Los Juegos como Herramienta Docente. Formalización de Juegos Lógicos en Prolog

Faraón Llorens, Mª Jesús Castel, Francisco Mora, Carlos Villagrá
Dept. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail:[faraon,chus,mora,villagra]@dccia.ua.es

Resumen

Ludwing Wittgenstein, filósofo austriaco, escribió que "podría escribirse una obra filosófica buena y seria compuesta enteramente por chistes". Si se entiende el chiste, se entenderá el argumento implícito en él. Podríamos trasladar este pensamiento a nuestro campo y decir que se podría estudiar lógica por medio de chistes, juegos y acertijos; la enseñanza de la Lógica no tiene porque ser aburrida si se saben encontrar caminos que la hagan agradable y amena. Como dice Martin Gardner [4] la virtud, y nosotros pensamos que la dificultad, está en encontrar el equilibrio entre el juego y la seriedad: el juego mantendrá interesados a nuestros alumnos y alumnas, y motivará su discusión más allá de las paredes del aula; la seriedad convertirá nuestras clases en algo útil y provechoso (y, para algunos, justificará nuestro sueldo).

1. Lógica y programación lógica

La Lógica nos permite formalizar conocimientos, y esta formalización es un paso previo e indispensable para poder automatizar formas de razonamiento. Cuando nos planteamos resolver un determinado problema, bien de forma manual o por ordenador, debemos empezar planteándonos una serie de interrogantes acerca del conocimiento implicado en dicho problema. Y en la representación y la manipulación (sistema de razonamiento) de ese conocimiento es donde juega un papel importante la Lógica.

La Lógica de Primer Orden pone a nuestra disposición un lenguaje que nos permitirá formalizar expresiones del conocimiento humano haciendo explícitos los objetos y las relaciones, así como sus restricciones. Y dicho lenguaje se convierte en uno de los mecanismos de representación del conocimiento, con la ventaja de que además proporciona un método, la deducción matemática, para obtener nuevo conocimiento a partir del antiguo.

La Lógica ha servido también como base para el desarrollo de un nuevo paradigma de programación : *Programación Lógica*. La idea central de la programación lógica la podemos expresar con la conocida ecuación de Kowalski [5]:

Algoritmo = Lógica + Control

de manera que el control (estrategia para encontrar la solución) la dejamos en manos de la máquina, y el programador sólo se preocupa de la lógica (información acerca del problema que queremos resolver).

El lenguaje Prolog (PROgramación LOGica), definido por Colmerauer [3], es el mejor representante de esta nueva concepción de la programación. Un sistema Prolog esta basado en un comprobador de teoremas por Resolución usando cláusulas de Horn y la estrategia de reevaluación. La Base de Conocimiento de Prolog estará constituida por hechos y reglas (cláusulas de Horn con cabeza) y su ejecución consistirá en la introducción de un objetivo o pregunta (cláusula decapitada o negada) que queremos hacer cumplir.

Así, programar en Prolog consiste en dar al ordenador un universo finito en forma de hechos y reglas, proporcionando los medios para realizar inferencias de un hecho a otro. A continuación, si se hacen las preguntas adecuadas, Prolog buscará las respuestas en dicho universo y las presentará en pantalla. Esquemáticamente, la programación en Prolog consiste en :

- declarar algunos *hechos* sobre los objetos y sus relaciones
- definir algunas *reglas* sobre dichos objetos y relaciones, y
- hacer *preguntas* sobre esos objetos y relaciones.

Para el desarrollo de la práctica hemos utilizado el compilador de programación lógica SWI-Prolog versión 4.0.0. para Windows. SWI-Prolog [1] es un compilador Prolog de dominio público para ordenadores PC desarrollado en el Dept. of Social Science Informatics (SWI) de la Universidad de Ámsterdam dirigido, principalmente, a la investigación y la educación.

2. Juegos de lógica

Vamos a resolver en Prolog juegos de lógica que se tienen que determinar por deducción a partir de un conjunto de pistas. El objetivo es correlacionar una serie de propiedades que cumplen distintos elementos de nuestro Dominio (Universo del Discurso). La restricción a la que está sujeto este juego es que dos elementos distintos del mismo dominio no pueden estar asociados a un mismo elemento de otro dominio. Gran cantidad de juegos de este tipo aparecen periódicamente en distintos tipos de revistas de pasatiempos [2]. Veamos un ejemplo:

Un alumno de Informática, debido al nerviosismo del primer día de clase, sólo recuerda el nombre de sus profesores (María, Jesús y Faraón), las asignaturas que se imparten (Lógica, Programación y Matemáticas) y el día de la semana de las distintas clases (lunes, miércoles y jueves). Además recuerda que :

1.- La clase de Programación, impartida por María, es posterior a la de Lógica.

