XV JENUI. Barcelona, 8-10 de julio de 2009
ISBN: 978-84-692-2758-9
http://jenui2009.fib.upc.edu/

FOLMO: First Order Logic in Moodle

José Luis Romero
Universidad de Huelva
Campus Univ. de La Rábida
21071 La Rábida
joseluis.romero@alu.uhu.es

Iñaki Fernández de Viana
Dpto. Tecnologías Información
Universidad de Huelva
Campus Univ. de La Rábida
21071 La Rábida
i.fviana@dti.uhu.es

Gonzalo A. Aranda
Dpto. Ciencias Computación
Universidad de Sevilla
E. T. S. Ingeniería Informática
41012 Sevilla
garanda@us.es

Resumen

La Lógica, dirigida a estudiantes universitarios de Ingeniería, aporta al alumno una visión razonada del aprendizaje basada en la formalización del conocimiento y en la automatización de distintas formas del razonamiento humano [6].

Con el objetivo de potenciar su aprendizaje nace First Order Logic in Moodle (FOLMO), un software de fácil integración con Moodle que permite el trabajo autónomo del alumno mediante la realización de ejercicios propuestos por el profesor.

1. Introducción

La Lógica de Primer Orden (LPO) se ha convertido en uno de los fundamentos matemáticos y en una base formal indispensable en todo ingeniero.

La formalización del conocimiento y la automatización del razonamiento son primordiales en muchas áreas de la Informática. La importancia de la lógica en los curriculums de Informática va tomando cuerpo propio, debido a sus aplicaciones en contextos específicos tales como la Programación, la Ingeniería del Software, el Diseño de Sistemas de Bases de Datos y la Inteligencia Artificial [6].

Si bien en los últimos años han aparecido diversos libros adaptados a la enseñanza de la lógica para Informáticos, los alumnos que dan sus primeros pasos con la LPO encuentran como principales dificultades la representación y comprensión de este lenguaje, así como la traducción del lenguaje natural a las reglas de la LPO [2].

Como complemento a estos textos, en Internet encontramos diversas páginas, como la Web de Logic software and logic education [17], con numerosas herramientas especializadas en la enseñanza de la LPO. Entre ellas podemos destacar Jape [14], LogicTutor [5], LogicITA [7], PlogicTutor [15] o Online Inference and Verification system (OLIVER) [4] que intentan ayudar al alumno con el aprendizaje de la LPO.

Los principales problemas que presentan casi todas estas herramientas son que no se centran en el lenguaje de la LPO sino en cómo construir pruebas formales mediante el uso de las reglas de deducción [14] y se limitan a la lógica proposicional [5, 7, 15, 4].

Sin embargo, existe una aplicación que proporciona todos los elementos necesarios para ayudar adecuadamente a un estudiante en el aprendizaje de la formalización en lenguaje LPO. Dicho software se denomina Knowledge Representation & Reasoning Tutor (KRRT) [2].

Si bien la potencia de KRRT es innegable, a nuestro entender presenta un grave problema: su integración con Modular Object-Oriented Dynamic Learning Environment (Moodle) [8] es extremadamente compleja. Con la idea de poder integrar fácilmente en Moodle una aplicación que comparta las mismas cualidades de KRRT nace FOLMO (First Order Logic in Moodle). Podemos definir a FOLMO como un paquete software basado en KRRT de fácil integración con Moodle, que permite a los profesores diseñar y proponer ejercicios online a los alumnos para facilitarles la comprensión del

lenguaje de la LPO.

Este artículo lo hemos estructurado de la siguiente manera: primero vamos a describir las características de KRRT. A continuación veremos las propiedades de Moodle. Seguidamente describiremos las características y el funcionamiento de FOLMO. Por último terminaremos con las conclusiones de esta aplicación.

2. Knowledge Representation & Reasoning Tutor (KRRT)

Knowledge Representation & Reasoning Tutor (KRRT) es un sistema basado en la Web que utiliza OTTER [19] como mecanismo de razonamiento automático, y que ayuda al estudiante en el aprendizaje de la LPO como un lenguaje de representación y razonamiento del conocimiento. Además ayuda a los estudiantes en la formalización del lenguaje natural al lenguaje de la LPO.

