Divulgaciones Matemáticas v. 6, No. 2 (1998), 165–182

Algunas Experiencias en la Utilización de Sistemas de EAC para la Enseñanza de la Inteligencia Artificial †

Some Experiences with the Utilization of CAI Systems in the Teaching of Artificial Intelligence

Mateo Lezcano Brito, Víctor Giraldo Valdés Pardo
uats@capiro.vcl.sld.cu
Universidad Central de Las Villas
Santa Clara, Cuba

Resumen

El presente trabajo expone las experiencias adquiridas en la utilización de dos sistemas para la enseñanza de la disciplina *Inteligencia Artificial* que se imparte como parte del Plan de Estudios de la carrera *Ciencias de la Computación* en la Universidad Central de Las Villas (UCLV), Cuba. El primero de ellos se denomina *Sistema para la Enseñanza de Sistemas Expertos* (SESE) y se utiliza para impartir tópicos relacionados con la programación de máquinas de inferencia y la comprensión de algunos mecanismos internos de los sistemas expertos. El segundo sistema se utiliza en la enseñanza del paradigma de la Programación Lógica.
Palabras y frases clave: inteligencia artificial, sistemas expertos, EAC, programación lógica.

Abstract

This paper gives an account of the experience adquired with the use of two systems for the teaching of the course *Artificial*

†Recibido 98/07/15. Aceptado 98/11/30.
MSC (1991): 68T35, 68T99.

Intelligence which is part of the *Computer Science* career at the Universidad Central de Las Villas (UCLV), Cuba. The first one is called *System for the Teaching of Expert Systems* and is used to teach topics related with the programming of inference machines and the understanding of some internal mechanisms of expert systems. The second one is used for the teaching of the Logic Programming paradigm.
Key words and phrases: artificial intelligence, expert systems, CAI, logic programming.

1 Introducción

La enseñanza de la Inteligencia Artificial debe vencer dos obstáculos fundamentales: el primero tiene que ver con el reducido tiempo que se le asigna en los planes de estudio y el segundo está relacionado con la diversidad de los temas. La solución puede parecer obvia si logramos añadirle más tiempo a los temas tratados, pero las cosas no son tan sencillas como para resolverlas con esa sola medida.

Muchos de los temas tratados en IA son conceptos y temas fundamentales compartidos con otras disciplinas de Ciencias de la Computación, tales como: Lógica Matemática, Prueba de Teoremas, Teoría de Probabilidades, etc. [6], el problema es que esos tópicos no se abordan en las asignaturas precedentes desde el punto de vista de la IA. Se hace necesario entonces establecer una vinculación entre todas las asignaturas relacionadas, lo que ayudará, en gran medida, a resolver algunos de los problemas tratados. En la inauguración del *AAAI Fall Symposium on Improving Instruction of Introductory AI*, Marti A. Hearst expresó: "Este simposio ha sido motivado por el deseo de dirigir las voces que lastimosamente se quejan de lo notoriamente difícil que resulta enseñar bien los cursos iniciales de Inteligencia Artificial ... "

El reconocimiento al problema de la enseñanza de la IA queda claro en estas palabras, en la sola idea de organizar un evento para discutir esta problemática y en el propio desarrollo que éste tuvo.

Con el aporte de nuestra experiencia pretendemos contribuir modestamente a la solución de este problema. Hemos seguido la idea de

fomentar un aprendizaje activo de manera que el estudiante no sea un mero recipiente que debe ser llenado de conocimientos ni el profesor el encargado de llenarlo.

2 Sistema para la Enseñanza de Sistemas Expertos (SESE)

El Sistema SESE está compuesto por los siguientes módulos:

- TeachShell. Una máquina de inferencia (MI) con fines docentes.
- UCShell. Un medio integrado para el desarrollo de sistemas expertos.
- Compiler. Un compilador de líneas que genera código para TeachShell y UCShell.

