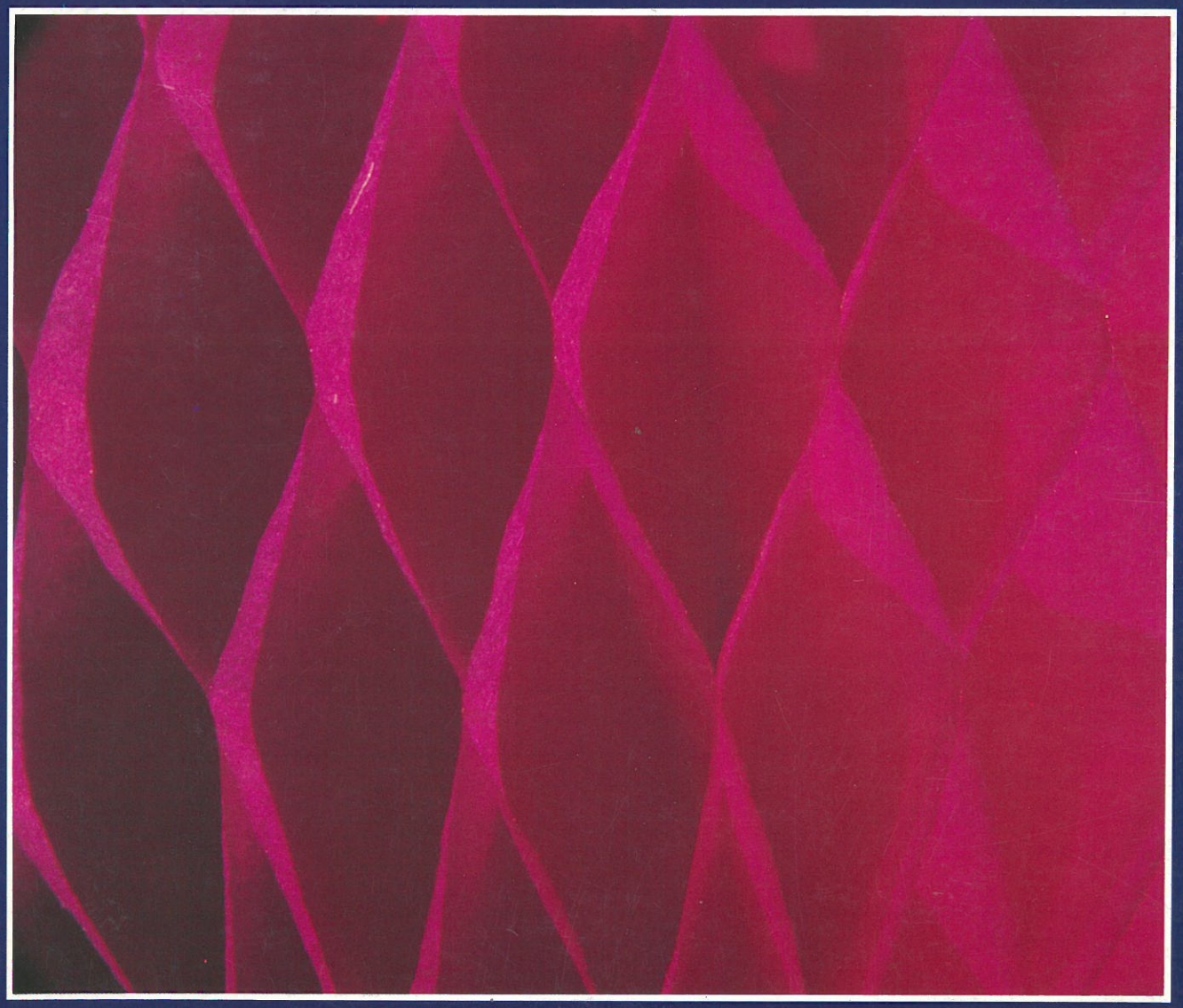
INFORMATICA Y AUTOMATICA

MEMORIA ASOCIATIVA PARA RECONOCIMIENTO DE PATRONES

ENTORNO DE SISTEMAS DE CONTROL EN CIM

CALIDAD DE SOFTWARE E ISO 9000

NOTICIAS AEIA: INFORMES CURSOS DE VERANO



Vol. 26 Núm. 3 Septiembre 1993

INFORMATICA Y AUTOMATICA

AEIA - Consejo Directivo

Junta Ejecutiva

Presidente
Antonio R. Vaquero Sánchez
Dep. Informática
y Automática
Facultad de Físicas
Universidad Complutense
28040 Madrid

Vicepresidente de relaciones
Salvador Ros Torrecillas
Instituto de Automática Industrial
La Poveda, Arganda del Rey
28500 (Madrid)

Vicepresidente de actividades
Ramón Yepes Segovia
Investrónica
c/ Tomás Bretón, 62
28045 Madrid

Tesorero
Juan Manual Sánchez Pérez
Dep. Informática
y Automática
Facultad de Físicas
Universidad Complutense
28040 Madrid

Secretario
Rafael Vargas Traid
Tecnatom
Ctra. N-1, km. 19
28709 San Sebastián de los Reyes (Madrid)

Vocales

Rosa Alonso Muñoz
ALCATEL SESA
c/ Ramírez de Prado, 5
28045 Madrid

José A. Cordero Martín
Inst. Automática Industrial
La Poveda
28500 Arganda del Rey
(Madrid)

Sebastián Dormido Bencomo
Dep. Automática e Informática
Facultad de Ciencias
U.N.E.D.
28040 Madrid

José C. González Cristóbal
Dep. Ingeniería de Sistemas Telemáticos
E.T.S.I. Telecomunicación
Universidad Politécnica de Madrid
28040 Madrid

Francisco J. López Aligué
Dep. de Electrónica
Facultad Ciencias
Universidad de Extremadura
Avda. de Elvas s/n
06071 Badajoz

Salvador Navas Gutiérrez
CESELSA
La Granja, 84
28100 Alcobendas
(Madrid)

Luis A. Petit Herrera
CITEMA
Plaza Alonso Martínez, 3
28004 Madrid

Alberto Prieto Espinosa
Dep. Electrónica y Tecn. de Computadores
Facultad de Ciencias
Universidad de Granada
18071 Granada

Eugenio Andrés Puente
Dep. de Automática
E.T.S.I. Industriales
Universidad Politécnica
José Gutiérrez Abascal, 2
28006 Madrid

José Rodellar Benedé
E.T.S.I. Caminos, Canales y Puertos
Universidad Politécnica de Cataluña
Jorge Girona Salgado, 31
08034 Barcelona

Fernando Sáez Vacas
Dep. Ingeniería de Sistemas Telemáticos
E.T.S.I. Telecomunicación
Universidad Politécnica
28040 Madrid

José María Troya Linero
Dep. Lenguajes y Ciencias de la Computación
Facultad Informática
Universidad de Málaga
29013 Málaga

Sumario

Artículo

— Memoria asociativa bidireccional Fuzzy para reconocimiento de patrones/Francisco L. Aligué, María I. Acevedo y Miguel A. Jaramillo 2

— Entorno de modelado y simulación de sistemas de control en CIM/José L. Villaroel, José A. Fernández, Javier Sainz y Pedro R. Muro 12

Colaboración

— Calidad de software e ISO 9000/Mikel Emaldi y José M. Villate 25

Artículo estudiante

— Inducción de reglas en bases de datos/Antonio J. Gómez 34

Resúmenes de artículos en inglés 46

Sección Abierta 48

Noticias de la AEIA 50

— Informe sobre la XV Escuela de Verano de Informática

— XV EVI. Informe sobre la mesa redonda «Informática y sociedad: demanda social en informática»

— Informe sobre el XIII Curso de Automática en la Industria

Carta de los lectores 54

Recortes 55

Tesis y Libros 58

Agenda de Congresos 61

Publica	Asociación Española de Informática y Automática
Editor Jefe	Fernando Sáez Vacas, Dep. Ingeniería de Sistemas Telemáticos. E.T.S.I. Telecomunicación. 28040 Madrid. E mail:fsaez@dit. upm.es.
Editores Asociados	José Carlos González Cristóbal. Dep. Ingeniería de Sistemas Telemáticos. E.T.S.I. Telecomunicación. 28040 Madrid. E-mail:jgonzalez@dit.upm. es. Juan Manuel Sánchez Pérez. Dep. Informática y Automática. Facultad de Físicas. 28040 Madrid. E-mail: jmasanch@dia.ucm.es