Para los autores de KRRT la realización de ejercicios es una manera de aprender a representar el conocimiento. Dado que las herramientas existentes no permiten al alumno buscar alternativas equivalentes a la formalización dada por el profesor, desarrollaron KRRT.

KRRT usa diferentes estrategias para solucionar tanto el problema de la representación como el del razonamiento:

1. En la etapa de representación hace uso de la arquitectura Formalization with an Intelligent Tutor System (FITS) [1], basado en el uso de sistemas automatizados de razonamientos, tales como OTTER [19] y Vampire [3], para demostrar la exactitud de la formalización escrita por parte de los usuarios.

2. En la etapa de razonamiento opta por la generalización del procedimiento de OLIVER [4], extendiendo su poder de razonamiento de la Lógica Proposicional a la LPO con igualdad.

En la Figura 1 vemos la arquitectura modular de KRRT. Consta de dos interfaces, una para el alumno y otra para el profesor.

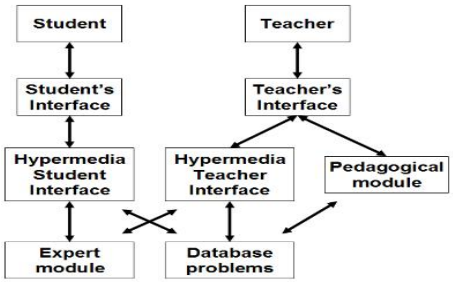

Figura 1: Arquitectura FITS

Dichas interfaces están comunicadas entre si gracias a la Base de Datos, donde se almacenan los ejercicios y las respuestas, y el módulo experto que es el encargado de comprobar, gracias a OTTER, si las respuestas del alumno son equivalentes a las respuestas del profesor.

3. Modular Object Oriented Dynamic Learning Environment (Moodle)

Como se indica en la propia documentación de Moodle [9]: *"Moodle es un paquete de software para la creación de cursos y sitios Web basados en Internet. Es un proyecto de desarrollo diseñado para dar soporte a un marco de educación social constructiva".*

Estamos hablando pues de un Learning Content Management System (LCMS), un sistema de gestión de contenidos (CMS) que se utiliza para la enseñanza [16].

Las características que proporciona Moodle, y que le hacen ser un referente entre los LCMS, son: es software libre licenciado bajo la Licencia Pública GNU, promueve una pedagogía constructiva social (colaboración, actividades, reflexión crítica, etc.), tiene gran disponibilidad, es estable, fácil de usar y seguro.

Pero hemos de decir que una de las características más importantes que Moodle presenta es su arquitectura modular, es decir, permite aumentar las prestaciones del mismo mediante la inserción de nuevos módulos. Debido a esto podemos encontrar en la página Web de Moodle [10] multitud de módulos disponibles

para esta plataforma implementados por los desarrolladores de Moodle y por terceras empresas y/o personas.

Tras un estudio detallado de todos y cada uno de ellos podemos afirmar que no hay un módulo específico para el aprendizaje del lenguaje LPO.

4. First Order Logic in Moodle (FOLMO)

Según un estudio de uso de Moodle en las Universidades Españolas [12], la gran mayoría de ellas usan Moodle como plataforma de aprendizaje. En nuestro caso particular esto también se cumple, salvo en aquellas asignaturas que en sus guías docentes tocan conceptos de la LPO. En estas es KRRT la principal herramienta usada para la comprensión de este lenguaje. Para solucionar los problemas que conlleva el uso de dos herramientas distintas, pensamos en la posibilidad de unificarlas.

Tras estudiar diversas posibilidades llegamos a la conclusión que la integración de KRRT en Moodle no es posible ya que, como hemos visto en los apartados anteriores, KRRT y Moodle usan arquitecturas diferentes.

Entonces, nos planteamos el desarrollo de FOLMO. Para ello vimos dos opciones que nos permitían su integración en Moodle:

1. Crear un nuevo módulo. El principal problema que presenta esta solución, es su realización siguiendo las normas que Moodle nos ofrece [11]. Esto conlleva un alto coste de desarrollo, ya que hay que implementar los archivos y funcionalidades del módulo partiendo desde cero.

2. Aumentar las características de un módulo existente. Esto siempre que no tengamos que modificar el código de los archivos que compongan dicho módulo.