2.1 TeachShell

TeachShell es una herramienta que ilustra paso a paso el algoritmo que sigue su máquina de inferencia (dirigida por objetivos). El sistema recibe como entrada una base de conocimiento previamente cojmpilada por Compiler o UCShell.

La traza mostrada tiene una importancia vital en el proceso de enseñanza, ya que a través de ella el estudiante puede apreciar con claridad:

- Cómo se ejecutan los dos algoritmos más importantes de la máquina de inferencia.
- El recorrido que hace la MI por la base de conocimiento mientras trata de probar un objetivo.
- Cuáles son las reglas exploradas.
- Cuál es el objetivo que se está probando en un instante de tiempo (el objetivo actual).

- Cuáles objetivos han quedado pendientes temporalmente.
- Qué valores han alcanzado las diferentes variables.

Estos y otros detalles mostrados ayudan a comprender todo el proceso llevado a cabo por el motor de inferencia.

El sistema puede usarse de diferentes formas, dependiendo del los objetivos del curso y de la formación previa de las personas que lo recibirán. Está especialmente concebido para la formación de especialistas en "Ciencias de la Computación", pero puede ser usado en otras especialidades. Si se van a explotar todas sus posibilidades deberá emplearse después que los alumnos hayan recibido cursos sobre Compiladores, Estructuras de Datos y Programación Orientada a Objetos. En el caso de que sólo se desee enseñar el mecanismo de inferencia sin analizar su programación no se necesitan requisitos previos.

La Figura 1 presenta una vista general de TeachShell en el momento en que está ejecutando una inferencia. La línea que se observa debajo de la barra del menú se denomina *barra de atributos* y la línea que aparece encima de la línea de estado tiene el nombre de *barra de reglas*. Se observan tres ventanas; la de la izquierda muestra un listado de la base de conocimiento actual, la ventana superior de la parte derecha muestra uno de los métodos fundamentales del sistema (`Find.Exec`) que es el encargado de buscar en el árbol de conocimiento un atributo que se desea probar. Existen tres posibilidades:

1. Que sea un hecho (`IsFact` será verdadero).
2. Que sea un atributo con pregunta asociada (`IsQuestion` será verdadero).
3. Que no sea ninguno de los dos anteriores y en ese caso habrá que inferirlo (llamada al método `Atrib.Infer`).

La ventana inferior de la derecha presenta el método `Attrib.Infer` referido anteriormente. Estos dos algoritmos son los fundamentales para el proceso de inferencia en sí. Comprender cómo funcionan significa entender lo esencial del mecanismo de inferencia (dirigido por objetivos) que sigue TeachShell. La posición, el tamaño, los colores y el hecho de

