3.4. APRENDIZAJE DE REGLAS CON VARIABLES: *FOIL*.

Uno de los sistemas pioneros en el aprendizaje inductivo de reglas con variables -lógica de 1º orden- es FOIL de Quinlan. Numerosos sistemas posteriores son proclamados por sus autores como descendientes de FOIL; este es el caso de FOCL que también se estudiará más adelante, dentro de los sistemas clasificados como sistemas integrados o de revisión de teorías.

- **Generalidades**

FOIL desciende y comparte autor con ID3 y utiliza, a semejanza de este último, una medida basada en la información para decidir el atributo que se debe de elegir en cada momento para completar una regla. Su proceder es, pues, de lo general a lo particular. Sus entradas consisten en un conjunto de predicados o relaciones expresadas extensionalmente; es decir, por las tuplas que las integran. A su vez, las tuplas de cada relación se clasifican en positivas o negativas según ejemplifiquen o no al predicado objetivo, cuya definición en forma de regla y a partir de los otros predicados se trata de captar. Asume la hipótesis del mundo cerrado, por la que toda tupla que no aparezca definida explícitamente como positiva se interpreta como negativa. El conocimiento aprendido lo expresa en la lógica de las cláusulas Horn con la notación de Kowalski, sin símbolos de función pero con literales negativos; aprende, pues, relaciones tales como:

nieta(y,x) ← padre(x,z), padre(z,y), mujer(y)

En la regla anterior el predicado o relación objetivo es *nieta* y viene definido por las relaciones *padre* y *mujer*.

La definición extensional de las relaciones que intervienen en esa regla serían del siguiente estilo:

padre = {(Juan, Pedro), (Pedro, Ana), (Pedro, Andrés), (Andrés, Inés)}
mujer ={Ana, Inés}.

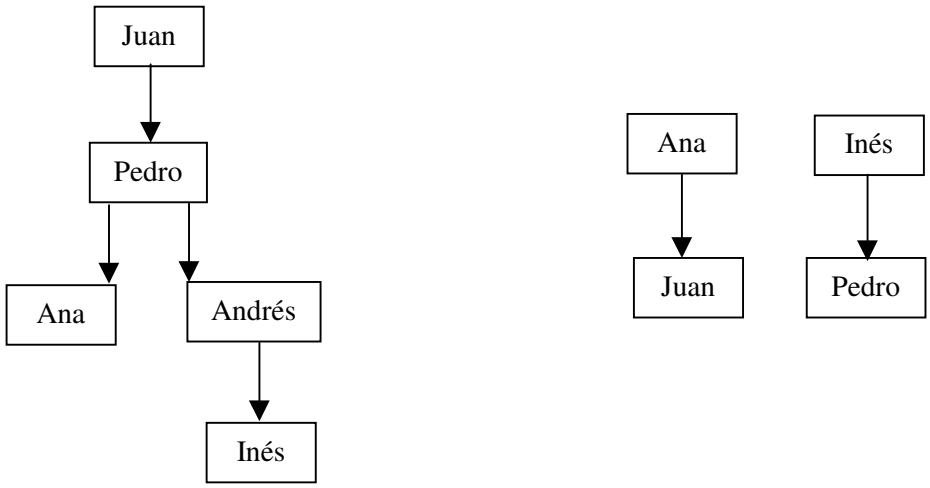

Figura 3.3: Digrafo de la relaciones *padre*, a la izquierda, y *nieta* a la derecha.

Para la relación *nieta* se proporcionarían pares en esa relación, indicándose que son pares positivos, y pares que no están en la relación con indicación de que son pares negativos:

nieta = {(Ana, Juan,+), (Inés, Pedro,+), (Andrés, Juan, -), (Inés, Andrés, -)}

De acuerdo con la hipótesis del mundo cerrado Andrés no estaría en la relación *mujer* por lo que el par (Andrés, Juan) no estaría en la relación *nieta*.