Consejo Editorial

- Arquitectura de ordenadores y diseño lógico:
  Antonio González Colás (por delegación temporal de Mateo Valero), Dep. Arquitectura de Computadores.
  Universidad Politécnica de Cataluña:
  Campus Norte - Edificio D4, c/. Gran Capitán s/n
  08071 Barcelona. E-mail: antonio@ac.upc.es
- Comunicaciones:
  Emilio San Vicente, Dep. Ingeniería Telemática. E.T.S.I. Telecomunicación.
  C/. Jorge Girona Salgado, s/n.
  08034 Barcelona
- Cibernética y control automático:
  Javier Aracil. Dep. Ingeniería Electrónica, de Sistemas y Automática. E.T.S.I. Industriales.
  Avda. Reina Mercedes, s/n.
  41012 Sevilla. E-mail:aracil@etsii.us.es.
- Factores humanos y educación:
  Carmen Fernández Chamizo. Dep. Informática y Automática.
  Facultad de Físicas.
  28040 Madrid. E-mail: cfernan@dia.ucm.es
- Informática teórica y lenguajes:
  Fernando Orejas. Dep. Lenguajes y Sistemas Informáticos.
  Facultad de Informática.
  C/. Pau Gargallo, 5.
  08028 Barcelona.
- Ingeniería del software y software básico:
  José María Troya. Dep. Lenguajes y Ciencias de la Computación.
  Facultad de Informática.
  Plaza El Egido, s/n.
  29013 Málaga. E-mail:troya@ctima.uma.es.
- Inteligencia artificial:
  José Carlos González. Dep. Ingeniería de Sistemas Telemáticos.
  E.T.S.I. Telecomunicación.
  28040 Madrid. E-mail:jgonzalez@dit.upm.es.
- Tratamiento de señales e imágenes:
  Aníbal Figueiras. Dep. Señales, Sistemas y Radiocomunicación
  E.T.S.I. Telecomunicación.
  28040 Madrid.
- Sistemas de información: (Vacante)
- Robótica: (Vacante)

Sede Social
AEIA
C/. Hortaleza, 104, 2.° izquierda
28004 Madrid
Tel.: 319 25 65 - Fax: 308 30 28

Secretaria de Redacción: Elena Navarro

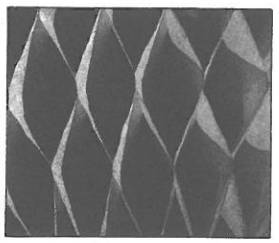
Producción: Studio 5, S.L.
Maquetación: Alfonso Parcet
Fotocomposición: Bocetto Gráfico, S.L.
Impresión: Grafisur, S.L. San José de Valderas (Madrid)
ISSN: 0214-932X
Depósito Legal: M-20.010-1968

La Asociación Española de Informática y Automática fue fundada en 1967 por D. José García Santesmases.

La revista «INFORMÁTICA Y AUTOMÁTICA» se publica trimestralmente, distribuyéndose a los miembros de la Asociación Española de Informática y Automática. La cuota anual de afiliación a la Asociación es de 6.000 pts. para socios numerarios, 1.500 pts. para socios estudiantes y 40.000 pts. para socios jurídicos. El precio de suscripción a la revista (sólo para bibliotecas) es de 12.000 pts.

Salvo que se mencione expresamente lo contrario, todos los artículos publicados en esta revista están protegidos por Copyright, que cubre los derechos exclusivos de reproducción, traducción y distribución.

Ningún material publicado en esta revista puede ser reproducido o almacenado por ningún medio sin disponer autorización por escrito del editor. La AEIA autoriza que se realicen fotocopias del material publicado solamente para uso personal.



Fotografía realizada para la revista por Juan Ramón Velasco Pérez.

Inducción de reglas en bases de datos*

Antonio J. Gómez

Se presenta un sistema propuesto por Quinlan que permite inducir definiciones lógicas a partir de relaciones definidas extensionalmente. Después se proponen nuevos heurísticos con los cuales se corrigen ciertas deficiencias encontradas en el sistema original, mejorando considerablemente la eficacia del sistema y su capacidad de inducción. Finalmente se sugiere su aplicación al descubrimiento en bases de datos (BD) de tipo relacional.

Palabras clave: *Aprendizaje inductivo, definición lógica, cláusula de Horn, relación, literal.*

1. INTRODUCCION

En muchas áreas del saber el conocimiento se ha venido obteniendo por el clásico método hipotético-deductivo de la ciencia positiva, según el cual el investigador aplica su intuición sobre un conjunto de observaciones para formular una hipótesis. Sin embargo, esta «intuición» resulta inoperante cuando no se enfrenta a observaciones aisladas y casuales, sino a millones de datos almacenados en soporte informático, como ocurre en campos como la exploración espacial, la biología, la economía, etc., en los que se aplican potentes técnicas de recogida y almacenamiento de datos. Además, esta recopilación de información se produce a una velocidad mucho mayor que la requerida para su procesado manual, de modo que se incrementa continuamente la diferencia entre la cantidad de datos disponibles y el conocimiento extraído de los mismos. Por ello, es cada vez más necesario desarrollar herramientas que automaticen ese paso inductivo, mediante técnicas de análisis inteligente [2] capaces de inducir reglas cualitativas previamente desconocidas a partir de las BD.

El tema ha sido abordado por numerosos investigadores, y se han propuesto diversos enfoques (algunos pueden encontrarse en [8]. El sistema FOIL (*First-Order Inductive Learner*) de Quinlan [10] y [11] parece muy prometedor, por dos motivos:

- se puede aplicar de forma sencilla a bases de datos relacionales (BDR), lo que implica una gran posibilidad de utilización, ante el predominio del modelo relacional entre las bases de datos existentes;
- el conocimiento que obtiene son definiciones lógicas de relaciones, en forma de reglas compuestas por cláusulas de Horn.

El sistema FOIL tiene, conceptualmente, dos partes, una algorítmica y otra heurística, que se describen en el apartado 2 con una terminología diferente a la usada por Quinlan. La aportación de este artículo se centra en la parte heurística, que ha sido mejorada, adaptando medidas sugeridas en [6], para subsanar algunas deficiencias que se producían durante la construcción de las cláusulas. En el apartado 3 pueden verse tanto los errores encontrados en el comportamiento del heurístico *Ganancia*, usado por Quinlan, como las mejoras introducidas en este sentido. Para comprobar las diferencias que pueden producirse en la regla inducida dependiendo el heurístico de evaluación que se emplee, en el apartado 4 se presentan dos sencillos ejemplos, cuyos resultados preliminares ya fueron mostrados en el congreso *PRODE'92* [4].

2. EL SISTEMA FOIL

2.1. Idea básica

El objetivo de FOIL es obtener una definición lógica de una relación P en función de otras relaciones Q_i y de ella misma (permitiendo definiciones recursivas). Tanto la relación P, que

A. J. Gómez, Dpto. Ingeniería de Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica Madrid, 28040 Madrid. E-mail: anto@dit.upm.es.

* La publicación de este artículo fue recomendada por el jurado del Premio José García Santesmases de informática para trabajos de estudiantes.

es fijada como *relación objetivo* por el usuario, como las demás relaciones Q_i están definidas inicialmente de forma extensional, es decir, por los conjuntos de tuplas para los que se satisfacen los correspondientes predicados.

El resultado dado por FOIL será una regla o definición lógica (intensional) de P, mediante un conjunto de cláusulas de Horn, para verificar cuándo se satisface el predicado **p**. Una regla podrá estar formada por una cláusula de Horn o por la disyunción de varias, siendo la forma general de estas cláusulas la siguiente:

$$p(X_1, ..., X_k) :- L_1, L_2, ..., L_n$$

donde la cabeza de cada cláusula es siempre el predicado objetivo ***p*** con todos sus términos variables $(X_1, ..., X_k)$, y el cuerpo lo forman una serie de literales $(L_1, ..., L_n)$. Estos literales son siempre predicados correspondientes a alguna de las relaciones Q_i definidas inicialmente, construidos sobre términos variables, o bien el predicado igualdad («=») aplicado a dos variables de la cabeza, y pueden llevar el signo «¬» de la negación lógica.

2.2. Algunas definiciones

— Sea $t = <a_1, ..., a_k>$ una tupla formada por k constantes (k es el grado de **p**).

- **t es positiva** (tupla $\oplus$ o ejemplo de p) cuando satisface el predicado **p**:

$$t \in T^+ \text{ sii } \models_t p(a_1, ..., a_k)$$

- **t es negativa** (tupla $\ominus$ o contraejemplo de **p**):

$t \in T^-$ sii $t \notin T^+$ (asunción de mundo cerrado) (*)

- **Conjunto de entrenamiento**: $T = T^+ \cup T^-$

— Sea $C = p(X_1, ..., X_k) :- L_1, ..., L_n$ una cláusula de Horn.

- **C cubre a t** sii $\models_t (L_1 \wedge ... \wedge L_n)$
- **C es consistente** sobre T sii $(\forall t \in T^-)\ (\neg \models_t (L_1 \wedge ... \wedge L_n))$
- **C es completa** sobre T sii $(\forall t \in T^+)\ (\models_t (L_1 \wedge ... \wedge L_n))$
- **Conjunto cubierto por C**: $T_C(C) = \{t \in T \mid \models_t (L_1 \wedge ... \wedge L_n)\}$
- **Relación residual asociada a C**: $T_R(C) \models \{t \in T^+ \mid \neg \models_t (L_1 \wedge ... \wedge L_n)\}$

— Sea $R = C_1 \vee C_2 \vee ... \vee C_j$ una regla (la cabeza de C_i es **p**).