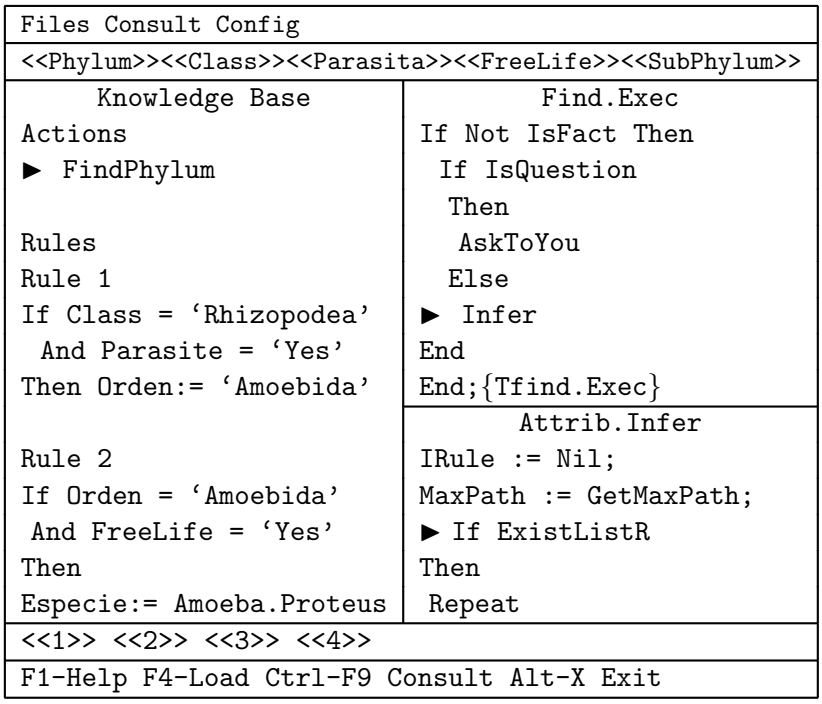

Figura 1. Instantánea de TeachShell en ejecución.

estar presentes o no las ventanas se dejan a elección del usuario, de acuerdo al problema que se esté analizando. No obstante, existen algunas recomendaciones para el docente en relación con este aspecto. La finalidad de la *barra de atributos* es mostrar todos los atributos que forman parte de la base de conocimiento actual. Como es lógico, cuando sean muchos no cabrán en la pantalla y el usuario puede explorarlos situando el cursor del ratón en los paréntesis angulares que los separan y oprimiendo el botón izquierdo para desplazarlos hacia adelante o hacia atrás. El atributo actual, o sea el que se esté probando en ese momento, toma un color rojo (Figura 1) mientras que los atributos pendientes quedan en azul. La *barra de reglas* tiene una finalidad similar, pero con relación a las reglas.

En la Figura 1 también se puede apreciar que cada ventana posee un cursor de línea que muestra el lugar por donde va la ejecución,

tanto dentro de la máquina de inferencia como dentro de la base de conocimiento actual. Se pueden destacar los siguientes aspectos:

- En el momento actual se está probando el atributo `Phylum` (en rojo en la barra de atributos).
- Se está explorando el bloque principal de la base de conocimiento (comienza con la palabra `Actions`). Este bloque se ejecuta cuando el motor de inferencia comienza el proceso de deducción en una base dada (pudieran ser varias). En el instante actual se ha recibido la orden `Find Phylum` (como lo indica el cursor en la ventana *Base de Conocimiento*).
- Por el motivo anterior se está ejecutando el método `Find.Exec`, como se aprecia en la ventana superior de la derecha. Más específicamente, `Find.Exec` ha ordenado que se ejecute el método `Atrib.Infer`, según se aprecia en la ventana inferior derecha. Su próxima acción será buscar una regla que contenga el atributo `Phylum` como conclusión.

Otras posibilidades que están al alcance del usuario de TeachShell le permiten ver en cualquier instante el tipo de valor y el valor en sí que tiene asociado cualquier atributo (puede ser desconocido cuando no se ha probado), las reglas por las que puede ser inferido un atributo dado, etc.

2.2 UCShell y Compiler

UCShell es un ambiente integrado para el desarrollo de sistemas expertos reales, pero su máquina de inferencia admite cualquier base de conocimiento preparada para TeachShell ya que el diseño y programación de ambas se hizo en forma similar. Las diferencias más notables son que UCShell no posee el módulo de enseñanza e incluye un compilador y un editor de bases de conocimiento que no lo tiene TeachShell.

Compiler es un compilador de líneas, básicamente igual al que está contenido en UCShell, pero que se entrega en forma independiente.

Hay varios motivos para esta separación:

1. Cuando se desea desarrollar un sistema experto real no es necesario utilizar las herramientas de enseñanza que posee TeachShell.

2. El estudiante puede comenzar el desarrollo o estudio de un sistema experto con teachShell con el único fin de aprender cómo funcionan sus mecanismos, pero la tarea docente puede convertirse en una aplicación real si logramos motivarlo lo suficiente como para continuar en el proyecto. Si esto sucede necesitará una herramienta para desarrollar su sistema ya que TeachShell, al no estar hecho con ese fin, no cumplirá con sus expectativas. Esa herramienta puede ser el ambiente integrado para sistemas expertos UCShell, que tendrá la ventaja adicional de permitir utilizar todo lo que se haya hecho hasta ese instante.

