View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by El Servicio de Difusión de la Creación Intelectual

Herramientas Educativas para la Enseñanza de la Lógica

Maria Virginia Mauco[1], Laura Felice[1], María Carmen Leonardi[1]

[1]Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Tandil, Argentina
{vmauco, lfelice, cleonard}@exa.unicen.edu.ar

Resumen. El software educativo ayuda a motivar y mejorar el proceso de enseñanza/aprendizaje. En este artículo se presenta un proyecto de desarrollo de herramientas didácticas, interactivas y visuales para cursos de Lógica dictados en los primeros años de carreras de Informática. Las herramientas fueron desarrolladas por alumnos de 2º año de la carrera como Trabajo Final de dos materias, una que introduce conceptos de Lógica para Computación y otra que aborda las bases de las técnicas de diseño de algoritmos, permitiéndoles integrar y profundizar en los contenidos de ambas.

Palabras clave: Software Educativo; Lógica Proposicional; Lógica de Predicados de Primer Orden; Técnicas de Diseño de Algoritmos

1 Introducción

La mayoría de las carreras de Computación incluyen el dictado de cursos básicos de Lógica. En estos cursos, los alumnos deben resolver gran cantidad de ejercicios para adquirir práctica en el manejo de formalismos. El uso de herramientas didácticas que acompañen el proceso de enseñanza/aprendizaje, sin requerir demasiado tiempo de estudio para su utilización, puede ser un gran aporte.

El plan de estudios de la carrera de grado Ingeniería de Sistemas de nuestra Facultad incluye un curso introductorio en Lógica Proposicional (LP) y Lógica de Predicados de Primer Orden (LdPPO) [1, 2] que se dicta en el primer semestre del segundo año de la carrera [12]. En un principio, la ejercitación práctica de los temas enseñados en este curso se realizaba con papel y lápiz, siendo difícil, y en muchos casos tedioso para los estudiantes detectar y corregir errores. En consecuencia, los estudiantes consideraban a estos cursos menos atractivos, y en algunos casos más difíciles que otros en los que interactuaban con la computadora. Como el feedback inmediato, la interacción y la visualización pueden ayudar a motivar y mejorar el proceso de enseñanza/aprendizaje, se consideró la posibilidad de brindar a los estudiantes herramientas didácticas que se ajustaran a los contenidos, la notación y la metodología seguidos en el curso, y que fueran sencillas e intuitivas en su uso.

También, el plan de estudios de la carrera incluye en el primer semestre del segundo año un curso de Análisis y Diseño de Algoritmos (AyDA) [11] que brinda a los estudiantes conceptos fundamentales tales como Programación Recursiva, Tipos de

Datos Abstractos (TDA), Complejidad Computacional, y Técnicas de Diseño de Algoritmos: Backtracking, Divide y Conquista, Programación Dinámica, Método Greedy [4]. Al finalizar este curso, los estudiantes deben realizar un proyecto integrador, aplicando los principales temas aprendidos. Se consideró entonces beneficioso realizar un proyecto que desarrollara una herramienta didáctica para la materia de la carrera que introduce los conceptos fundamentales de LP y LdPPO.

Teniendo en cuenta lo anteriormente mencionado, se propuso definir un proyecto de desarrollo de herramientas educativas para asistir en el proceso de enseñanza/aprendizaje de los temas abordados en el curso de Lógica. Cada una de estas herramientas fue desarrollada por un equipo de 2 ó 3 estudiantes del 2º año de la carrera, como trabajo final de los cursos de AyDA y de Lógica. En este marco, se desarrollaron varias herramientas didácticas, visuales e interactivas que se ajustan a los contenidos, la notación y la metodología seguidos en el curso de Lógica, y que son fáciles e intuitivas en su uso [3, 5, 6, 9, 10, 17, 18].Estas herramientas son utilizadas como soporte en el proceso de enseñanza/aprendizaje, tanto por docentes como por estudiantes del curso de Lógica.

