1 |F F
```

2.2 FITS from the teacher point of view

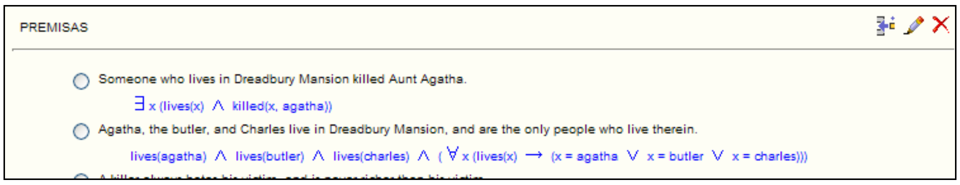
The first FITS service for teachers is to help on problem edition. To create a new problem, FITS presents a page with 3 sections. The "Symbol" section, the "Premise" section and "Conclusion" section, and all of them with "New", "Edit" and "Delete" button.

The teacher can create arguments and formalize them. In the main window a list of all the arguments in natural language appears, together with the right formal equations written in a mathematical way.



SIMBOLOGIA

◯ agatha	Agatha
◯ butler	Butler
◯ charles	Charles
◯ lives(x)	x lives in Dreadbury Mansion



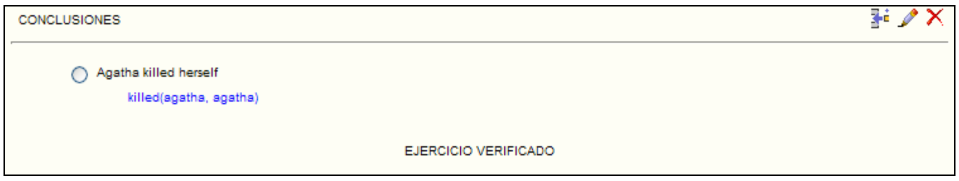
PREMISAS

Someone who lives in Dreadbury Mansion killed Aunt Agatha.

$\exists$ x (lives(x) $\wedge$ killed(x, agatha))

Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein.

lives(agatha) $\wedge$ lives(butler) $\wedge$ lives(charles) $\wedge$ ($\forall$ x (lives(x) $\rightarrow$ (x = agatha $\vee$ x = butler $\vee$ x = charles)))



CONCLUSIONES

Agatha killed herself

killed(agatha, agatha)

EJERCICIO VERIFICADO

During edition the expert module validates syntactically the formalization of each one of the sentences, and the correction of the whole argumentation. At the end, FITS adds the new problem to its database to make it available for students.

FITS offers other services for teachers, as the student's history and its evaluation. Also it can do an evaluation about the hardness of the problem. These services are being incorporated currently.

5. Conclusions and future works

In this work, we present *FITS*, the first intelligent tutor for teaching formalization with semantics correction.

Our projects of future works are leaded in two lines: to incorporate FITS during the present course 2006-07 on teaching the subject of "Computational logic" and to enlarge their services, from the point of view of their automatization, adaptation and evaluation.

References

[1] Logic software and logic education. http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html

[2] D. Abraham,L. Crawford, L.Lesta, A. Merceron, and K. Yacef. The logic Tutor: A multmedia presentation. Electronic Journal of Computer-Enhanced Learning. October 2001.

[3] R. Bornat and B. Sufrin. Jape: A calculator for animating proof-on-paper. In W. Mccune, editor, Proceedings of 14th International Conference on Automated Deduction, volume 1249 of Lecture Notes in Artificial Intelligence, pages 412-415. Springer, 1997

[4] M. Huth and M. Ryan. LICS web tutor. http://www.cs.bham.ac.uk/research/projects/lics/tutor.

[5] L. Lesta and K. Yacef. An intelligent teaching assistant system for Logic. In S.A. Cerri, G. Gouardères, and F. Paraguaçu, editors, ITS 2002, volume 2363 of Lecture Notes in Computer Science, pages 421-431. Springer-Verlag, 2002.

[6] S. Lukins, A. Levicki, and J. Burg. A tutorial program for propositional logic with human/computer interactive learning. In SIGCSE'02, pages 381-385. ACM, 2002.

[7] W. McCune. OTTER 3.3 reference manual. Argonne National Labratory, 2003

[8] W. Schwarz. Tree Proof generator. http://www.umsu.de/logik/trees.

[9] J. Wielemaker. SWI-Prolog 5.6.18 Reference manual. University of Amsterdam, 2006.

[10] A. Wildenberg and C. Scharff. OLIVER: an OnLine Inference and VERification system. In 32th ASEE/IEEE Frontiers in Education Conference. IEEE, 2002.