De las dos posibles soluciones, hemos implementado la segunda. Entre todos los módulos existentes en Moodle (módulo de tareas, foro, encuesta, etc.) consideramos que el módulo cuestionario, por sus características, es el idóneo para integrar nuestro sistema.

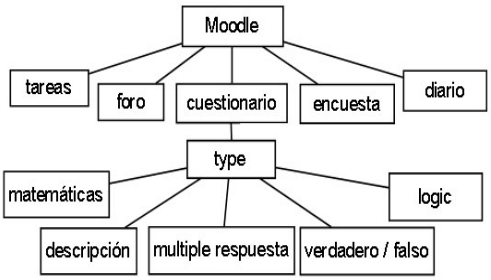

Figura 2: árbol jerarquía de Moodle

4.1. El módulo cuestionario

Este módulo ofrece la posibilidad de diseñar y plantear cuestionarios que se califican automáticamente y en el momento [9].

Además dispone de varios tipos de preguntas: preguntas verdadero/falso, de múltiple elección, de respuestas cortas, etc. De todos estos tipos ninguno se adapta al aprendizaje del lenguaje de la LPO, ya que:

1. No hacen uso de una herramienta para comprobar la corrección de las respuestas escritas en lenguaje de la LPO.

2. El módulo cuestionario usa un editor que se carga por defecto, el cual no contiene los símbolos necesarios a usar en el lenguaje de la LPO.

Los distintos tipos de preguntas se encuentran divididos por carpetas. Con esta organización la inserción de un nuevo tipo de pregunta se realiza mediante la creación de una nueva carpeta dentro de este módulo.

Cada carpeta consta de ficheros comunes a implementar (ver la Figura 3), pero a la vez independientes del resto de tipos de preguntas.

Los ficheros editquestion.html y editquestion.php son los encargados de realizar la interfaz del profesor, tanto para crear una nueva pregunta como para editar una existente.

El fichero display.html es el encargado de realizar la interfaz para el alumno. Los datos que muestra se obtienen gracias al fichero questiontype.php.

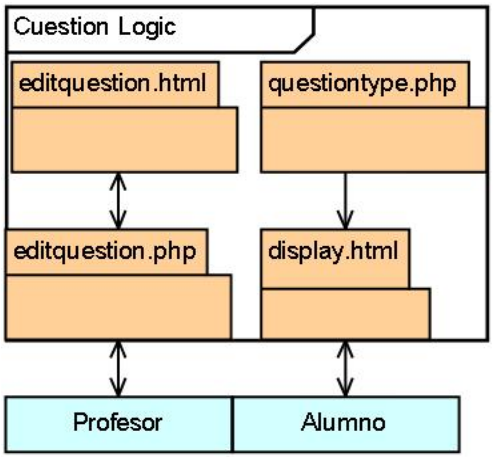

Figura 3: Archivos a implementar para un tipo de pregunta

El fichero questiontype.php es muy importante ya que define la clase del tipo de pregunta, que en nuestro caso es question_logic_qtype, y además es el encargado de crear una instancia de nuestro tipo de pregunta y hacerlo visible para el profesor en el menú de selección del nuevo tipo de ejercicio.

4.2. Implementación de FOLMO

Para crear FOLMO como un nuevo tipo de pregunta en Moodle, lo primero que hicimos fue crear una carpeta llamada logic dentro de la carpeta type que está en la carpeta question del directorio de Moodle. En esta carpeta creamos los ficheros que vimos en la Figura 3.

Además creamos en la Base de Datos las tablas: logic, premisas, conclusiones, variables y predicados. Especial explicación necesita la tabla logic que es donde se guarda la información general del tipo de ejercicio como el identificador, la pregunta, las respuestas de las premisas y de la conclusión.

A continuación vamos a explicar la funcionalidad que hemos implementado en los ficheros. Empezaremos con el fichero que tiene la funcionalidad del módulo, después explicaremos el fichero del alumno y seguidamente los ficheros que forman la parte del profesor.

En el fichero questiontype.php creamos la clase question_logic_qtype e implementamos las funciones que aparecen en el diagrama UML de la Figura 4.