En este trabajo se describen brevemente algunas de las herramientas desarrolladas. Cada herramienta permite experimentar con algunos de los contenidos del curso de Lógica, y a la vez para algunos contenidos se ha desarrollado más de una herramienta con el objetivo de abordarlos con un "enfoque" complementario, y en algunos casos, diferente.

Este artículo está estructurado de la siguiente manera. En la Sección 2 se describen los contenidos de las materias en cuestión: Análisis y Diseño de Algoritmos y Ciencias de la Computación II. En la Sección 3 se presentan las características principales de las herramientas desarrolladas y se describe brevemente, por razones de espacio, la funcionalidad de cada una. La Sección 4 describe algunas experiencias en el uso de las herramientas. Finalmente, se presentan las conclusiones sobre el trabajo realizado en la Sección 5.

2 Materias Involucradas en el Desarrollo de las Herramientas

En esta sección, se describen brevemente los contenidos de las dos materias involucradas en el desarrollo de las herramientas. Ambas materias se dictan en el primer semestre del 2º año de la carrera.

2.1 Ciencias de la Computación II

Es un curso introductorio a la LP y a la LdPPO que se dicta en el primer semestre del segundo año de la carrera. Los contenidos del curso están orientados a brindar a los estudiantes fundamentos de lógica para computación. Para esto, se introducen la LP y la LdPPO, abarcando por un lado los conceptos más clásicos de la lógica (sintaxis, semántica, métodos de deducción y demostración, resultados de corrección y completitud, satisfacibilidad y validez, formalización de especificaciones en lenguaje natural), y por otro, los fundamentos para algunas aplicaciones de la lógica en computación: verificación de software, programación lógica, bases de datos, etc.

Las clases se dictan en la modalidad teórico-práctica. Se presenta semanalmente un núcleo temático, utilizando filminas elaboradas por los docentes, acompañadas de ejercicios desarrollados en pizarrón con la participación de los estudiantes. Los estudiantes desarrollan los ejercicios propuestos, consultando a los auxiliares de la materia, o con la asistencia de las distintas herramientas didácticas que fueron desarrolladas para ser usadas en la materia por otros estudiantes de la carrera [12].

2.2 Análisis y Diseño de Algoritmos I

Entender un problema en términos computacionales e intentar escribir un programa que lo resuelva es el primer escollo que el estudiante debe confrontar al abordar problemas del mundo real. En general, ellos se encuentran con que la descripción de este tipo de problemas es algo acotada, y los primeros obstáculos surgen al intentar separar los detalles necesarios de los innecesarios para obtener una vista abstracta o modelo del problema.

El objetivo del curso de Análisis y Diseño de Algoritmos está centrado en la introducción de las bases para el desarrollo de software como lo es la especificación de TDAs, la integración de especificaciones e implementaciones, la construcción de librerías de componentes y la reusabilidad del software desarrollado. Para lograr este objetivo, es necesario abstraer el diseño de una manera apropiada, eficiente e implementable para resolver los problemas. Esta tarea sin buenas prácticas de desarrollo puede resultar engorrosa, es necesario que el programador acceda a los aspectos principales del conocimiento: las técnicas de diseño. Los buenos diseños incluyen técnicas básicas como la búsqueda Exhaustiva, Divide y Conquista, el Método Greedy, la Programación Dinámica y otras heurísticas junto con estructuras de datos adecuadas.

Para este curso se considera que implementar soluciones para problemas relacionados con el desarrollo de herramientas para el aprendizaje, es un buen tópico para la enseñanza de estas técnicas. Así, los estudiantes pueden aplicarlas en proyectos reales, académicos, didácticos y también atractivos generando desafíos para lograr buenas soluciones. Se ha trabajado durante los últimos años en este tipo de proyectos integradores significando para el alumno un desarrollo a gran escala, donde se aplica la metodología propuesta para resolver un problema basado en una situación concreta y real [11].