3. Si se quiere estudiar cómo se construye un árbol de conocimiento a través del proceso de compilación de las bases, sólo es necesario estudiar el compilador, lo que justifica su existencia en forma independiente, además de permitir su uso en la asignatura Compiladores.

4. Si se orienta un ejercicio que implique la modificación de la máquina de inferencia, los cambios se harán en UCShell ya que el hecho de no poseer el módulo de enseñanza simplificará la tarea que por sí sola resulta un poco difícil para los estudiantes.

2.3 ¿Cómo utilizar el sistema?

Seguidamente se brindan algunas sugerencias para organizar las actividades del curso, aunque estamos conscientes de que la creatividad de los profesores debe jugar un papel importante en la organización de éstas. Proponemos realizar, al menos y en este orden, las siguientes actividades:

1. Forma interna de las bases de conocimiento del sistema SESE.

    1.1. Objetivo: Presentar y discutir la jerarquía de clases de la forma interna generada por el compilador del sistema.

1.2. Contenido: Explicar la sintaxis de las bases de conocimiento, la jerarquía de clases del sistema y la forma interna del árbol de conocimiento generado por el compilador.

1.3. Medios: Debe utilizarse algún medio visual para mostrar la jerarquía de clases así como una representación gráfica de la forma interna.

1.4. Método: La actividad debe hacerse en forma de conferencia participativa. Debe provocarse la discusión con relación a la jerarquía de clases y la forma interna que se propone. Cada clase y elemento de la representación gráfica de la forma interna serán presentados de forma escalonada, destacándose el porqué de su necesidad.

2. Método de inferencia dirigido por objetivos.

2.1. Objetivo: Comprender y generalizar el método de inferencia y de solución de conflictos de la máquina TeachShell.

2.2. Contenido: Estudio visual del mecanismo de inferencia de TeachShell.

2.3. Medios: Computadoras personales.

2.4. Método: La clase se desarrollará en el laboratorio. El profesor pondrá en disco varias bases de conocimiento. Deberá escoger temas sencillos y dominados por todos: la clasificación de los animales y las plantas son buenos ejemplos. Deben ubicarse en el mismo directorio los archivos fuente y los archivos compilados; se puede usar el compilador de líneas Compiler que es muy sencillo. Se configurará TeachShell para mostrar solamente el listado de la base de conocimiento. Se les dirá a los estudiantes que carguen la base y que deduzcan cómo es que se realiza el proceso de inferencia. Posteriormente se hará cambiar el orden de las reglas y se les pedirá que hagan todo el proceso de compilación, carga y corrida paso a paso. Después de esta actividad se preguntará cómo influyen los cambios realizados en el proceso de inferencia y se explicará cuál es el método de solución de

problema que se utiliza cuando existe más de una regla con la misma conclusión.

3. Algoritmos de inferencia.

   3.1. Objetivo: Comprender los algoritmos de inferencia fundamentales de la MI TeachShell.

   3.2. Contenido: Estudio visual de los algoritmos de inferencia de TeachShell.

   3.3. Medios: Computadoras personales.

   3.4. Método: Ls tercera actividad también se realizará en el laboratorio y en este caso se hará un proceso similar al anterior, pero ahora se configurará TeachShell para que muestre a la vez el recorrido por la base de conocimiento y por sus algoritmos internos. Al finalizar la clase se debe orientar la realización de ejercicios independientes tales como: transformar los algoritmos para poder utilizar otro método de solución de problemas, agregar nuevas sentencias a las permitidas y toda una gama de ejercicios complejos que deben ser resueltos en actividades extra curriculares y revisados por el profesor.