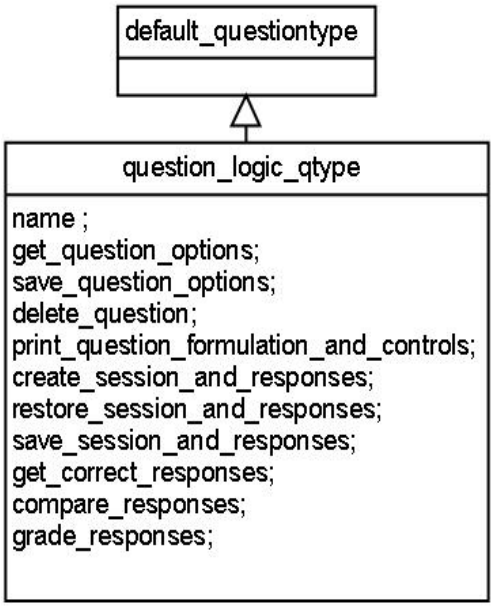

Figura 4: Clase question_logic_qtype

Con la llamada a la función save_question_options almacenamos en las tablas propias del módulo el ejercicio introducido por el profesor. Antes de guardarlo, verificaremos que es correcto mediante la llamada a OTTER.

Para verificar un ejercicio e=$\{(p_1..p_n)(c_1..c_m)\}$, siendo p_i las premisas y c_j las conclusiones introducidas por el profesor, llamamos a OTTER mediante otter$(p_1, p_2, .., p_n, -c_1, -c_2, .., -c_m)$ y nos devolverá si ha habido error o no.

Mediante la llamada a delete_question, el profesor borra el ejercicio seleccionado.

La función print_question_formulation_and_controls carga las opciones del ejercicio para su realización por parte del alumno.

La función save_session_and_responses es llamada para almacenar en las tablas de la Ba-

se de Datos las respuestas del alumno, una vez haya finalizado el ejercicio.

La función grade_responses se encarga de calcular la nota del alumno. Para ello, hace uso de la función compare_responses, y una vez ha obtenido el resultado de las comparaciones devuelve la nota que se ha obtenido.

La función compare_responses compara las respuestas del alumno r=$\{(p_1^1..p_n^1)(c_1^1..c_m^1)\}$, siendo p_i^1 las premisas y c_j^1 las conclusiones, con las dadas en el ejercicio e (definido anteriormente) del profesor. Para ello vamos invocando a OTTER de la siguiente manera otter$(-(p_1^1 \leftrightarrow p_1)) \wedge .. \wedge$ otter$(-(p_n^1 \leftrightarrow p_n)) \wedge$ otter$(-(c_1^1 \leftrightarrow c_1)) \wedge .. \wedge$ otter$(-(c_m^1 \leftrightarrow c_m))$ y nos va devolviendo en cada caso si son equivalentes las respuestas o no.

Añadimos al fichero questiontype.php la funcionalidad para que añada el tipo de pregunta al menú desplegable de selección de tipo de cuestión para que el profesor pueda seleccionarlo.

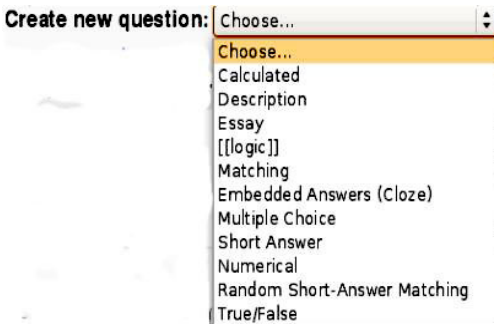

Figura 5: Menú de selección del tipo de ejercicio

En la parte del alumno, implementamos el archivo display.html, que creará dinámicamente la interfaz del alumno con los datos obtenidos al cargar el ejercicio. Declaramos tantos editores como premisas y conclusiones conste el ejercicio.

Este editor está basado en TinyMCE [13], y reemplaza al editor por defecto de Moodle, facilitando la escritura de premisas y conclusiones gracias a la barra de herramientas que hemos personalizado. Consta de los símbolos básicos de la LPO y su inserción se realiza mediante un click en uno los botones (ver Figura 7).