3 Las Herramientas Desarrolladas

Todas las herramientas que se describen en esta sección fueron desarrolladas por estudiantes de la carrera al finalizar 2º año como Proyecto Final de las materias que involucran contenidos de lógica y análisis y diseño de algoritmos. Son herramientas didácticas, visuales e interactivas que se ajustan a la notación, metodología y contenidos de Ciencias de la Computación II, lo que hace que sean muy fáciles e intuitivas de usar para los estudiantes. Las herramientas son software libre bajo

licencias GNU GPL[7], siendo posible su descarga para uso, estudio o modificación del código fuente.

Las herramientas fueron implementadas en el lenguaje de programación C++, siguiendo un diseño orientado a clases. Mediante la identificación de TDAs (Tipos Abstractos de Datos), se abstrajeron características y comportamientos propios de los mismos que posteriormente se plasmaron en clases de C++. La implementación de cada herramienta tuvo en cuenta los aspectos necesarios para lograr la mayor eficiencia en cuanto a la complejidad computacional de los algoritmos desarrollados.

Para la interfaz gráfica de todas las herramientas se usó el framework Qt [16], que brinda un entorno completo y documentado que permite la implementación de interfaces gráficas intuitivas y amigables. Las interfaces fueron desarrolladas inicialmente en español, pero pueden ser traducidas al inglés fácilmente usando un conjunto de herramientas provistas por Qt. El desarrollo de estas interfaces resultó tan desafiante como productivo ya que no forma parte del contenido de las materias en cuestión.

Todas las herramientas cuentan con Ayuda on-line para el usuario y mediante mensajes de error brindan al usuario la información necesaria ante la presencia de un error.

En las siguientes subsecciones se describe brevemente la funcionalidad de cada una de las herramientas desarrolladas y, por razones de espacio, se muestran solamente algunas interfaces de las mismas.

3.1 SAT y DP Solver

La herramienta SAT (Problema de la satisfacibilidad booleana) fue desarrollada para determinar la satisfacibilidad de un conjunto de cláusulas de LP de manera eficiente y para mostrar la importancia de la LP, ya que la mayoría de los estudiantes en sus primeros años carece del conocimiento de la cantidad de problemas que pueden ser resueltos mediante el uso de la lógica. Esta herramienta permite al usuario experimentar con dos problemas reales, como son el coloreo de un grafo y el de las N reinas, verificando que pueden resolverse como un problema de satisfacibilidad booleana [8, 18]. En la Figura 1 se puede observar una de las ventanas para el problema de las N reinas.

DPSolver se pensó como una herramienta didáctica e interactiva que sirviera como apoyo a la hora de afianzar conceptos de LP, particularmente la aplicación del algoritmo de Davis Putnam a partir de un conjunto de cláusulas para determinar su satisfacibilidad. DPSolver también permite calcular la forma normal conjuntiva para cualquier fórmula de LP dada, y la posibilidad de trabajar con consecuencias semánticas [3].