- **Conjunto cubierto por R**: $T_C(R) = T_C(C_1) \cup T_C(C_2) \cup ... \cup T_C(C_j)$
- **Relación residual asociada a R**: $T_R(R) \models T_R(C_1) \cap T_R(C_2) \cap ... \cap T_R(C_j)$
- **R completa** sii $T_C(R) = T^+ \equiv T_R(R) = \emptyset$

(*) También pueden estar declaradas explícitamente las tuplas que no pertenecen a la relación P.

2.3. Algoritmo

2.3.1. FOIL: bucle externo

Construye una regla completa mediante la disyunción de cláusulas consistentes (cubre T^+ con cláusulas consistentes):

```
procedure FOIL (p, T+, T-, q1, q2, ..., VAR R);
    R:= FALSE;  { inicia R }
    TR(R):= T+;
    repetir
        buscaC (p, TR(R), T-, q1, q2, ..., C);
        R:= R ∨ C;  { añade cláusula }
        TR(R):= TR(R) - TC(C);
    hasta que TR(R) = ∅ ;
end.
```

Esta idea es, esencialmente, la misma ya utilizada por algoritmos bien conocidos en el campo del aprendizaje, como el sistema AQ.

2.3.2. buscaC: bucle interno

Construye una cláusula consistente, cuya cabeza es el predicado objetivo:

```
procedure buscaC (p, TR, T-, q1, q2, ..., VAR C);
    C0:= p(X1, ... ,Xk):- TRUE;  { inicia C }
    TL1+:= TR;  { conj local de tuplas ⊕ }
    TL1-:= T-;  { conj local de tuplas   }
    i:= 1;
    repetir
        buscaL (p, TLi, q1, q2, ..., Li);
        Ci:= Ci-1 ∧ Li;  { añade literal (antecedente i) }
        TLi+1+:= modifTL (TLi+, Li); ⊖
        TLi+1-:= modifTL (TLi-, Li);
        INC (i);
    hasta que TLi- = ∅ ;
    C:= Ci-1;
end;
```

La función *modifTL* crea un nuevo conjunto local de entrenamiento TL_{i+1}, seleccionando las tuplas de TL_i que satisfacen

L_i (conservando cada nueva tupla de TL_{i+1} la etiqueta $\oplus$ o $\ominus$ de su antecesor). Cuando L_i introduce nuevas variables, las tuplas de TL_{i+1} expanden su tamaño respecto de las de TL_i, y una tupla de TL_i puede dar lugar a más de una en TL_{i+1}.

2.3.3. *buscaL: selección del mejor literal*

El objetivo de la función *buscaL* es explorar el espacio de los literales candidatos para elegir el mejor entre ellos. Aunque existen algunos heurísticos (apartado 2.4) para restringir un poco la búsqueda, ésta se realiza de forma prácticamente exhaustiva entre todos los literales candidatos.

Para medir la bondad de cada literal, Quinlan utiliza en FOIL un heurístico (***Ganancia***) basado en una medida de la información; de modo similar se comporta otro sistema suyo muy conocido: ID3 [9].

2.4. Heurísticos usados en FOIL

Aparte del algoritmo anterior, el sistema FOIL consta de una serie de heurísticos que determinan su funcionamiento (pueden verse con detalle en [10] y [3]):

— El más importante es la función *Ganancia*, utilizado dentro de *buscaL* para evaluar los literales candidatos y decidir cuál es el mejor. La *Ganancia* se define [10] como

$$Ganancia(L_i) = N_i^{++} \ (Info(TL_i) - Info(TL_{i+1})) \quad (1)$$

llamaremos N_i^+ y N_i^- al número de tuplas $\oplus$ y $\ominus$ respectivamente de TL_i,

N_i^{++} y N_i^{-+} al número de tuplas y de TL_i^+ que satisfacen L_i,

$Info(TL_i) = -\log_2(N_i^+/(N_i^+ + N_i^-))$ a la información aportada por cada tupla $\oplus$ dentro de TL_i

— También en *buscaL* se emplean otros heurísticos para guiar la búsqueda de literales: restricción en la forma de los candidatos, poda alpha-beta y uso de literales determinados.

— Por último, existe un heurístico (usado en *buscaC* y en el bucle externo de *FOIL*) para limitar la complejidad de las descripciones, permitiendo la construcción de definiciones inconsistentes y/o incompletas. Este heurístico se basa en el principio de *Longitud de Descripción Mínima* de Rissanen:

LDE(C) < LDI(C) ⇒ usar descripción explícita (enumeración tuplas cubiertas)

LDE(C) > LDI(C) ⇒ usar descripción implícita (Cláusula C) (2)

considerando que LDI(C) y LDE(C) son, respectivamente, la longitud de descripción implícita y explícita de una cláusula C.

FOIL sólo añade a la regla las cláusulas que, cumpliendo el principio de Rissanen, sean consistentes (precisión(*) = 100 por 100) o bien, a pesar de ser *inconsistentes*, superen cierto umbral de precisión (por ej., el 85 por 100). Cuando no es posible construir ninguna cláusula con estas condiciones, finaliza también la construcción de la regla, aunque no se cubra totalmente T^+, permitiendo reglas *incompletas*.

3. EVALUACION DE LITERALES

La elección de un heurístico que permita evaluar de forma justa los diferentes literales candidatos no es fácil. El empleo de la función *Ganancia*, como hace la versión de FOIL creada por Quinlan, puede provocar la elección incorrecta de un literal, dando lugar a un error en la construcción de la cláusula (y por tanto también en la regla total). En el apartado 3.1 se estudian con detalle algunos de estos errores detectados, y en el apartado 3.2 se sugieren nuevos heurísticos de evaluación para prevenirlos.

3.1. Errores de Ganancia: crítica a FOIL

Durante la construcción de una cláusula, el sistema FOIL utiliza la función *Ganancia* para evaluar los literales candidatos y poder elegir el *mejor* entre ellos. El valor de *Ganancia* para un literal sólo depende de la información de los conjuntos de entrenamiento TL_i y TL_{i+1} (función de la concentración de tuplas $\oplus$) y del número de tuplas $\oplus$ de TL_i cubiertas por L_i (es decir, N_i^{++}); por ello, como vamos a demostrar, este valor de la función *Ganancia* en algunas ocasiones no se corresponde con la *bondad* de un literal.

Debemos recordar que en el sistema FOIL se busca construir definiciones completas, que caractericen a todas las tuplas $\oplus$ del conjunto de entrenamiento T^+, mediante una o varias cláusulas consistentes, y al mismo tiempo es deseable que cada definición tenga el mínimo número posible de cláusulas y cada cláusula el mínimo número de antecedentes, buscando la sencillez de la definición. Se puede definir también la complejidad de una regla R con n cláusulas C_i como:

$$LDI(R) = \sum_{i=1}^{n} LDI(C_i) - \log_2(n!)\,\text{bits} \quad (3)$$

siendo $LDI(C^i)$ la longitud de codificación de una cláusula C^i con i literales:

(*) Se define la *precisión* de una cláusula C como el cociente entre el número de tuplas $\oplus$ cubiertas por C y el número total de las que cubre precisión (C) = card $(T_c^+(C))$/ card $(T_c(C))$.

$$LDI(C_i) = \sum_{j=1}^{i} LDI(L_j) - \log_2(i!)\,\text{bits} \qquad (4)$$

y LDI(L_i) la de un literal L_i:

$$LDI(L_i) = 1 + \log_2(n^o\ relaciones\ en\ BD) + \log_2(n^o\ posibles\ argumentos) \qquad (5)$$

En muchos casos el comportamiento de *Ganancia* es correcto y permite elegir, en cada paso, el literal candidato que dará lugar a la regla más sencilla. Sin embargo, pueden encontrarse ejemplos en los que la elección hecha con *Ganancia* origina la construcción de una regla excesivamente compleja (apartado 4); y dado que estamos trabajando con un sistema que sigue el enfoque simbólico, la construcción de una regla más compleja de lo necesario puede considerarse un error.

Los heurísticos usados por FOIL para limitar la complejidad de las definiciones pueden entrar en funcionamiento y permitir la existencia de una regla incorrecta (incompleta o con cláusulas inconsistentes), si la elección de los literales no es la adecuada. En estos casos, las consecuencias del error producido son más graves que un simple incremento de complejidad en la regla.

Hemos agrupado los errores detectados en el heurístico *Ganancia* en tres tipos, descritos a continuación, aunque sus efectos pueden ser los mismos: desde un simple incremento en la complejidad de la regla hasta la construcción de reglas inexactas.

3.1.1. *Error tipo I*

Supongamos la situación descrita en la figura 1: en el espacio de búsqueda del literal L_i de una cláusula existen dos candidatos L_{ai} (mitad izquierda de la figura) y L_{bi} (mitad derecha), que evaluados sobre un mismo conjunto TL_i generan conjuntos TL_{i+1} con igual número de tuplas ⊕ y ⊖ ; además, ambos cubren el mismo número de tuplas ⊕ en TL_i. La diferencia que existe entre ellos es:

- L_{ai} sólo cubre una tupla ⊖ de TL_i,
- mientras que L_{bi} cubre dos tuplas ⊖ (y el mismo número de tuplas ⊕);

es decir, la precisión de la nueva cláusula C^i obtenida con L_{bi} es menor que la correspondiente a la cláusula resultante de añadir L_{ai}. Por tanto, L_{bi} debería ser el peor de estos candidatos, sin embargo, ambos producen igual *Ganancia*, ya que ésta no considera el número de tuplas ⊖ de TL_i cubiertas por cada literal.