En display.html las comparaciones de las respuestas del alumno son independientes del resto de premisas, comparamos mediante OTTER la premisa/conclusión seleccionada por el alumno con la dada por el profesor, siendo la llamada otter$(-(z_i^1 \leftrightarrow z_i))$, z_i^1 es la premisa/conclusión del alumno y z_i la dada por el profesor.

En la parte del profesor, el fichero editquestion.php obtiene los valores de inicialización de la pregunta, siendo estos: los símbolos (variables y premisas), predicados, conclusiones y el nombre del ejercicio. Con estos datos editquestion.html creará la interfaz, insertando dos editores personalizados, uno para insertar las premisas y otro para insertar las conclusiones. Estos editores son iguales a los vistos en la parte del alumno.

Las premisas y conclusiones introducidas se muestran en una tabla debajo de cada editor, permitiendo al profesor editar o borrar cada una de ellas.

En este fichero, la llamada a OTTER se realiza con todas las premisas y conclusiones introducidas.

La aplicación se puede poner tanto en inglés como en castellano.

4.3. FOLMO para el profesor

Cuando un profesor vaya a crear una nueva pregunta para su test verá que en el menú desplegable aparece el tipo logic tal como observamos en la Figura 5.

En la Figura 6 vemos la primera parte de la interfaz a la que este tipo da lugar.

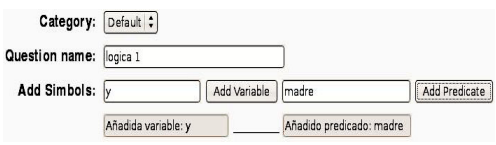

Figura 6: Detalle interfaz profesor. Añadir variable y predicado

Lo primero que debe hacer el profesor es introducir el nombre de la pregunta (Question

name).

Después pasará a declarar los símbolos, variables y predicados, que usará tanto en las premisas como en las conclusiones. El profesor debe saber que tanto las variables como los predicados tienen restricciones al declararlas: empezar con letra, no ser una palabra reservada ni haber sido declarada en el ejercicio, no contener espacios y estar compuestas sólo por caracteres. Si ha habido o no un error en la declaración, nos aparecerá un mensaje en el correspondiente campo que se encuentra debajo de la introducción del texto, como podemos observar en la Figura 6.

El siguiente paso consiste en ir creando las premisas. En la Figura 7 vemos los elementos que hemos creado para esto.

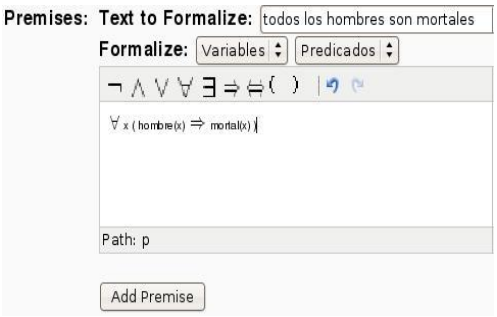

Figura 7: Interfaz Profesor. Detalle editor premisas

Tenemos un campo para insertar el texto a formalizar (Text to Formalize), dos listas desplegables que se van actualizando cada vez que añadimos una nueva variable o predicado y un editor personalizado, explicado anteriormente, en donde escribiremos las cláusulas.

Una vez que hemos escrito la formalización en el editor, pulsamos en el botón Add Premise que aparece en la Figura 7 para almacenarla temporalmente hasta que se envíe el formulario.

Al terminar de insertar las premisas, el profesor tiene que añadir las conclusiones.

En la Figura 8 observamos los elementos existentes para insertar una conclusión.

Estos elementos son los mismos que en la inserción de una premisa, al igual que el mecanismo de insertarla, debiendo pulsar sobre el botón Add Conclusion para añadirla a la lista.

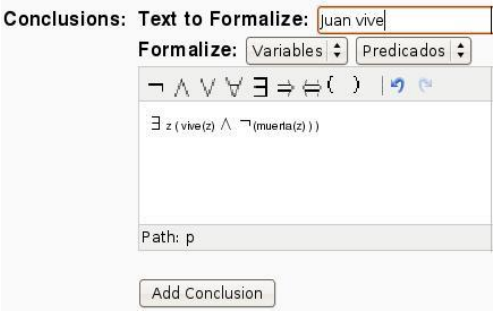

Figura 8: Interfaz profesor. Detalle editor conclusiones

Una vez definidas las premisas y las conclusiones, observamos en la Figura 9 la existencia de un botón "Comprobar", que es el encargado de validar el ejercicio como hemos visto anteriormente.