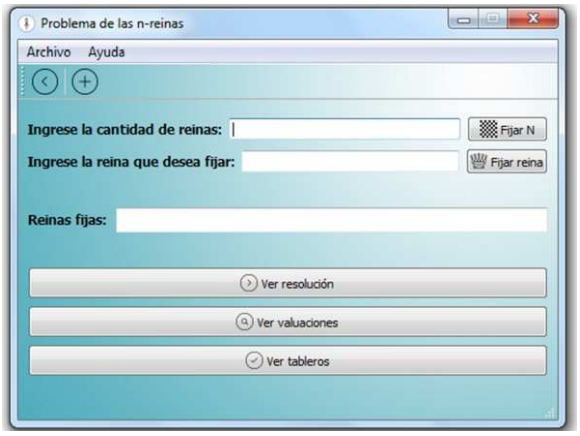

Figura 1. Satisfacibilidad booleana aplicado al problema de las N-reinas

3.2 YAT3

La herramienta YAT3 (Yet Another Truth Table Tool) permite trabajar con Árboles de Refutación tanto en LP como en LdPPO [17].Dentro del marco práctico de un curso de lógica, es habitual que se requieran construcciones de estos árboles a partir de fórmulas lógicas para poder determinar consecuencias lógicamente válidas y validez de fórmulas. De esta manera, la construcción de un árbol se desarrolla aplicando una serie de reglas hasta determinar si cada rama del mismo se puede cerrar o no. La herramienta considera tres modos distintos de construir un árbol de refutación. El primero consiste en ofrecer, a partir de una fórmula ingresada por el usuario, el Árbol de Refutación construido. El mismo se va mostrando en pantalla automáticamente hasta completarse. El segundo modo, consiste en ofrecer al usuario la posibilidad de generar el árbol paso por paso, una regla a la vez. De esta manera, se puede ir verificando el desarrollo del mismo y proceder a un análisis más lento de los pasos (reglas) que se ejecutan. El tercer modo, y de más interacción con el usuario, consiste en que él mismo decida la forma en que se genera el árbol, logrando así algo más cercano a lo que puede desarrollar con lápiz y papel.

3.3 FOLST y LogicChess

FOLST [10, 14] y LogicChess [9] son dos herramientas que permiten la evaluación sintáctica y semántica de fórmulas de LdPPO en modelos construidos por el usuario en el dominio provisto por cada herramienta.

FOLST permite definir modelos sobre dos dominios: Mundo y Granja. La Granja (Figura 2) consiste en una imagen de una granja en la que distintos animales (por ejemplo, vacas y gallinas) se pueden agregar en distintos lugares (el corral, el bosque, etc.). Cada animal tiene atributos (su especie, ubicación y si está durmiendo) y relaciones que permiten formalizar información real en este contexto. Se proveen once

predicados unarios, dos predicados binarios y una función. El Mundo provee un mapa dividido en continentes sobre el cual se pueden agregar ciudades (capitales o no) que se pueden conectar. Se definieron seis predicados unarios, cinco predicados binarios y una función.

LogicChess permite trabajar sobre un conjunto finito de modelos. Cada modelo representa un tablero de Ajedrez y está formado por Fichas, Números y Vacío (referencia a un casillero que no contiene fichas). El usuario puede crear y manejar de forma fácil estos modelos quitando, agregando y modificando cada componente. En LogicChess se definieron un total de quince relaciones clasificadas en: relaciones de identificación, relaciones de posición relaciones de distancia.

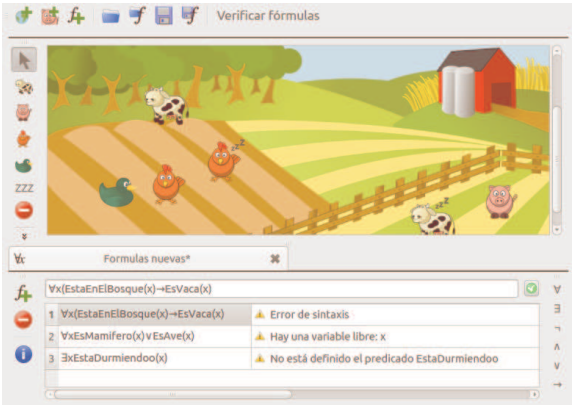

Figura 2. Mundo Granja definido por FOLST

3.4 Clausula y Clprover

Clausula permite a estudiantes y docentes experimentar con conjuntos arbitrarios de cláusulas de la LdPPO con el objetivo de determinar su (in)satisfacibilidad [6,13]. Para esto, Clausula implementa métodos clásicos y fundamentales de la LdPPO como el método de Resolución de cláusulas. Además, la herramienta brinda la posibilidad de calcular el unificador más general (umg) de un par de literales, usando el Algoritmo de Unificación de Robinson, y aplicar estrategias de simplificación al conjunto inicial de cláusulas, tales como eliminación de cláusulas tautológicas, eliminación de literales por idempotencia y simplificación de cláusulas por literales puros. Con respecto a la determinación de la satisfacilidad de un conjunto de cláusulas, la herramienta detecta si se trata de una Lógica de Programas, un conjunto de cláusulas de Horn o un conjunto arbitrario para aplicar la estrategia pertinente en cada caso.