Una elección incorrecta en situaciones de este tipo puede no tener ninguna consecuencia en el resultado final del algoritmo, aunque en el caso peor dará lugar a una regla incompleta: si se supera la complejidad máxima permitida por FOIL para una cláusula (medido en número de bits de codificación) y el principio de longitud de descripción mínima obliga a finalizar su construcción, la precisión de esta cláusula será la que determine si debe ser añadida a la regla o no. Por ello, una cláusula inconsistente en la que haya tenido lugar algún error de tipo I tendrá mayor probabilidad de ser rechazada, finalizando así la construcción con una regla incompleta. Pero incluso si eligiendo el literal que da lugar a la cláusula con mayor precisión (L_{ai}) tampoco se evita la existencia de una regla incompleta, al menos esta regla cubrirá a menos tuplas ⊖ del conjunto de entrenamiento inicial, i.e., su inconsistencia será menor.

Los heurísticos *RI* y *RI**, basados en medidas de interés (apartado 3.2), sí tienen en cuenta el número de tuplas ⊖ de TL_i cubiertas por cada candidato L_i, permitiendo diferenciar entre ambos literales y elegir correctamente el que proporciona mayor precisión, como se comprueba en la figura 1.

3.1.2. *Error tipo II*

En la figura 2 se muestran dos candidatos a ser el siguiente literal L_i de una cláusula en construcción C^{i-1}:

- Uno de ellos (L_{bi}) cubre pocas tuplas ⊕ de TL_i, pero produce un conjunto TL_{i+1} con gran concentración de tuplas ⊕ (i.e. con poca información), de modo que su *Ganancia* es grande.

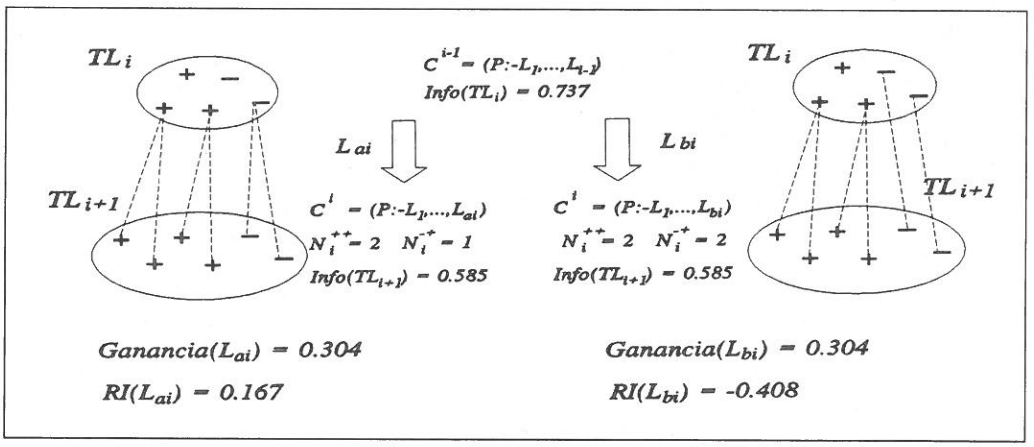


Figura 1. Error tipo I.

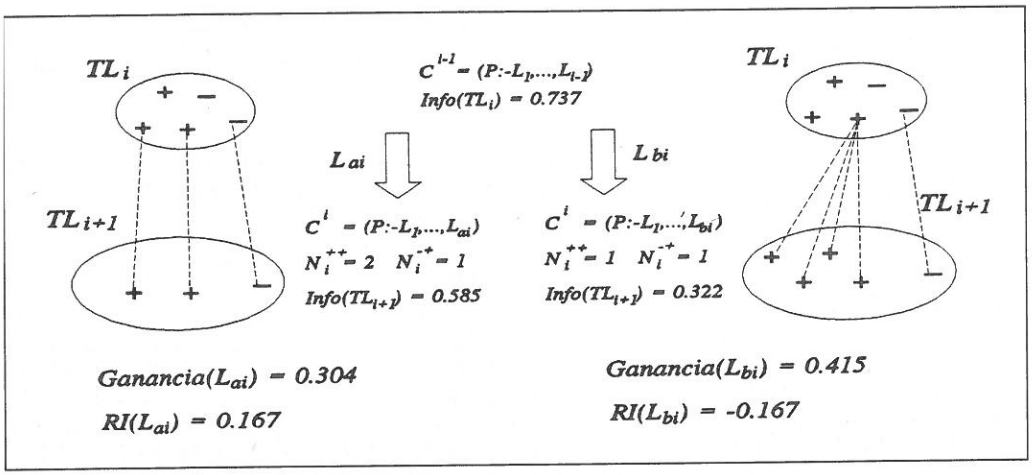


Figura 2. Error tipo II.

- Por el contrario, el otro literal (L_{ai}) genera un TL_{i+1} con menor proporción de tuplas ⊕ (mayor información), lo que hace que su *Ganancia* sea menor que la de L_{bi}, a pesar de cubrir a más tuplas ⊕ en TL_i.

Usando como heurístico de evaluación la función *Ganancia*, FOIL elegirá el literal L_{bi}, favoreciendo así la construcción de una cláusula demasiado específica, es decir, que es satisfecha por pocas tuplas ⊕ dentro de TL_i. Esto obliga a que las reglas construidas necesiten un mayor número de cláusulas para cubrir completamente el conjunto de ejemplos de entrenamiento, lo que repercute en un incremento de la complejidad de las definiciones (medida por su longitud de codificación en bits) y, posiblemente, un incremento también en el coste asociado a su construcción (coste de la búsqueda de literales). Por ello, situaciones de este tipo pueden considerarse como errores de evaluación de literales.

Pero el mayor problema aparece de nuevo cuando una cláusula supera la longitud de descripción permitida y se pone en marcha el heurístico de FOIL que permite la existencia de reglas inexactas: el literal L_{bi}, elegido por su mayor *Ganancia*, tiene menor precisión que el otro candidato L_{ai}, por lo que la precisión de la cláusula construida con L_{bi} será también menor, así como su probabilidad de ser añadida a la regla correspondiente (en el supuesto de permitirse su inconsistencia). Por el contrario, si la elección del literal fuera L_{ai} y también finalizara la construcción de la cláusula (inconsistente), sería más probable superar el umbral de precisión y llegar hasta una regla completa o, al menos, que cubra a mayor número de tuplas ⊕ en el conjunto de entrenamiento que con la opción L_{bi}.

La función *RI* y su modificación *RI'* evitan el error de tipo II, al favorecer al literal (L_{ai}) que cubre a más tuplas dentro de TL_i.

3.1.3. *Error tipo III*

No debemos olvidar que las tuplas ⊕ que realmente queremos cubrir con las cláusulas de Horn que constituyen nuestra definición no son las del conjunto local de entrenamiento TL_i^+ sino el conjunto inicial T^+. Por tanto, al aplicar el heurístico de evaluación *Ganancia* sobre los parámetros medidos en los sucesivos conjuntos TL_i, obtenemos resultados que sólo reflejan de manera indirecta el comportamiento de cada literal. La cobertura por cierto literal L_i de **p** tuplas positivas y **n** negativas en el conjunto TL_i tendrá diferente importancia dependiendo del número **p'** de tuplas ⊕ y **n'** de tuplas ⊖ de TL_i de las que procedan.

En la figura 3 puede verse el problema de forma gráfica. En la mitad izquierda de dicha ilustración se refleja la construcción de una cláusula con dos antecedentes, llamados L_1 y L_{a2}, mientras que en la mitad derecha se pretende reflejar la construcción de otra cláusula con los antecedentes L_1 y L_{b2}; la diferencia entre ambas es precisamente el segundo antecedente. En ambos casos, los candidatos L_{a2} y L_{b2} cubren a tres tuplas ⊕ de TL_2, y producen un nuevo conjunto TL_3 con tres tuplas ⊕ (aunque diferentes para cada candidato). Puesto que los parámetros N_2^{++}, Info(TL_2) y Info(TL_3) son iguales para ambos, la función *Ganancia* también será igual para ambos (de forma similar ocurre con *RI*), a pesar de la gran diferencia existente:

- L_{a2} origina una cláusula consistente y completa (cubre a las tres tuplas ⊕ de TL_1),
- mientras que con L_{b2} se obtiene una cláusula que sólo es consistente (sólo cubre una tupla ⊕ de TL_1).

Por tanto, eligiendo L_{a2} finaliza la construcción de la cláusula y de la regla, pero con L_{b2} sólo finaliza la cláusula, necesitándose nuevas cláusulas para completar la regla.