Figura 9: Interfaz profesor. Detalle botón comprobar y mensaje

Apreciamos en la Figura 9 el mensaje de la formalización, siendo el primer texto en caso correcto y el segundo en caso de error.

Una vez que el profesor ha terminado de introducir los símbolos, las premisas y las conclusiones, tiene que pulsar en el botón Save Change para terminar el ejercicio.

4.4. FOLMO para el alumno

El alumno al entrar en el ejercicio se encuentra con los editores de las premisas y de la conclusión a la que tiene que llegar. Debe de ir respondiendo a las preguntas en sentido descendente. Aunque el orden no influye en el resultado de la prueba, si puede influir en una mejor comprensión por parte del alumno.

Como apreciamos en la Figura 10 encontramos un campo Formalize donde muestra el texto a formalizar por el alumno, teniendo que

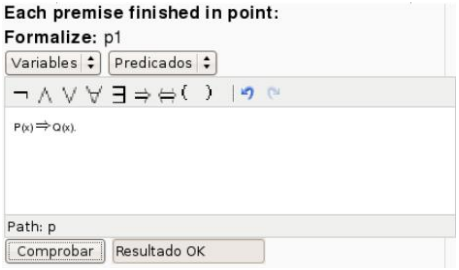

Figura 10: Interfaz alumno. Detalle editor premisas

escribir la solución en el correspondiente editor. Este editor es idéntico al del profesor. Al seleccionar un campo de una de las listas mostrará su valor en el editor al que pertenezca la lista.

Debajo del editor, el alumno tiene a su disposición un botón, Comprobar, cuya función es la de comparar su formalización con la dada por el profesor. El resultado se mostrará en el lado derecho del botón, como podemos apreciar en la Figura 10.

El editor para las conclusiones es igual que para las premisas, como podemos observar en la Figura 11.

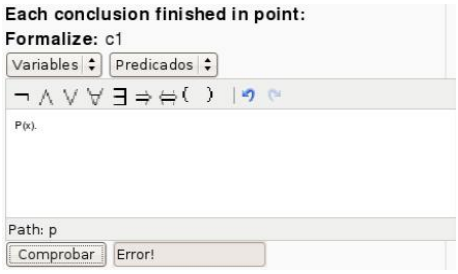

Figura 11: Interfaz alumno. Detalle editor conclusiones

Los elemento de una conclusión y el mecanismo para insertarla es idéntico al de insertar una premisa.

En la Figura 10 vemos una formalización correcta, mientras que en la Figura 11 se ha encontrado algún error.

Una vez haya terminado el alumno el ejercicio, tendrá que hacer click en el botón Submit de la Figura 12, en este caso la finalización del ejercicio es correcta y se muestra la nota que ha obtenido el alumno, 1/1.

Figura 12: Terminación del ejercicio mostrando la nota obtenida

5. Conclusiones

Este trabajo es la presentación de FOLMO como herramienta para ayudar a los alumnos en el aprendizaje de la formalización del lenguaje natural al lenguaje de la LPO.

Además hemos conseguido una herramienta de fácil integración en Moodle que dispone de la misma funcionalidad que KRRT, y con la que evitamos dualidad de herramientas para el aprendizaje de la LPO.

Otro aspecto a destacar es el fácil manejo de FOLMO, tanto por la creación simple e intuitiva de ejercicios por parte de los profesores, como la flexibilidad y facilidad de uso en la realización de los ejercicios por parte del alumno.

No obstante, la interfaz es mejorable con el uso de nuevas tecnologías como es Ajax [18], ya integrada en Moodle 1.9, permitiéndonos añadir nuevas funcionalidades como son: el uso de teclas rápidas en conectores y cuantificadores de la LPO, analizar posibles errores léxicos y semánticos de forma online antes de añadir la premisa/conclusión, detección de variables y predicados repetidos, así como la inserción de una variable/predicado mediante texto predictivo.