Clprover es la otra herramienta desarrollada para determinar (in)satisfacibilidad de cláusulas de la LdPPO, agregando la posibilidad de trabajar también a partir de fórmulas arbitrarias. Además de brindar la funcionalidad provista por Clausula, Clprover permite simplificación por subsunción de cláusulas y factorización de cláusulas [15]. En la Figura 3 se muestra una captura de pantalla de la herramienta.

Figura 3. Método de Resolución en Clprover

3.5 TheM

La herramienta THeM (Teaching Herbrand Models) permite determinar la satisfacibilidad de un conjunto de cláusulas de la LdPPO a través de la existencia de Modelos de Herbrand [5]. Esto resulta importante ya que cualquier fórmula de esta lógica puede llevarse a forma clausular, y luego aplicar la herramienta para conocer su validez.

Con la intención de que la herramienta sea didáctica y de utilidad para los estudiantes, se implementó un algoritmo que muestra lo más claro posible el procedimiento que se realiza de forma manual para resolver este tipo de problemas. En primera instancia, se genera el conjunto S, es decir, el conjunto de cláusulas definidas y luego se definen el Universo y la Base de Herbrand. Luego de expresar la suposición de que existe un Modelo de Herbrand que satisface cada una de las cláusulas, se busca probar la existencia de éste. Para hacerlo se realiza la selección cláusula por cláusula de literales ya instanciados, evitando contradicción con aquellos que pertenezcan a la solución parcial. En caso de insatisfacibilidad, se realiza un procedimiento similar, utilizando los números de pasos para mostrar la contradicción que lleva a tal resultado. La Figura 4 muestra los posibles modelos para el conjunto de cláusulas ingresado.

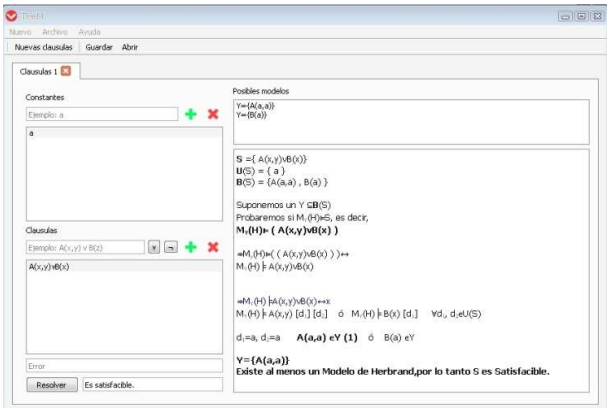

Figura 4. Modelos de Herbrand con THeM

4 Experiencia en el Uso y Desarrollo de las Herramientas

Las herramientas presentadas en la sección anterior se han utilizado en la materia Ciencias de la Computación II como software de apoyo, tanto por los docentes para introducir y ejemplificar algún tema durante las clases, como por los estudiantes para realizar la ejercitación práctica correspondiente a la materia. Los estudiantes han reportado experiencias positivas en el uso de las distintas herramientas (la materia tiene entre 100 y 120 estudiantes por año), manifestando que son fáciles de usar, y que les permiten aplicar los conceptos estudiados en la materia y comprobar sus resultados, participando así más activamente en su proceso de aprendizaje, y autoevaluando sus conocimientos con anterioridad a las instancias de evaluación.