2.- A Faraón no le gusta trabajar los lunes, día en el que no se imparte Lógica.

¿Serías capaz de ayudarle a relacionar cada profesor con su asignatura, así como el día de la semana que se imparte?

NOTA : Sabemos que cada profesor imparte una única asignatura, y que las clases se dan en días diferentes.

Podemos extraer la siguiente información:

- trabajaremos con tres dominios (profesor, asignatura y día), que determinarán tres propiedades (predicados con un argumento).
- cada uno de estos tres dominios tendrá 3 objetos:
  - D1 = {profesor} = {*María, Jesús, Faraón*}
  - D2 = {asignatura} = {*Lógica, Programación, Matemáticas*}
  - D3 = {día} = {*lunes, miércoles, jueves*}
- estas tres propiedades al relacionarse dos a dos, determinarán 3 relaciones binarias (profesor-asignatura, profesor-día, y asignatura-día).

Para entender mejor su funcionamiento vamos a resolverlo primero de forma manual (figura 1), y para ello nos podemos ayudar de un cuadro de múltiples entradas. Marcaremos con un círculo cuando sepamos que la relación es cierta, y descartaremos, marcando con una cruz, aquellas relaciones que sean falsas. Al marcar una casilla con un círculo deberemos eliminar todas las celdas de su horizontal y vertical llenándolas con cruces. Una vez anotadas todas las pistas y, de ser necesario, completar el cuadro y correlacionar algunas relaciones binarias, obtendremos la solución.

3. Resolución del juego en Prolog

Prolog se basa en el Cálculo de Predicados, y por ello vamos a formalizar el problema utilizando las herramientas que nos proporciona el Lenguaje de la Lógica de Primer Orden. Para diseñar la solución realizaremos los siguientes pasos:

3.1. Definición de Propiedades

Determinaremos el número de propiedades que intervienen en el problema (n) y le asignaremos a cada propiedad un predicado (palabras o fragmentos de palabras que nos recuerden su función). Además, para cada propiedad, determinaremos el número de elementos que pertenecen al dominio (m). Por definición del problema, todos los dominios tienen el mismo número de objetos. Y por último, para cada una de las propiedades escribiremos tantos *hechos Prolog* como elementos tenga el dominio.

Anotación de pistas :

Pista 1.-

	Lógica	Program.	Matemát.	Lunes	Miércoles	Jueves
María	X	O	X			
Jesús		X				
Faraón		X				
Lunes		X				
Miércoles						
Jueves	X					

Pista 2.-

	Lógica	Program.	Matemát.	Lunes	Miércoles	Jueves
María	X	O	X			
Jesús		X				
Faraón		X		X		
Lunes	X	X				
Miércoles						
Jueves	X					

Completar el cuadro :

	Lógica	Program.	Matemát.	Lunes	Miércoles	Jueves
María	X	O	X			
Jesús		X				
Faraón		X		X		
Lunes	X	X	O			
Miércoles	O	X	X			
Jueves	X	O	X			

Correlacionar las relaciones binarias :

	Lógica	Program.	Matemát.	Lunes	Miércoles	Jueves
María	X	O	X	X	X	O
Jesús	X	X	O	O	X	X
Faraón	O	X	X	X	O	X
Lunes	X	X	O			
Miércoles	O	X	X			
Jueves	X	O	X			

La solución es :

Profesor	Asignatura	Día
María	Programación	Jueves
Jesús	Matemáticas	Lunes
Faraón	Lógica	Miércoles

Figura 1. Resolución gráfica del juego

```
numeroPropiedades(3).
objetosUniverso(3).

/*-   PROPIEDADES  -*/

profesor(maria).
profesor(jesus).
profesor(faraon).

asignatura(logica).
asignatura(programacion).
asignatura(matematicas).

dia(lunes).
dia(miercoles).
dia(jueves).
```