En su estrategia de funcionamiento, FOIL parte de la regla más general con el predicado objetivo como consecuente y sin antecedente, -p.e. *nieta(x,y)* ←, en el caso de la regla anterior- y le va añadiendo los predicados a su antecedente que considera óptimos en el momento de su elección, según una técnica de decisión heurística que explora entre unos tipos preestablecidos de predicados . De esta forma prosigue hasta obtener una regla que cubra solamente ejemplos positivos. Al igual que los algoritmos de recubrimiento, tras eliminar los ejemplos cubiertos por la regla recién construida, continua el cálculo de la siguiente a partir de los ejemplos positivos remanentes. Así hasta alcanzar una familia de reglas que cubra solo a los ejemplos positivos. Los ejemplos negativos se clasifican por defecto.

- **Algoritmo**

FOIL (Predicado Objetivo, Predicados, Ejemplos)

Comenzar
- Comprueba la existencia de relaciones binarias simétricas para evitar una ordenación redundante de las variables.
- Ordena las constantes para prevenir la no terminación desde cualquier cláusula recursiva.
- Inicializa Positivos con las tuplas para las que el predicado objetivo es Verdad*/
- Inicializa Negativos con las tuplas para las que el predicado objetivo es Falso*/
- Reglas_aprendidas:= [ ]
- **mientras** Positivos ≠ { } hacer
  - */*Aprende_una_NuevaRegla*/*
  - NuevaRegla:= Regla que predice el predicado objetivo sin antecedente
  - NuevaReglaNegativa:= Negativos
  - **mientras** NuevaReglaNegativa ≠ { } hacer
    - */*Añadir un nuevo literal para especializar NuevaRegla*/*
    - Literales_candidatos:= nuevos literales candidatos para NuevaRegla, generados basándose en Predicados
    - Mejor_literal:= L∈ Literales_candidatos : GananciaFOIL(L, NuevaRegla) es máxima
    - Añadir Mejor_literal al antecedente de NuevaRegla
    - NuevaReglaNegativa:= subconjunto de NuevaReglaNegativa que no satisfacen el antecedente de NuevaRegla
    - Positivos:= Positivos - {miembros de Positivos cubiertos por NuevaRegla}
  - **fin_mientras**
- **fin_mientras**
- Poda cláusulas -aquí sinónimo de regla- redundantes en Reglas_aprendidas
- Reordena las cláusulas de forma que las no recursivas precedan a las recursivas
- Devuelve Reglas_aprendidas
- Opcionalmente comprueba el rendimiento de las cláusulas aprendidas sobre el conjunto test
- **Fin.**

- **Otros detalles del algoritmo**

Además del proceso principal que refleja detalladamente el pseudocódigo, FOIL incorpora otros subprocesos como son:

- La eliminación de predicados redundantes en el antecedente de una regla. Se sabe que mediante procedimientos voraces, como el que utiliza FOIL, es posible que las reglas obtenidas resulten redundantes, de ahí que este sistema, al igual que otros estudiados -p.e. ARCO-, incorpore una depuración posterior.

- Posibilidad de indicarle que busque reglas recursivas del predicado objetivo; al estilo de:

antecesor(x,y) ← padre(x,y)

antecesor (x,y) ← padre(x,z), antecesor(z,y).

Posee procedimientos que le previenen de una recursión infinita.

- Cuando un nuevo literal incluye variables nuevas en la regla con un consecuente como p.e. *antecesor* se expande este predicado y se reconstruye el conjunto de entrenamiento de ese momento para recoger la expansión. El objetivo de esta expansión es llevar cuenta de las nuevas variables que han de concordar con las del predicado objetivo en esa cláusula.(véase el **ejemplo 3.6**).

- **Detalles sobre la generación de una nueva regla**

Supongamos que en un momento de terminado de la construcción de una nueva regla se tenga:

$$\mathbf{p(x_1,x_2, \ldots ,x_n) \leftarrow l_1,l_2, \ldots ,l_m.}$$

En la elección de un nuevo literal, l_{m+1}, para añadir a la regla que todavía cubre n-tuplas negativas (es decir, que no están en la relación que se está aprendiendo), se tendrán en cuenta los predicados con las siguientes formas:

- ***q*(y_1,y_2, ... ,y_r)**, donde q es un símbolo de predicado de los proporcionados en la entrada y las y_i son variables que aparecen ya en la regla o nuevas variables, aunque al menos una de ellas tiene que aparecer en la regla. Obsérvese que cada combinación de unas mismas variables genera un literal diferente. Cuando se habla de regla durante su construcción ha de entenderse que se refiere a la parte construida de la regla hasta el momento.
- ***igual*(x_j,x_k)**, con ambas variables presentes en la regla.
- La negación de uno de los dos literales anteriores.