Es importante destacar que en todas estas actividades la labor del estudiante resulta fundamental. Las herramientas desarrolladas están hechas con el claro fin de que el conocimiento fluya hacia ellos de una manera natural, en la mayoría de los casos a partir de sus propias conclusiones. Por supuesto que el papel del profesor resulta importante como director de todo el proceso y como catalizador de todas las dudas y contradicciones que puedan surgir.

3 Sistema para la Enseñanza del Prolog (SEP)

El sistema SEP está compuesto por los siguientes módulos:

1. Teaching: un sistema para la enseñanza del paradigma de la Programación Lógica.

2. Una metodología para programar sistemas expertos que utilizan el Prolog como máquina de inferencia.

3. Progen: un sistema que aplica la metodología anterior para la generación automática de sistemas expertos.

3.1 Teaching

Para el diseño y la programación del sistema Teaching se utilizó el paradigma de la Programación Lógica ya que uno de los objetivos finales del curso es el estudio del diseño y programación del sistema en sí y también la inclusión de nuevos predicados o la modificación de los predicados que están contenidos en él.

El sistema incluye un tutorial estructurado como hipertexto que explica teóricamente los conceptos fundamentales del lenguaje, a fin de que el estudiante pueda analizarlos tomando en cuenta su definición formal o más precisa.

Debido a que se considera que la barrera fundamental para aprender algo no está generalmente determinada por la mayor o menor dificultad para manejar información, sino por la carencia de motivación del estudiante para hacer algo con ella, y a que varios estudios señalan que la información que se recibe pero no se usa durante el proceso de aprendizaje difícilmente se recuerda cuando se necesita [4], el componente principal de Teaching es su herramienta de simulación, la cual dispone de un conjunto de programas ejemplos que facilitan el aprendizaje al mostrar los caminos de ejecución tomados y poder apreciar cómo influyen éstos sobre cada una de las estructuras de datos utilizadas.

3.1.1 Aspectos pedagógicos

Se dice, no sin razón, que el uso de un código existente permite a los estudiantes terminar un proyecto en una cantidad de tiempo razonable [5]. En realidad se logra más que eso cuando se emplea un código bien hecho, que permite la asimilación de un estilo de programación adecuado. Esta es la idea medular del sistema Teaching, enseñar a programar observando el comportamiento de ejemplos significativos que

ilustran los mecanismos fundamentales del Prolog, mostrando cómo dichos mecanismos resuelven el problema planteado.

Se ha procurado crear un banco de problemas que sean conocidos por los estudiantes. Algunos de ellos son reales y otros clásicos pero requieren de cierta dosis de ingenio para resolverlos. Entre los primeros se puede mencionar el programa para obtener las derivadas de una función. Al estudiante se le pide que presente las ideas generales para hacer un programa que permita encontrar la derivada de una función general. En este punto conviene que se le permita justificar la elección de un lenguaje X para enfrentar la tarea y después mostrarle las ventajas que brinda el Prolog cuando se hace procesamiento simbólico. Este tipo de ejemplo sirve, además, para establecer un vínculo entre la Programación Lógica y otras asignaturas, como el Análisis Matemático, en este caso.

En el segundo grupo de problemas resaltan, entre otros, el de ubicar ocho reinas en un tablero de ajedrez sin que se ataquen mutuamente, el de las torres de Hanoi, etc. Dichos problemas contribuyen notablemente al desarrollo del pensamiento lógico, además de que en este caso se logra vincular la Programación Lógica con aspectos que se abordan en otros cursos, particularmente el de Matemática Discreta.

Un tercer grupo de problemas, que es realmente la combinación de los dos anteriores, está formado por ejercicios propios de alguna asignatura de la especialidad, por ejemplo el problema de programar un autómata de pila no determinístico, que responde a la asignatura Compiladores. El autómata es una necesidad real en un compilador, pero la simplificación que se hace del mismo, en función de la enseñanza, sólo debe abarcar la idea de reconocer una cadena de entrada simple, para no abrumar a los estudiantes con numerosos conceptos que compliquen el tema central de su aprendizaje.