3.2. Definición de Relaciones

Relacionaremos todas las propiedades entre sí, con lo que tendremos tantas relaciones como combinaciones podamos hacer de las *n* propiedades tomadas de dos en dos:

$$C_n^2 = \binom{n}{2} = \frac{n!}{2!(n-2)!}$$

Así, como en nuestro ejemplo teníamos 3 propiedades, tendremos que definirnos 3 relaciones. En principio, cualquier profesor puede impartir cualquier asignatura, cualquier profesor puede dar clase cualquier día y cualquier asignatura puede ser impartida cualquier día. Para formalizar estos predicados crearemos 3 reglas Prolog, que nos relacionarán las distintas propiedades:

```
/*-   RELACIONES   -*/

prof_asig(Prof,Asig) :-
            profesor(Prof),
            asignatura(Asig).

prof_dia(Prof,Dia) :-
            profesor(Prof),
            dia(Dia).

asig_dia(Asig,Dia) :-
            asignatura(Asig),
            dia(Dia).
```

3.3. Interpretación de las Pistas

Interpretaremos cada una de las pistas para poder deducir la solución del problema. Tendremos dos tipos de pistas: las que afirman alguna relación y las que la niegan.

- *Pista afirmativa*: nos aporta información positiva, es decir, que cierto objeto del Universo está relacionado con otro determinado objeto de otro Universo por medio de una determinada relación. Lo representaremos por un hecho Prolog en la relación entre ambos Universos:
  ```
  prof_asig(maria,programacion).
  ```

- *Pista negativa*: nos aporta información que no debe cumplirse, es decir, que cierto objeto del Universo no está relacionado con otro determinado elemento de otro Universo. Lo representaremos por una regla Prolog para la relación entre ambos Universos, de manera que elimine la posibilidad de que dichos objetos se relacionen:
  ```
  prof_dia(faraon,Dia) :-
            dia(Dia),Dia\=lunes.
  ```

Finalmente, deberemos asegurarnos que tanto para las pistas afirmativas como para las negativas se elige la cláusula correcta, y que en caso de reevaluación no se unifique con la cláusula general. Para ello modificaremos las reglas generales introducidas en el punto 2, quedando de la siguiente manera:

```
prof_asig(Prof,Asig):-
            profesor(Prof),
            Prof\=maria,
            asignatura(Asig).
```

3.4. Correlación de Propiedades

Hay que relacionar cada elemento de un dominio con uno y sólo uno de los elementos de cada uno de los otros dominios. Por tanto buscaremos una nueva solución individual y comprobaremos que dichos objetos no pertenezcan a la solución parcial que tenemos en ese momento. Para ello definiremos una función recursiva que vaya obteniendo todas las soluciones. Habremos encontrado una solución total al problema cuando el número de elementos que tenga nuestra

solución sea el mismo que objetos tengan los dominios. Finalmente, por reevaluación, Prolog encontrará automáticamente todas las soluciones posibles al problema.

```
/*---- PROGRAMA PRINCIPAL  ----*/
iniciar :- ini([]).

ini(L):- objetosUniverso(M),
        nel(L,M),escribir(L),
        nl, pausa, fail.

ini(L) :-
      prof_asig(Prof,Asig),
      nopertenece(Prof,L,1),
      nopertenece(Asig,L,2),
      prof_dia(Prof,Dia),
      nopertenece(Dia,L,3),
      asig_dia(Asig,Dia),
      ini([Prof,Asig,Dia]|L]).
```

3.5. Rutinas de carácter general

Definimos, también, una serie de rutinas de carácter general y de manejo de listas que utilizamos a lo largo del programa.

4. Generalización de la solución

Podemos observar que los pasos 4 (correlacionar propiedades) y 5 (rutinas generales) son iguales, con pequeñas diferencias, para todos los problemas que queramos resolver de este tipo de juego. Mientras que los pasos 1 (propiedades), 2 (relaciones) y 3 (pistas) dependen del enunciado concreto del problema a resolver. Por tanto vamos a separar el código Prolog en dos partes:

- *Juego*: esquema general del juego, único e igual para todos.
- *Base de Conocimientos*: dependiente del enunciado del problema.

Continuando con el ejemplo anterior, vamos a desarrollar ambas partes.

4.1. Juego

Necesitará unas pequeñas modificaciones ya que queremos que sirva para todo tipo de enunciado. En primer lugar, desde este programa leeremos la Base de Conocimientos que queramos utilizar y que ya estará elaborada a partir de un problema concreto que debamos resolver. Una vez consultada, deberemos tener en cuenta el número de propiedades (*n*) y el número de objetos de cada Universo (*m*). Como el número de relaciones, y por consiguiente la cláusula que las correlaciona, depende del número de propiedades, la cláusula `iniciar` llamará a la correspondiente `ini` según el número de propiedades del problema concreto en que estemos.

```
/*----   JUEGO   ----*/
iniciar :-
      write('B.C.: '),
      read(BC),
      reconsult(BC),!,
      nl,write('B.C. '),
      write(BC),
      write(' consultada'),nl,nl,
      numeroPropiedades(N),
      ini(N,[]).

iniciar :-
      nl,write('ERROR:  B.C.  no
encontrada'),nl.
```