Este efecto, que no es medido por *Ganancia* (ni por *RI*) al manejar parámetros de TL_i, puede presentarse sólo durante la construcción de cláusulas con dos o más literales. Sus consecuencias pueden ir desde un simple incremento en la complejidad de una regla, debido al aumento del número de cláusulas que la forman (cuando se elige incorrectamente el literal que cubre menos tuplas ⊕ dentro de TL_i), hasta la construcción de reglas inconsistentes y/o incompletas (cuando se ponen en marcha los heurísticos que limitan la longitud de una descripción).

El heurístico llamado *RI*', descrito en el apartado 3.2.3, solucionará el error de tipo III, ya que, aparte de medir el *interés* de los literales en vez de su *Ganancia*, utiliza el conjunto de entrenamiento inicial TL_1 y no el conjunto local intermedio TL_i.

3.2. Heurísticos basados en medidas de interés

3.2.1. *RI: Interés de una regla*

El heurístico llamado *RI* (*Rule Interest*), descrito en [6], es totalmente diferente de la función *Ganancia* usada por FOIL, pues no se basa en la teoría de la información (no mide la información aportada por cada tupla positiva dentro de los conjuntos de entrenamiento), sino que mide el *interés* de una regla, en función del número de casos cubiertos por el antecedente y por el consecuente de la misma.

Podemos llegar a la función *RI* partiendo de las características que deseamos en ella, para luego construirla «a la medida».

Dada una regla $\mathbf{A \rightarrow B}$, sea:

- $\mathbf{N}$ el número total de tuplas que forman el conjunto de entrenamiento;
- $\mathbf{N_A}$ el número de tuplas que satisfacen el antecedente A;
- $\mathbf{N_B}$ el número de tuplas que satisfacen el consecuente B;
- $\mathbf{N_{A \wedge B}}$ el número de tuplas que satisfacen A y B dentro del conjunto de entrenamiento.

Se pretende construir una función *RI* para evaluar el interés de una regla lógica como la anterior. Algunas condiciones que debe cumplir *RI* son:

1) Si A y B son estadísticamente independientes, la regla no tendrá interés ninguno; esto se puede expresar como: $prob(B \mid A) = prob(B) \Rightarrow RI = 0$, y teniendo en cuenta las igualdades

$$prob(A \wedge B) = prob(B \mid A) \cdot prob(A)$$
$$prob(A \wedge B) = N_{A \wedge B} / N$$
$$prob(A) = N_A / N$$
$$prob(B) = N_B / N$$

podemos deducir la condición equivalente:

$$N_{A \wedge B} = N_A \cdot N_B / N \Rightarrow RI = 0 \qquad (6)$$

2) La función *RI* debe crecer monótonamente con $N_{A \wedge B}$, si los demás parámetros son constantes.
3) La función *RI* decrecerá monótonamente al crecer N_A, si los demás parámetros son constantes.
4) La función *RI* decrecerá monótonamente al crecer N_B, si los demás parámetros no varían.

Puede comprobarse que la función más sencilla que cumple estas cuatro condiciones es

$$RI_1(A \rightarrow B) = N_{A \wedge B} - N_A \cdot N_B / N \qquad (7)$$

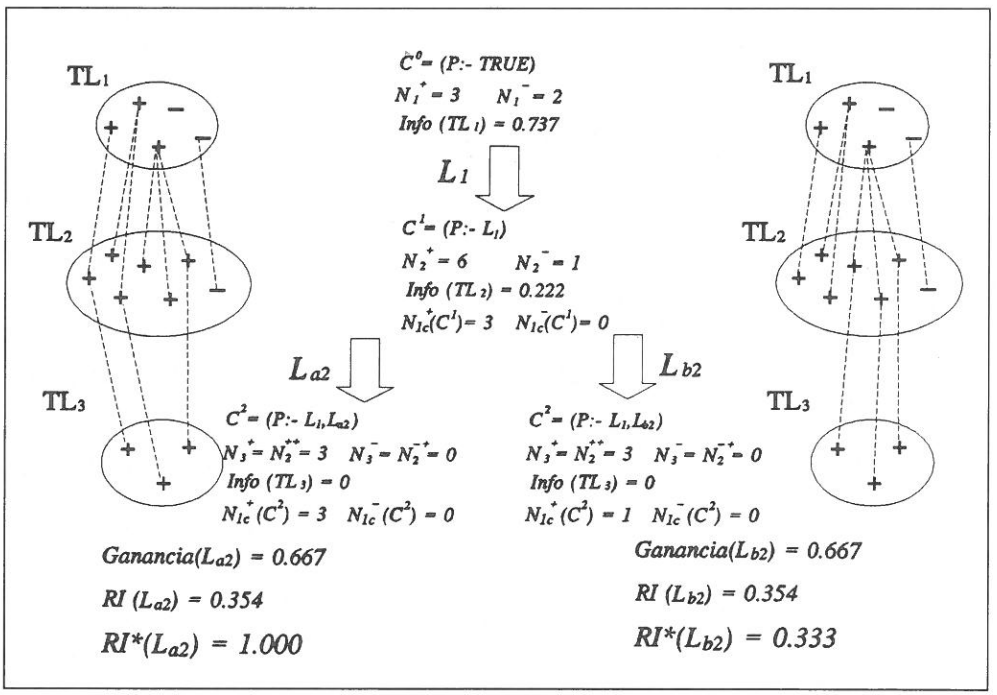


Figura 3. Error tipo III.

que intuitivamente puede ser entendida como la diferencia entre el número real de tuplas para las que se satisfacen A y B (i.e. $N_{A\wedge B}$), y el número esperado de tuplas para las que se satisfarían si A y B fuesen estadísticamente independientes. Otra función que también cumple las condiciones anteriores, y además tiene la ventaja de estar normalizada entre –1 y 1, es la expresión

$$RI_2(A \rightarrow B) = \frac{N_{A\wedge B} - N_A \cdot N_B / N}{\sqrt{N_A \cdot N_B \cdot (1 - N_A/N) \cdot (1 - N_B/N)}} \qquad (8)$$

que no es más que una medida de la correlación entre A y B (esto es un problema estadístico estándar para tablas de contingencia 2 x 2).

Puede ampliarse el conjunto de condiciones que debe cumplir la función *RI*; por ejemplo, incluyendo la condición de que el *interés* de la regla decrezca al aumentar su complejidad. En este caso, si medimos la complejidad de una regla *R* como la longitud necesaria para su codificación, expresada en número de bits (*LDI(R)*), y suponemos que la regla que estamos construyendo define intensionalmente cierta relación P, deberá cumplirse que *RI(R) = 0* cuando *LDI(R)* supere a la longitud de codificación de todas las tuplas que forman la relación objetivo P. El establecimiento de nuevas restricciones para la función *RI* (por ej., añadiendo costes al uso de ciertos predicados o limitando el número de nuevas variables en cada cláusula) deja abierta una posible línea de investigación.

La expresión que utilizaremos a partir de aquí al referirnos a *RI* será la correspondiente a RI_2 (ecuación 8).

3.2.2. *Aplicación de RI al sistema FOIL*

En principio, la función *RI* es aplicable directamente a la medida del *interés* de reglas de la forma **A → B**, que pueden pertenecer a la lógica de proposiciones o a la lógica de predicados de primer orden; esta medida se usa para elegir el mejor antecedente **A** para un consecuente dado **B**. Sin embargo, *RI* puede adaptarse fácilmente para evaluar literales y guiar la construcción de cláusulas de Horn de la forma

$$C^i = (p(X_1, X_2, ..., X_k) :- L_1, L_2, ..., L_i)$$

ya que podemos ponerlas como **A → B** haciendo:

$$\mathbf{A} := L_i$$
$$\mathbf{B} := P(X_1, X_2, ..., X_k) :- L_1, L_2, ..., L_{i-1}$$

gracias a la siguiente equivalencia

$$(L_1 \wedge L_2 \wedge ... \wedge L_{i-1} \wedge L_i) \rightarrow P \equiv P \vee \neg L_1 \vee \neg L_2 \vee ... \vee \neg L_{i-1} \vee \neg L_i \equiv$$
$$\equiv L_i \rightarrow ((L_1 \wedge L_2 \wedge ... \wedge L_{i-1}) \rightarrow P)$$

De este modo, el *interés* de $A \rightarrow B$ permite evaluar los literales L_i candidatos y elegir el mejor entre ellos. Obsérvese que la expresión de *B* es la cláusula C^{i-1} (i.e., C^i antes de añadirle el literal L_i), por lo que los ejemplos y contraejemplos del consecuente *B* forman el conjunto de entrenamiento local TL_i, tal como se definió en el bucle interno del algoritmo FOIL (*buscaC*, apartado 2.3.2).

La relación existente entre los parámetros definidos para *Ganancia* (apartado 2.4) y los definidos para *RI*, cuando se utiliza para evaluar literales de la forma establecida, será:

$\mathbf{N} = \mathbf{N_i^+} + \mathbf{N_i^-}$ = nº total de tuplas en TL_i;

$\mathbf{N_A} = \mathbf{N_i^{++}} + \mathbf{N_i^{-+}}$ = nº tuplas de TL_i que satisfacen L_i;

$\mathbf{N_B} = \mathbf{N_i^+}$ = nº tuplas que satisfacen la condición indicada por B dentro de TL_i, i.e., nº tuplas ⊕ de TL_i cubiertas por la cláusula B (cardinal de TL_i^+);

$\mathbf{N_{A\wedge B}} = \mathbf{N_i^{++}}$ = nº tuplas de TL_i^+ que además satisfacen L_i;

considerando así que para la búsqueda del literal L_i utilizamos como conjunto de entrenamiento TL_i, como se hace en la versión original de FOIL.