Un posible problema que nos podemos encontrar, es el uso de Ajax para integrar alguna de estas funcionalidades, ya que la versión que integra Ajax en Moodle es la 1.9, y FOLMO está pensado para la versión 1.6 de Moodle.

Actualmente FOLMO se está probando en diversas asignaturas de la Universidad de Huelva en el curso 2008/2009, y se intentará implantar para el curso 2009/2010 en el res-

to de asignaturas, donde se estudiará el rendimiento académico obtenido por los alumnos mediante la realización de estos ejercicios.

Referencias

[1] A. Alonso, G. A. Aranda, F. J. Martín-Mateos, *FITS: Formalization with an Intelligent Tutor System*, IV International Conference On Multimedia And Information And Communication Technologies In Education, 2006.

[2] A. Alonso, G. A. Aranda, F. J. Martín-Mateos, *KRRT: Knowledge Representation and Reasoning Tutor System.* Lecture Notes in Computer Science, volume 4739, EUROCAST, 400–407, Springer, 2007.

[3] A. Riazanov, A. Voronkov, *Vampire 1.1 System Description*, Lecture Notes in Computer Science, volume 2083, Proceedings of the First International Joint Conference on Automated Reasoning, IJCAR 2001, pages 376–380, Springer Verlag, 2001.

[4] A. Wildenberg, C. Scharff, *OLIVER, an OnLine Inference and VERification system.* In 32 ASEE/IEEE Frontiers in Education Conference. IEEE Computer Society, 2002.

[5] D. Abraham, L. Crawford, L. Lesta, A. Merceron, K. Yacef, *The Logic Tutor: A multimedia presentation*, Electronic Journal of Computer Enhanced Learning, 2001.

[6] F. Llorens, M.J. Castel, Lógica de Primer Orden en las Ingenierías Informáticas, volume 2, I Jornadas Nacionales de Innovación en las Enseñanzas de las Ingenierías, pages 840–847, Instituto de Ciencias de la Educación, Universidad Politécnica de Madrid, 1996.

[7] L. Lesta, K. Yacef, *An intelligent teaching assistant system for Logic*, Lecture Notes in Computer Science, volume 2363,In S. A. Cerri, G. Gouardéres, and F. ParaguaÇu, editors, ITS 2002, Springer Verlag, 2002.

[8] M. Dougiamas, *Modular Object-Oriented Dynamic Learning Environment (Moodle)*, disponible en `http://moodle.org/`.

[9] M. Dougiamas, varios autores, Documentación del módulo cuestionario de Moodle, disponible en `http://docs.moodle.org/es/Cuestionarios`.

[10] M. Dougiamas, varios autores, Módulos y plugins para Moodle, disponible en `http://moodle.org/mod/data/view.php?id=6009`.

[11] M. Dougiamas, varios autores, *Normas para el desarrollo de módulos de Moodle*, disponible en `http://docs.moodle.org/en/Development:Modules`.

[12] M. MOLIST, *Moodle llena la geografía educativa española de campus virtuales*, Diario El Pais, disponible en `http://www.elpais.com/articulo/portada/Moodle/llena/geografia/educativa/espanola/campus/virtuales/elpepisupcib/20081204elpcibpor_1/Tes/`.

[13] M. Systems AB, *TinyMCE 3.2.1.1*, disponible en `http://tinymce.moxiecode.com/`.

[14] R. Bornat y B. Sufrin, *Jape: A calculator for animating proof-on-paper*, Lecture Notes in Artificial Intelligence, volume 1249, Proceedings of the 14th International Conference on Auto-mated deduction Springer, 1997.

[15] S. Lukins, A. Levicki, J. Burg, *A tutorial program for propositional logic with human/computer interactive learning*, In SIGCSE02, pages 381–385. ACM, 2002.

[16] Varios Autores, Learning Content Management System (LCMS), disponible en `http://es.wikipedia.org/wiki/LCMS`.

[17] Varios Autores, Logic software and logic education, disponible en `http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html`.

[18] Varios Autores, Asynchronous JavaScript And XML (Ajax), disponible en `http://es.wikipedia.org/wiki/AJAX`.

[19] W. McCune, *OTTER 3.3 reference manual*, Argonne National Laboratory, 2003.