4.2. Base de Conocimientos

Al generalizar y tener que interpretar todas las Bases de Conocimientos desde un mismo programa, debemos en primer lugar determinar dos valores concretos implícitos en el enunciado: el número de propiedades tratadas en el problema (*n*) y el número de individuos de cada universo (*m*). Para ello incluiremos al principio de la Base de Conocimientos dos hechos que informen al juego de dichos parámetros. Por otro lado, como el procedimiento que correlaciona las propiedades está incluido en el juego, deberemos nombrar a las relaciones de manera general (`r12`, `r13`, ...), para que nos puedan servir para cualquier enunciado. Por lo demás, el resto queda igual que lo comentado en los puntos 1, 2 y 3 del apartado anterior.

```
/*-- BASE DE CONOCIMIENTOS --*/

numeroPropiedades(3).
objetosUniverso(3).
```

```
/*- PROPIEDADES -*/
profesor(maria).
profesor(jesus).
profesor(faraon).

asignatura(logica).
asignatura(programacion).
asignatura(matematicas).

dia(lunes).
dia(miercoles).
dia(jueves).

/*- RELACIONES -*/
r12(maria,programacion).
r12(Prof,Asig) :-
      profesor(Prof),
      Prof\=maria,
      asignatura(Asig).

r13(faraon,Dia) :-
      dia(Dia), Dia\=lunes.
r13(Prof,Dia) :-
      profesor(Prof),
      Prof\=faraon,
      dia(Dia).

r23(logica,Dia) :-
      dia(Dia),
      Dia\=lunes,Dia\=jueves.
r23(programacion,Dia) :-
      dia(Dia), Dia\=lunes.
r23(Asig,Dia) :-
      asig(Asig),
      Asig\=logica,
      Asig\=programacion,
      dia(Dia).
```

A partir de este momento, ante un nuevo problema de este tipo, solamente deberemos preocuparnos de crear la Base de Conocimientos oportuna (la cuál deberá contener las propiedades, relaciones y pistas definidas adecuadamente) y resolverlo utilizando el juego que ya tenemos.

5. Conclusión

La enseñanza de la Lógica no tiene por qué ser aburrida. Todos recordamos las novelas de Sherlock Holmes o Hercules Poirot, y cómo sus mentes lógicas y deductivas eran capaces de realizar hábiles razonamientos para descubrir a los culpables a partir de pequeños indicios. Por otra parte, el rigor y la exactitud deben ser cualidades indispensables en cualquier trabajo serio. La lógica nos proporciona herramientas para poder trabajar formalmente con "conocimiento". En este artículo hemos pretendido presentar como los juegos pueden convertirse en una herramienta docente. Además, el lenguaje de programación lógica Prolog puede servirnos para solucionar problemas cuya resolución implique un proceso de deducción. Prolog es "pura lógica" y por tanto se puede trabajar con él de forma intuitiva.

La experiencia aquí comentada ha sido aplicada como prácticas de la asignatura cuatrimestral "Lógica de Primer Orden" de los estudios de Ingeniería Informática en la Universidad de Alicante [6]. Pensamos que podría ser interesante aplicarlo a otros estudios y/o niveles que necesiten de una formación científica e investigadora.

Los objetivos perseguidos al realizar esta práctica han sido :

- Representación del conocimiento mediante el lenguaje de la Lógica de Primer Orden.
- Interpretación de pistas : conceptos y esquemas lógicos del pensamiento.
- Utilización del lenguaje de programación lógica Prolog.

Referencias

[1] http://www.swi.psy.uva.nl/projects/SWI-Prolog

[2] *LOGIC. Juegos Lógicos.* Publicación editada por Zugarto Ediciones, S.A., Madrid.

[3] Colmerauer, A. y otros. *Un Systeme de Communication Homme-Machine en Francais*, Groupe de Recherche en Intelligence Artificielle, Université d'Aix-Marseille, 1973

[4] Gardner, M. *Carnaval Matemático.* Alianza Editorial (libro de bolsillo 778), 1987.

[5] Kowalski, R. *Lógica, programación e inteligencia artificial*. Ediciones Díaz de Santos, 1986.

[6] Llorens, F. y otros. *Formalización del Razonamiento.* JENUI'98, Sant Julià de Lòria (Principat d'Andorra), julio 1998.