De este modo, se llega a una expresión de *RI* (ecuación 8) que es función de:

$$RI(L_i) = f(N_i^+, N_i^-, N_i^{++}, N_i^{-+})$$

por lo que soluciona algunos de los errores vistos en apartado 3.1:

- al usar N_i^- puede corregir el error de tipo I, como se ve en la figura 1.
- al ser *RI* una medida de la correlación entre N_A y N_B, corrige también el error de tipo II (figura 2).

Sin embargo, como sus parámetros son medidos sobre el conjunto local de entrenamiento TL_i, tampoco diferencia entre los candidatos de la figura 3. La solución a este último error se trata a continuación.

3.2.3. *El heurístico RI* para evaluar literales*

Para solucionar el problema de tipo III se puede definir un nuevo conjunto de entrenamiento para la búsqueda de L_i, que llamaremos $T_{1C}(C^{i-1}) = TL_i \cap T_C(C^{i-1})$, con las tuplas pertenecientes a TL_i cubiertas por C^{i-1}. El conjunto $T_{1C}(C^{i-1})$ no es más que la proyección del conjunto local TL_i sobre el conjunto inicial TL_1 (i.e. tuplas ⊕ y ⊖ de TL_1 que han generado alguna tupla ⊕ y respectivamente en TL_i al añadir los literales $L_1,...,L_{i-1}$ a la cláusula actual C^{i-1}). En la búsqueda del primer literal L_i de cada cláusula, este conjunto coincidirá con el inicial: $T_{1C}(C^0) = TL_1$.

Si hacemos:

$A := L_i$

$B := C^{i-1}$

la relación entre los parámetros medidos en $T_{1C}(C^{i-1})$ y los usados en la expresión de $RI(A \rightarrow B)$ (apartado 3.2.1) es:

$N = N_{1C}(C^{i-1})$ = cardinal del conjunto entrenamiento $T_{1C}(C^{i-1})$;

$N_A = N_{1C}(C^i)$ = nº tuplas de TL_1 cubiertas por C^i;

$N_B = N_{1C}^+(C^{i-1})$ = nº tuplas $\oplus$ en $T_{1C}(C^{i-1})$: sus descendientes en TL_1 cumplen la relación P y los literales $L_1, ..., L_{i-1}$;

$N_{A \wedge B} = N_{1C}^+(C^i)$ = nº tuplas $\oplus$ en $T_{1C}(C^{i-1})$ que además satisfacen L_i.

Se puede llamar $RI^*(L_i)$ a la expresión obtenida para $RI(L_i)$ (ecuación 8) (cuando se miden sus parámetros sobre $T_{1C}(C^{i-1})$. Por tanto, $RI^*(L_i)$ será función de

$$RI^*(L_i) = f(N_{1C}^+(C^{i-1}), N_{1C}^-(C^{i-1}), N_{1C}^+(C^i), N_{1C}^-(C^i))$$

y con este heurístico podremos elegir el literal L_i que dé lugar a la cláusula con mayor interés respecto del conjunto inicial de entrenamiento.

Evaluando los literales candidatos con RI^* se evitan los tres errores vistos en apartado 3.1:

- por utilizar la medida del interés *RI*, soluciona los dos primeros errores (como se vio en apartado 3.2.2)
- por utilizar el conjunto de entrenamiento $T_{1C}(C^{i-1}) \subseteq TL_1$, soluciona también el error de tipo III (figura 3).

4. EJEMPLOS. COMPARACION DE RESULTADOS

La diferencia entre las reglas inducidas al utilizar un heurístico de evaluación u otro no se presenta en todos los ejemplos sobre los que hemos realizado pruebas, ya que los errores de comportamiento vistos en apartado 3.1 para la función *Ganancia* sólo aparecen en situaciones concretas. Sin embargo, las consecuencias de estos errores pueden ser desastrosas, como veremos con dos ejemplos.

4.1. Ejemplo de los *enfermos*

Hemos construido una BD con cinco relaciones, cuyo significado es el siguiente:

enfermo(V_1) ≡ la persona V_1 está enferma;

hijo_de(V_1,V_2) ≡ la persona V_1 es hija de V_2;

jefe_de(V_1,V_2) ≡ la persona V_1 es jefe de V_2;

barbudo(V_1) ≡ la persona V_1 tiene barba;

fumador(V_1) ≡ la persona V_1 fuma.

Los atributos usados en todas las relaciones pertenecen a un dominio formado por los nombres de 17 personas diferentes:

Dominio = { *a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3, c4, c5, d1, d2, d3, d4* }

Los hechos que forman la BD son:

enfermo(b3), enfermo(b4), enfermo(c3), ..., hijo_de(b1,a1), ..., jefe_de(a1,b1), ..., barbudo(a3), ..., fumador(b3), ..., fumador(d2).

tal como se representa de forma gráfica en las figuras 4 y 5.

Las líneas rectas verticales que aparecen en la figura 4 entre dos individuos unen a un padre con su hijo (relación *hijo_de*); además, la relación unaria *enfermo* se verifica en las personas dibujadas con la cara triste y sombreada, la relación *fumador* en los que tienen una pipa en la boca, y la de *barbudo* en los que tienen perilla. De un modo similar, en la figura 5 se representan la relación binaria *jefe_de* mediante las líneas verticales, que unen en este caso a cada jefe con sus subordinados; las relaciones unarias de la BD (*enfermo, fumador* y *barbudo*) se visualizan del mismo modo que en la figura anterior.

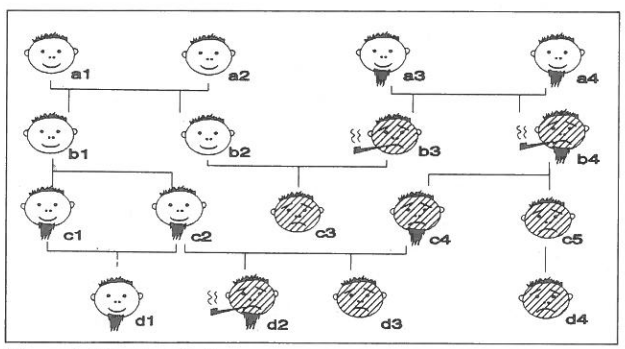


Figura 4. Relación «hijo de».

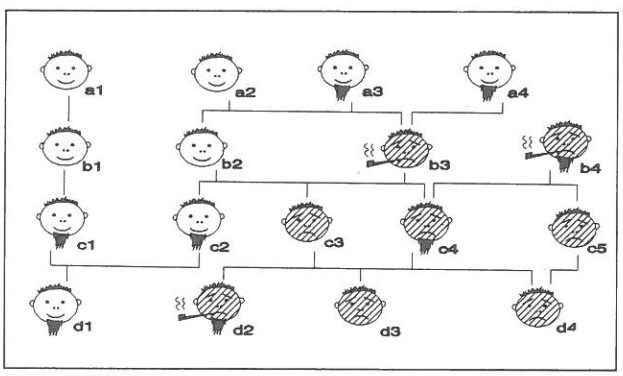


Figura 5. Relación «jefe».

Queremos encontrar una definición lógica de la relación *enfermo* en función de las cinco relaciones existentes en la BD (se invita al lector a que lo intente de forma manual a partir de la observación de las figuras 4 y 5). Aplicando el algoritmo FOIL sobre la BD anterior y eligiendo como relación objetivo *enfermo*, llegaremos a una definición de la misma de forma automática. Según utilicemos como heurístico para evaluar literales la función *Ganancia*, la función *RI* o la función *RI'*, la regla que se construirá será diferente, como vamos a comprobar.

En la figura 6 se muestra el comportamiento de dos de los literales candidatos a ser el primer antecedente de la primera cláusula:

— en la mitad izquierda de la figura se representa el literal

$$L_a = \text{hijo_de}(X_1, V_2)$$

que cubre a las 8 tuplas ⊕ y a las 5 tuplas ⊖ del conjunto inicial TL_1 que aparecen dentro del recuadro punteado (subconjunto $T_{1C}(C^1)$). Puesto que la relación objetivo elegida es *enfermo*, las tuplas ⊕ de TL_1 serán las personas enfermas de la BD y las ⊖ las sanas. El literal L_a genera un nuevo conjunto TL_2 (resultante de modificar las tuplas de TL_1 cubiertas) cuya información es 0.692, por lo que produce una *Ganancia* de 3.165.

— Por otro lado, a la derecha de la figura se muestra el literal

$$L_b = \text{jefe_de}(V_2, X_1)$$

que sobre el mismo conjunto inicial TL_1 cubre a 7 tuplas ⊕ y a 5 tuplas ⊖, es decir, una tupla ⊕ menos que L_a. Sin embargo, produce un conjunto TL_2 con mayor concentración de tuplas ⊕, es decir, menor información (0.585) por lo que la *Ganancia* resultante es 3.518, mayor que la de L_a. Por tanto, la función *Ganancia* favorece a un literal L_b que cubre menos tuplas ⊕ que otro literal L_a (que es óptimo dentro de este espacio de búsqueda), con lo que se produce el error que hemos denominado de tipo II. Si se utilizase el heurístico *RI* o *RI'* para evaluar los literales, puede comprobarse que este error no se hubiera producido.