En la actualidad, se dispone de al menos una herramienta para asistir en el proceso de enseñanza/aprendizaje de todos los contenidos de la materia de lógica: DPSolver y SAT para LP y sus aplicaciones; YAT3 para árboles de refutación en LP y LdPPO; FOLST y LogicChess para semántica en LdPPO; THeM para Modelos de Herbrand; Clausula y Clprover para experimentar con resolución en LdPPO. YAT3, THeM, Clausula y Clprover permiten además abordar, indirectamente, los conceptos de consecuencia semántica y validez de fórmulas en LdPPO.

En cuanto al desarrollo de las herramientas involucra la puesta en marcha de no sólo los programas diseñados con las técnicas adecuadas en cada caso e implementados en lenguaje C++, sino también la vinculación de este desarrollo con el entorno gráfico. El diseño independiente de la solución y la interfaz gráfica favorece al mantenimiento del software desarrollado y también al reuso del mismo. En muchos casos, se han abordado nuevos desarrollos, por ejemplo extensiones en cuanto a la funcionalidad de alguna herramienta, basados en implementaciones anteriores, resultando un trabajo ordenado y sistemático para el estudiante.

5 Conclusiones

En este trabajo se describió un proyecto de desarrollo de herramientas didácticas como trabajo final de dos materias del 2º año de la carrera.

El objetivo de estas herramientas es asistir a los estudiantes de la materia Ciencias de la Computación II en el proceso de enseñanza/aprendizaje de los distintos temas abordados en la misma. Las herramientas usan la misma notación y metodología seguidas en la materia, y son utilizadas como software de apoyo para los estudiantes al realizar la ejercitación práctica correspondiente a cada tema, y por los docentes en las clases para introducir y ejemplificar temas en los que cada herramienta brinda asistencia. Desde el punto de vista pedagógico, faltaría una evaluación más sistemática y formal del aporte de cada una de estas herramientas en el proceso de enseñanza/aprendizaje de los contenidos que cada herramienta aborda.

Para los estudiantes que desarrollaron estas herramientas, el aporte involucró dos aspectos: por un lado, haber tenido una primera experiencia de un desarrollo de software completo, desde la etapa de requisitos hasta la implementación, aprendiendo nuevas tecnologías y aplicando temas aprendidos en la materia Análisis y Diseño de Algoritmos, y por otro lado, la integración y profundización de los temas de lógica involucrados en la funcionalidad de la herramienta. Otro de los aportes para estos estudiantes, fue que tuvieron la posibilidad de presentarla herramienta desarrollada durante una clase de la materia Ciencias de la Computación II a los estudiantes de la misma. De esta forma, los desarrolladores tuvieron una primera experiencia de presentación oral a un grupo numeroso, permitiendo un intercambio con los estudiantes del curso, quienes se sintieron motivados al ver que estudiantes sólo un poco más avanzados que ellos fueron capaces de desarrollar un software completo con la formación adquirida hasta ese momento de la carrera. Además, los docentes de ambas materias alentaron a los estudiantes a presentar las herramientas desarrolladas en concursos de trabajos estudiantiles nacionales y en congresos nacionales e internacionales (en la descripción de cada herramienta se halla la referencia correspondiente). Esto permitió a los estudiantes tener una primera experiencia en la preparación de un artículo para un congreso así como también en la presentación del mismo.

Las herramientas tienen otra ventaja para la enseñanza: son software libre. Cuando se utiliza software libre en los procesos de enseñanza/aprendizaje, se da la posibilidad a los estudiantes de utilizar y compartir los recursos que éste les ofrece. Pero por sobre todo, permite potenciar las capacidades didácticas de los programas de computadora, permitiendo a los estudiantes explorar el software y analizar las implementaciones reales de los algoritmos propuestos. De esta forma, además de utilizarlas como un instrumento de aprendizaje, el estudiante puede ser partícipe en la construcción de sus propias herramientas.