Según Reeves [3], en un aula el conocimiento se presenta muchas veces como algo acabado, sin embargo sería más adecuado considerarlo como algo útil para resolver problemas o comprender eventos. De acuerdo a la concepción del sistema Teaching algunos programas son generados a partir de las ideas elaboradas por los estudiantes, otros son parcialmente generados y el sistema en general se puede modificar

sobre la base de lo que él nos ha enseñado.

3.1.2 Forma de utilización

Como se mencionó anteriormente, el enfoque que instrumenta el sistema como recurso fundamental consiste en visualizar la trayectoria de ejecución de los programas y los efectos que se producen sobre sus datos.

Tomemos por ejemplo el caso de un autómata de pila no determinístico. Cuando el estudiante seleccione este problema se le presenta una definición formal:

> *Un autómata de pila es un 7-tuplo* $P = (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$,
> ...

El enunciado permite recordar algunos conceptos estudiados en otra asignatura de la especialidad. Seguidamente el sistema pedirá al alumno que defina:

- El conjunto de estados (Q).
- El conjunto de símbolos del alfabeto (Σ).
- El conjunto de transiciones entre los estados (δ).
- El conjunto de símbolos permitidos en la pila (Γ).
- El estado inicial (q_0).
- El símbolo inicial en la pila (z_0).
- El conjunto de estados finales (F).

De esta forma sencilla, el estudiante participa en la construcción del conocimiento ya que a partir de estos datos se generan algunas de las cláusulas Prolog que formarán el programa. Otras cláusulas conforman la parte invariable del problema a resolver y son generadas sin intervención del estudiante. Por último se solicita la cadena de entrada que deberá ser analizada por el autómata para decir si la acepta o no.

Seguidamente se le presentará al estudiante un programa formado por las cláusulas derivadas del diálogo, más una parte invariable. Cuando el sistema se aplica en clases, el profesor debe destacar por qué existe esa parte invariable y en el caso que se esté usando en forma individual, se debe haber orientado algún ejercicio para que se analice el programa presentado y se justifiquen cada una de sus cláusulas.

En el caso del autómata, ls pila y la cadena de entrada se simulan mediante listas Prolog. Estas y otras estructuras se presentan en ventanas aparte de manera que que se pueda apreciar en cada instante los valores con que estén instanciadas y y cómo reciben su valor.

Seguidamente Teaching permite ejecutar paso a paso el programa, pudiendo apreciar los mecanismos típicos del lenguaje, tales como la unificación, el backtracking, la recursividad y otros.

La gama de ejemplos abarca todos los mecanismos fundamentales de la programación lógica y en virtud de la concepción del sistema se puede plantear como ejercicio de culminación del curso el diseño y programación de pequeños programas que los estudiantes consideren apropiados para ayudar a otros a aprender lo que ellos ya han aprendido. Estos nuevos ejemplos son incorporados al sistema por los propios estudiantes. La experiencia muestra que este tipo de situación en la que al alumno se le da la oportunidad de convertirse en profesor logra motivar considerablemente a los estudiantes.

Prolog es un lenguaje declarativo en el que se debe seguir el principio de centrarse en el "qué hacer", declararlo y dejar que el lenguaje resuelva el problema sin procuparse por cómo lo resolverá. Se pudiera pensar que una herramienta que presenta el funcionamiento del lenguaje no resulta adecuada, pero la experiencia de aplicación de Teaching ha mostrado que los programadores bisoños no logran aprovechar las posibilidades del lenguaje hasta que no han comprendido la potencia de sus mecanismos. El ejemplo antes mencionado resulta típico ya que la solución se basa, ante todo, en los conceptos de unificación de patrones y *backtracking* y cuando éstos no se conocen bien, la tendencia es a resolver el problema usando estrategias imperativas ya conocidas y despreciando los principales recursos lógicos.