- La versión original de FOIL, que utiliza la función *Ganancia*, generaría una regla más compleja de lo necesario:

$$R_{\text{Ganancia}} = \begin{array}{l} enfermo(X_1) :- jefe_de(V_2, X_1),\ enfermo(V_2), \\ \neg barbudo(X_1) \\ enfermo(X_1) :- fumador(X_1) \\ enfermo(X_1) :- jefe_de(V_2, X_1),\ fumador(V_2), \\ jefe_de(X_1, V_3),\ enfermo(V_3) \end{array}$$

suponiendo que el heurístico para limitar la complejidad de las reglas (principio de longitud de descripción mínima) no actuase. Sin embargo, en una ejecución real, no se permitiría la construcción de una regla tan compleja como ésta, sino otra más sencilla resultante de eliminar la última cláusula, obteniendo entonces una regla incompleta. Por tanto, el sistema FOIL original no es capaz de inducir una regla lo suficientemente sencilla para describir de forma completa la relación *enfermo* de nuestra BD.

- Si probamos a inducir la definición de *enfermo* usando ahora el heurístico *RI*, el primer antecedente de la primera cláusula se elige correctamente, como acabamos de ver, sin embargo no es así para el segundo. En la figura 7 podemos ver el comportamiento de otros dos literales:

L_a = enfermo(V_2) en la mitad izquierda

L_b = barbudo(V_2) en la mitad derecha

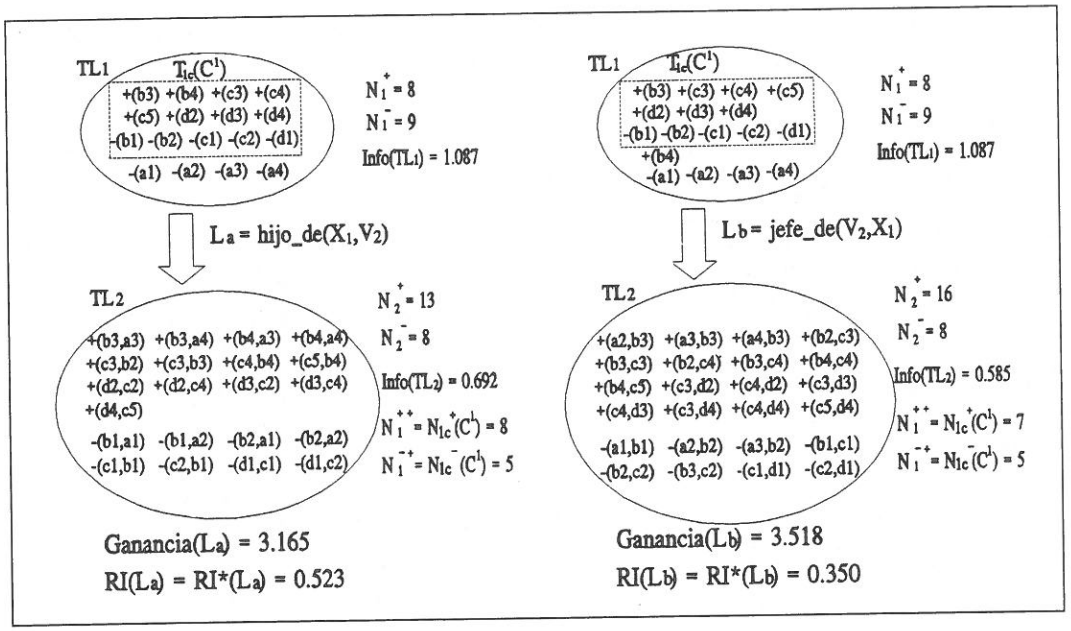


Figura 6. Candidatos para primer literal de primera cláusula.

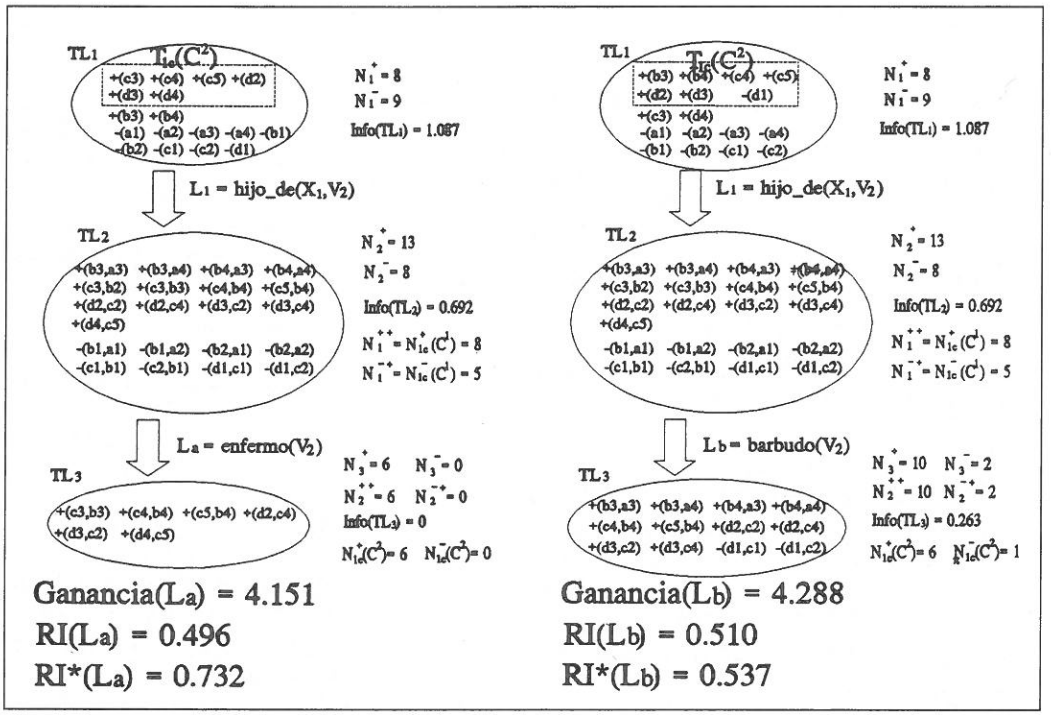


Figura 7. Candidatos para segundo antecedente de primera cláusula.

candidatos para este segundo antecedente. El conjunto TL_1 es el mismo que antes (las tuplas $\oplus$ son enfermos y las $\ominus$ sanos), y el conjunto TL_2 es el resultante de elegir como primer literal L_1=hijo_de(X_1,V_2). Los conjuntos $T_{1C}(C^2)$ correspondientes a cada uno de los candidatos aparecen punteados dentro de TL_1.

El literal L_a genera un conjunto TL_3 con 6 tuplas $\oplus$, cuya proyección sobre TL_1 ($T_{1C}(C^2)$) son también 6 tuplas $\oplus$, es decir, la cláusula que se obtendría con L_a sería consistente. Por el contrario, el conjunto generado por L_b consta de 10 tuplas $\oplus$ y 2 $\ominus$, de modo que se le asigna un mayor valor de *Ganancia* y de *RI*, a pesar de que en el correspondiente conjunto $T_{1C}(C^2)$ haya 1 tupla $\ominus$ y 6 tuplas $\oplus$. De este modo, con el heurístico *RI* (y con *Ganancia*) se elige L_b, produciendo un error de tipo III, mientras que el heurístico *RI** hubiera permitido elegir correctamente el literal L_a (que es el mejor) y finalizaría la construcción de esta primera cláusula.

La regla resultante al utilizar el heurístico de evaluación *RI* define de forma consistente y completa la relación objetivo, aunque no es la más sencilla posible:

$$R_{RI} = \begin{array}{l} enfermo(X_1) :- hijo_de(X_1,V_2),\ barbudo(V_2),\ hijo_de(V_3,X_1) \\ enfermo(X_1) :- hijo_de(X_1,V_2),\ enfermo(V_2) \end{array}$$

- Por último, el resultado obtenido para la prueba con el heurístico *RI** fue la regla:

$$R_{RI^*} = \begin{array}{l} enfermo(X_1) :- hijo_de(X_1,V_2),\ enfermo(V_2) \\ enfermo(X_1) :- fumador(X_1) \end{array}$$

que sí es la más sencilla de las que definen la relación *enfermo* de forma completa y consistente (puede comprobarse con la figura 4 que todos los enfermos son fumadores o hijos de fumadores, y que ninguno de los sanos cumple esta condición).