Se considera que el desarrollo de un proyecto transversal a dos materias como el que se presentó en este artículo, permite que los estudiantes integren y profundicen en los contenidos de ambas, y no visualicen a las mismas como cursos aislados e independientes, sino que forman parte de un proceso que de manera incremental los irá formando en su carrera. Es por esta razón, que ambas cátedras van a seguir apostando a este tipo de propuestas integradoras.

Referencias

1. Arenas Alegría, L.: Lógica Formal para Informáticos. Ediciones Diaz de Santos (1996).
2. Ben-Ari, M.: Mathematical Logic for Computer Science. Springer-Verlag London (2012).
3. Cicconi, D., Fernández Cocirio, M. DPSolver: Una herramienta interactiva para la implementación del algoritmo de Davis Putnam. Simposio de Trabajos Estudiantiles, 2º CoNaIISI, Argentina, (2014).
4. Cormen, T.; Lieserson, C.; Rivest, R. Introduction to Algorithms. Ed. The MIT Press. 2009.
5. Dahl, J., Fujii, D. THeM: Una Herramienta Didáctica para Modelos de Herbrand. Aceptado para ser presentado en Concurso de Trabajos Estudiantiles, 45ºJAIIO, Argentina(2016).
6. Ferrante, E. Clausula: Herramienta Didáctica para la Enseñanza de Lógica de Predicados de Primer Orden. Concurso de Trabajos Estudiantiles, 38ºJAIIO, Argentina(2009).
7. GNU General Public License (GPL). http://www.gnu.org/licenses/gpl.html
8. Kelley, J. The Essence of Logic, Prentice Hall (1997).
9. Kiehr, A., Re Medina, M. LogicChess: Herramienta Didáctica para la Ejercitación en Lógica de Predicados de Primer Orden. Concurso de Trabajos Estudiantiles, 41ºJAIIO, Argentina(2012).
10. Maggiori, E., Gervasoni, L. FOLST: Una Herramienta Didáctica para la Lógica de Predicados de Primer Orden. Primer Premio en el Concurso de Trabajos Estudiantiles, 41ºJAIIO, Argentina(2012).
11. Materia Análisis y Diseño de Algoritmos I. http:// http://aydalgor.alumnos.exa.unicen.edu.ar
12. Materia Ciencias de la Computación II. http:// http://ccomp2.alumnos.exa.unicen.edu.ar
13. Mauco, M.V., Ferrante E.: Clausula: A Didactic Tool to Teach First Order Logic. In: Proceedings de ISECON 2009, Information Systems Education Conference, v26: §4142. WashingtonDC. USA. (2009).
14. Mauco, M.V., Maggiori, E., Gervasoni, L., Ferrante, E., Felice, L.FOLST: A Didactic Tool to Support First Order Logic Semantics Learning. In Proceedings of International Conference on Future Computers in Education. Shangai. China. Lecture Notes in Information Technology, Vols.23-24, 302-307 (2012).
15. Mauco, M.V., Moauro, L., Felice, L. Una Herramienta Didáctica para la Enseñanza de Lógica de Predicados de Primer Orden. En Anales del Congreso Iberoamericano de Educación Superior en Computación (CIESC 2010), Paraguay (2010).
16. Qt:Documentación oficial sobre el entorno gráfico. http://doc.qt.io
17. Ruau, K., Tosini, J.M. Yet.anotherTruth.Tree.Tool: Una herramienta didáctica sobre Árboles de Refutación. Concurso de Trabajos Estudiantiles, 44ºJAIIO, 63-71, Argentina(2015).
18. Santillán Cooper, M., Horquin, E., Covelli, T. SAT: Herramienta para la resolución del Algoritmo de Davis Putnam LogemannLoveland y problemas computacionales que pueden ser resueltos a partir de lalógica proposicional. Reporte Interno UNCPBA (2015).