Lo primero que debe enseñarse en un curso de programación lógi-

ca debe ser el paradigma de programación, sin entrar en los detalles propios de una implementación particular. Tomando en cuenta esta afirmación se propone usar Teaching de la siguiente forma:

- Utilizar su hipertexto para apoyar el estudio de los contenidos impartidos en clase. Esa parte del sistema sólo hace referencia a conceptos generales y puede usarse desde el primer día de clases en forma independiente en el laboratorio.

- Usar los programas que simulan el comportamiento del lenguaje para introducir nuevos conceptos. Las primeras actividades de este tipo deben ser dirigidas por el profesor pero el estudiante debe jugar un papel activo. Con ese fin se puede hacer la corrida paso a paso del programa y explicar en cada caso sólo los detalles necesarios para que los estudiantes puedan deducir cómo funciona el mecanismo que se enseña.

- Se deben orientar actividades independientes con un fin bien dirigido, por ejemplo "analizar cómo es que el programa deriva una función dada". Los resultados de esta actividad se deben discutir en un seminario y extraer conclusiones a partir de los criterios cosechados.

- Cuando el curso haya avanzado lo suficiente y el estudiante haya adquirido los conocimientos necesarios se le debe explicar cómo es que se hizo el sistema que le ha servido de apoyo al estudio.

- En esta etapa final se entiende que el dominio alcanzado por los estudiantes con relación a los conceptos estudiados y las propias dificultades a las que se han enfrentado los faculta para proponer modificaciones a los programas presentados o nuevos programas que permitan ayudar a comprender mejor los conceptos estudiados. Se les pide propuestas que deben ser discutidas en el aula y llegado al consenso necesario se propone, como un proyecto extra clase, la implementación de las ideas presentadas.

En esta parte del trabajo hemos presentado nuestra experiencia en la utilización de Teaching, estando convencidos de que el uso de esta o

cualquier otra herramienta educativa por la comunidad de educadores logrará enriquecerlo sustancialmente.

3.2 Una metodología para programar sistemas expertos que utilizan el Prolog como máquina de inferencia

La enseñanza de la programación debería comenzar por la forma declarativa y no por la forma imperativa. Se conoce de experiencias de la enseñanza de la Programación Lógica a niños pequeños con resultados sorprendentes. Esto se debe a que la forma de pensar del ser humano se acerca más a la lógica y cuando enseñamos a programar en forma imperativa realmente estamos tratando de imponer un razonamiento que no es propio de los humanos y que está determinado, en última instancia, por la arquitectura secuencial de las computadoras. No obstante, en contados lugares se comienza a enseñar de esta manera.

La metodología presentada requiere de un cierto conocimiento de Programación Lógica y de sistemas expertos. Su aporte a la enseñanza se basa en la disciplina de programación que impone, lo que ayuda, en gran medida, al desarrollo de hábitos correctos de programación.

Debido a las limitaciones naturales de espacio, no se exponen en este trabajo todos los detalles de esta metodología y sólo se mencionan algunas ideas generales.

Una base de conocimiento se puede considerar como un conjunto de reglas que definen relaciones entre objetos y por esa razón resulta natural pensar en Prolog como una máquina de inferencia apropiada para el desarrollo de sistemas expertos. No obstante escribir sistemas de este tipo para una máquina de inferencia cualquiera resulta una tarea relativamente fácil, mientras que escribir las mismas reglas para Prolog no resulta, en algunas ocasiones, todo lo fácil que desearían los Ingenieros del Conocimiento, quienes no están interesados en la programación en sí sino en el hecho de "descubrir" dentro del universo intelectual de los expertos humanos todas las reglas no escritas que han logrado establecer a través de muchos años de trabajo, de experiencias vividas, de fracasos, etc., para poder obtener una aplicación en un tiempo razonable y con el grado de experticidad necesario.

No obstante lo dicho en el párrafo anterior son demasiadas las ventajas que puede darnos un lenguaje de alto nivel con una máquina de inferencia interna poderosa, para no utilizarla en el desarrollo de sistemas expertos.