Si consideramos que la complejidad se puede medir por el número de bits necesarios para codificarla, tendremos los siguientes valores de complejidad para cada una de las reglas anteriores (según expresión (3)):

- para la regla obtenida con *Ganancia* (suponiendo que no se limita su complejidad máxima):
  $LDI(R_{Ganancia}) \approx 11.25 + 3.60 + 16.16 - \log_2(3!) = 28.43$ bits
- para la regla obtenida con *RI*:
  $LDI(R_{RI}) \approx 14.25 + 8.51 - \log_2(2!) = 21.76$ bits
- para la regla obtenida con *RI**:
  $LDI(R_{RI^*}) \approx 8.51 + 3.60 - \log_2(2!) = 11.11$ bits

Por tanto, respecto de la regla completa y consistente que se obtendría con la función *Ganancia*, se produce una simplificación (ahorro de bits) del 23,5 por 100 al usar *RI*, y del 60,9 por 100 al usar el heurístico *RI**. Pero, como ya se ha mencionado, en una ejecución real del algoritmo no se permitiría una regla como la obtenida con *Ganancia*, sino otra resultante de eliminar su última cláusula, es decir, sería una regla incompleta (no cubre dos de las tuplas $\oplus$), aunque consistente.

4.2. Ejemplo de los *abuelos*

Veamos otro ejemplo en el que los resultados obtenidos por el algoritmo de FOIL dependen del heurístico que se emplee

para evaluar los literales. La BDR que usaremos consta de 4 relaciones:

abuelo(V_1) = la persona V_1 es abuela;
padre_de(V_1,V_2) = la persona V_1 es padre de V_2;
varon(V_1) = la persona V_1 es un varón;
tiene_hijos(V_1) = la persona V_1 tiene hijos.

Los atributos usados en esta BD pertenecen a un dominio formado por el nombre de 9 personas:

Dominio = { *a1, a2, a3, b1, b2, b3, b4, b5, b6* }

Las tuplas que forman esta BDR son:

abuelo(a1) abuelo(a2)
padre_de(a1,b4)padre_de(a1,b5)padre_de(a2,b1)
padre_de(a2,b2)padre_de(a2,b3)padre_de(a2,b5)
padre_de(a3,b6)
varon(b1) varon(b2) varon(b3)
tiene_hijos(a1)tiene_hijos(a2)tiene_hijos(a3)
tiene_hijos(b5)

La relación que fijaremos como objetivo será *abuelo*. Los resultados fueron(*):

- Con el heurístico *Ganancia* se obtiene una regla cuya primera cláusula crece excesivamente:

$$R_{\text{Ganancia}} = abuelo(X_1) :- padre_de(X_1,V_2), padre_de(X_1,V_3), padre_de(X_1,V_4), \ldots$$

por lo que finaliza su construcción, dando lugar a una cláusula inconsistente cuya precisión es: *precision* = 2/9 = 0.22, pues cubre a todas las tuplas de la relación objetivo (2 tuplas ⊕ y 7 ⊖). Normalmente el umbral de precisión será más alto, y la cláusula no se añadirá a la regla, de modo que no podría construirse ninguna definición para *abuelo*; en caso contrario, la definición constará de esta única cláusula inconsistente y completa.

- Con el heurístico *RI* se obtendría una regla consistente y completa con respecto al conjunto de entrenamiento, aunque no válida en general:

$$R_{RI} = abuelo(X_1) :- padre_de(X_1,V_2), varon(V_2)$$
$$abuelo(X_1) :- padre_de(X_1,V_2), tiene_hijos(V_2)$$

(su complejidad, medida por su longitud es: $LDI(R_{RI}) \approx 8.23 + 8.23 - \log_2(2!) = 15.46$ bits).

- Por último, con el heurístico *RI** se consigue la definición correcta, que es la más sencilla:

$$R_{RI}^* = abuelo(X_1) :- padre_de(X_1,V_2), tiene_hijos(V_2)$$

(su longitud de codificación es: $LDI(R_{RI^*}) \approx 8.23$ bits < $< LDE(R_{RI^*}) = 8.34$ bits).

5. CONCLUSIONES

Hemos comprobado que ciertos heurísticos permiten mejorar el funcionamiento del sistema FOIL con respecto a la versión dada por Quinlan. En algunos casos la regla inducida es independiente del heurístico usado, pero en muchos otros mejora la «calidad» de las reglas inducidas; incluso en alguno de los ejemplos, se pasa de no encontrar ninguna regla a inducir una definición exacta permitida para la relación objetivo gracias a las mejoras propuestas.

El gran tamaño de algunas bases de datos origina una explosión combinatoria en el número de literales candidatos durante la construcción de las cláusulas (en FOIL se realiza una búsqueda prácticamente exhaustiva), por lo que también ha sido necesario desarrollar nuevos y más potentes heurísticos de búsqueda (tratados en [3]) que aceleren el funcionamiento del sistema. También se han realizado pruebas, con las mejorar propuestas, sobre bases de datos reales, obteniéndose resultados que algunos expertos han calificado de interesantes, porque aparte de ser útiles, eran difíciles de inducir de una forma manual. Además, en el futuro se desarrollará una interfaz que permita al sistema FOIL trabajar sobre bases de datos de tipo relacional, accediendo a ellas con un lenguaje estándar (por ej., SQL), ampliando así las posibilidades de aplicación a problemas reales.

En cualquier caso, el sistema sólo puede inducir una definición lógica para una relación definida extensionalmente y «señalada» como objetivo por el usuario. Sería muy interesante obtener un *sistema de descubrimiento*, que permitiera obtener reglas generales a partir de las relaciones almacenadas en la base de datos, desconociendo a priori cuál es la «relación objetivo». O bien, de manera más ambiciosa, que pueda *formar conceptos*, entendiendo como «concepto» un subconjunto interesante de alguna relación, en el sentido de que tenga una definición lógica sencilla en función de otras relaciones.

(*) La longitud de descripción explícita de la relación objetivo en una BD tan sencilla como la de este ejemplo es muy pequeña:

$$LDE(abuelo) = \log_2(9) + \log_2\left(\binom{9}{2}\right) = 8.34 \text{ bits}$$

por ello, sólo la cláusula obtenida con *RI** cumple que: *LDI(C) < LDE(C)*. Para permitir una mejor comparación de los resultados, en este ejemplo se ha modificado el heurístico que limita la complejidad de las cláusulas, para que actúe sólo cuando: *LDI(C) > 16 bits*.

BIBLIOGRAFIA

[1] Cendrowska, J.: «PRISM: An algorithm for inducing modular rules». En *Knowledge-based systems*, vol. 1, págs. 255-276, 1988.

[2] Frawley, W. J., Piatetski-Shapiro, G., y Matheus, C. J.: «Knowledge Discovery in Databases: An Overview». En *Knowledge Discovery in Databases*, págs. 1-27. G. Piatetsky-Shapiro y W. Frawley, AAAI-MIT Press, Menlo Park, California, 1991.

[3] Gómez, A. J.: *Sistema para aprendizaje de reglas en Bases de Datos Relacionales*. Proyecto Fin de Carrera, Dpto. Ing. Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1992.

[4] Gómez, A. J., y Fernández, G.: «Inducción de definiciones lógicas a partir de relaciones: mejoras en los heurísticos del sistema FOIL». En *Primer Congreso Nacional de Programación Declarativa PRODE'92*, págs. 292-302. Madrid, septiembre 1992.

[5] Michalski, R. S., Carbonell, J. G., y Mitchell, T. M.: *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing, Palo Alto, California, 1983.

[6] Piatetsky-Shapiro, G.: «Discovery and Analysis of Strong Rules in Databases». En *Proceedings of Advanced Database System Symposium' 89*, págs. 135-142. Kyoto, Japón, diciembre 1989.

[7] Piatetsky-Shapiro, G.: «Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop». *AI Magazine*, págs. 68-70, Special Issue, 1990.

[8] Piatetsky-Shapiro, G., y Frawley, W. J. : *Knowledge Discovery in Databases*. AAAI-MIT Press, Menlo Park, California, 1991.

[9] Quinlan, J. R.: «Induction of Decision Trees». *Machine Learning*, vol. 1, págs. 81-106, 1986.

[10] Quinlan, J. R.: «Learning Logical Definitions from Relations». *Machine Learning*, vol. 5, págs. 239-266, 1990.

[11] Quinlan, J. R.: «Knowledge Acquisition from Structured Data. Using Determinate Literals to Assist Search». *IEEE Expert*, págs. 32-37, diciembre 1991.

AEIA

Asociación Española de Informática

Es una asociación sin ánimo de lucro, de carácter científico y técnico, cuyos fines son (artículo 4.º de sus estatutos):

- Organizar cursos, congresos, simposios y coloquios de carácter nacional e internacional que cubran aspectos científicos y técnicos relacionados con la informática y la Automática.
- Contribuir a la representación española en las Asociaciones o Federaciones Internacionales conectadas directamente con la Informática y la Automática.
- Establecer grupos de trabajo para realizar tareas sobre temas de carácter científico, técnico o docente dentro del campo de la Informática y Automática.
- Cooperar con empresas, centros de investigación, de enseñanza u otras entidades relacionadas con la Informática y Automática.
- Divulgar, por medio de publicaciones periódicas, memorias científicas, notas técnicas y noticias referentes a aspectos diversos de la Informática y Automática.
- Asesorar a Organismos o Entidades que así lo soliciten.