Al observar la correspondencia casi natural que existe entre un sistema de reglas de producción y las cláusulas de Prolog queda claro que la programación de un sistema experto en Prolog se podrá lograr más fácilmente si tratamos a este último como una máquina de inferencia que utiliza esta forma de representación del conocimiento. Estos y otros detalles son los que se deben destacar en el aula cuando se propone Prolog como motor de inferencia.

Otro aspecto a destacar es el hecho de que cuando se utiliza una máquina de inferencia convencional el Ingeniero de Conocimiento no tiene que preocuparse por controlar, en cada regla, si una pregunta que se debe hacer ha sido o no realizada con anterioridad, ya que el mecanismo propio del sistema se encarga de hacerlo, mientras que en Prolog no ocurre así y esa responsabilidad queda en manos del programador.

El profesor que utilice la metodología [2], no debe presentarla como algo aislado, sino que deberá ante todo hacer observaciones como las anteriores y, basado en el conocimiento previo que tienen los estudiantes, provocar que se vaya llegando en forma natural a las mismas ideas que se desarrollan en ella. Se puede lograr este propósito debido a que las ideas descriptas por la metodología son claras y lógicas. Nuestra experiencia nos demuestra la similitud sorprendente entre las conclusiones de los estudiantes y la metodología descripta a partir de preguntas apropiadas y situaciones dadas.

3.3 Progen: un sistema que aplica la metodología anterior para la generación automática de sistemas expertos

Una forma efectiva de culminar el curso es la presentación de esta herramienta que automatiza casi todo el proceso de escribir sistemas expertos en Prolog. Es una buena oportunidad para mostrar a los estudiantes cómo muchas de las ideas sugeridas por ellos no sólo están

plasmadas en una metodología sino que también forman parte de una herramienta profesional que podrán utilizar en lo sucesivo.

Progen en sí no es una herramienta pedagógica (por ese motivo no se explica en detalle) pero la metodología que sigue sí constituye una ayuda al desarrollo de correctos hábitos de programación y el dominio de la metodología y del propio lenguaje ayudan a la mejor explotación del sistema.

Desarrollos de este tipo permiten hacer una educación más real, que acerque más al estudiante universitario al ejercicio de su futura profesión.

4 Conclusiones

El uso de los sistemas SESE y SEP ha permitido que los estudiantes alcancen un mayor dominio de los conceptos estudiados en la disciplina Inteligencia Artificial.

El estudiante participa activamente en la construcción de su conocimiento, pudiendo incluso modificar y enriquecer los medios de enseñanza utilizados para ayudar a otros estudiantes que utilizarán los sistemas en cursos posteriores.

El hecho de disponer de herramientas reales logra interesar a los estudiantes y hace que la enseñanza sea más productiva a la vez que la acerca más a la realidad.

La experiencia del uso de estos sistemas refleja que este es un camino correcto para lograr una educación superior realmente efectiva.

Referencias

[1] Hearst, M. A., *Preface: Improving Instruction of Introductory AI*, AAAI Fall Symposium on Improving Instruction of Introductory AI (IIIA), 1994.

[2] Lezcano, M. *Prolog y Sistemas Expertos*, Ediciones Universidad Central de Las Villas, Santa Clara (1995).

[3] Reeves, T. et al. *Designing CAL to Support Learning: The Case of Multimedia in Higher Education*, Nordic Conference on Computer Aided Higher Education, Helsinki University of Technology, Helsinki (1991).

[4] Schank, R., Kass, A. *A Goal-Based Scenario for High School Students*, Communications of the ACM, **39**(4), 1996.

[5] Walker, R. *The Use of Computers for Teaching Artificial Intelligence*, AAAI Fall Symposium on Improving Instruction of Introductory AI (IIIA), 1994.

[6] Zilberstein, S. *Teaching Graduate Level Artificial Intelligence*, AAAI Fall Symposium on Improving Instruction of Introductory AI (IIIA), 1994.