Para cada uno de los literales *q* candidatos se tienen en cuenta como argumentos, todos las posibles combinaciones de las constantes que intervienen en el proceso, y que no se hallen implícitamente en la relación contraria, *¬q,* al no aparecer en la definición extensional de la relación q.

La elección de uno de los literales entre todos los candidatos se realiza mediante una optimización que, como se sabe, se basa en la medida de la información de Shannon. Concretamente, se utiliza la Ganancia-FOIL definida como sigue:
donde:

$$\textbf{Ganancia FOIL(R,L)} = t \left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0} \right)$$

- **R** es la regla en construcción en su fase inmediatamente previa a añadir el nuevo literal que salga elegido en el proceso.
- **L** es el literal que se valora para su incorporación en la regla
- **t** es el número de instancias positivas cubiertas por la regla R antes de añadir L y que genera al menos una extensión positiva (véase el ejemplo **3.6**) cubierta por la regla añadir L a su antecedente.
- $\mathbf{p_0}$ es el número de tuplas positivas cubiertas por la regla R antes de la incorporación de L.
- $\mathbf{n_0}$ es el número de tuplas negativas cubiertas por la regla R antes de la incorporación de L.
- $\mathbf{p_1}$ es el número de tuplas positivas cubiertas por la regla R tras la incorporación de L.
- $\mathbf{n_1}$ es el número de tuplas negativas cubiertas por la regla R tras la incorporación de L.

Desde el punto de vista de la Teoría de la Información, $-\log_2(p_0/(p_0+n_{0)})$ es el mínimo número de bits necesarios para codificar la clasificación de una tupla positiva entre las cubiertas por la regla R antes de la incorporación de L. Similarmente, $:-\log_2(p_1/(p_1+n_{1)})$ es el mínimo número de bits necesarios para codificar la clasificación de una tupla positiva entre las cubiertas por la regla R después de la incorporación de L. Por último, si se tiene en cuenta el valor de ***t,*** resulta que se puede interpretar la Ganancia FOIL como la reducción, tras la incorporación de L a R, del número total de bits necesarios para codificar al clasificación de las tuplas positivas cubiertas por R.

- **Un Ejemplo**

Ejemplo 3.6. En del árbol de la figura se trata de captar la relación de ser hoja a partir de la relación de subárbol.

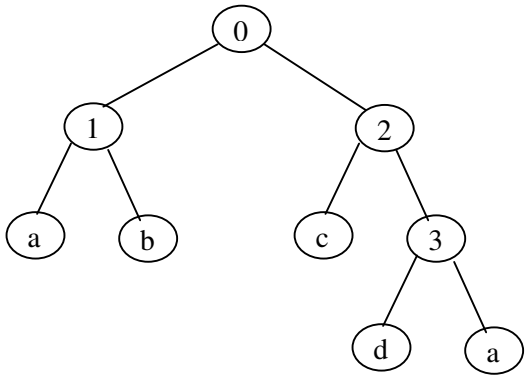

Figura 34 Árbol que se toma de referencia para construir los ejemplos positivos y negativos en el ejemplo 3.6.

- Consideramos la relación de *subárbol(x, y, z)* cuya verdad supone que ***x*** tiene los suárboles ***y*** y ***z***. En le caso del árbol anterior la relación consiste en el siguiente conjunto de pares:

{ (0, 1, 2) (1, a, b) (2, c, 3) (3, d, a) }

- La relación *hoja_de(x, y)* cuya verdad se interpreta como que ***x*** es una hoja del árbol ***y***, e ***y*** no es una hoja. Será la relación objetivo y extensionalmente para el árbol anteior es:

{(a,0) (a,1) (a,2) (a,3) (b,0) (b,1) (c,0) (c,2) (d,0) (d,2) (d,3) }

El conjunto pares de entrenamiento para la última relación es:

(a,0) +	(0,0) -	(1,3) -	(2,c) -	(a,b) -	(c,a) -
(a,1) +	(0,1) -	(1,a) -	(2,d) -	(a,c) -	(c,b) -
(a,2) +	(0,2) -	(1,b) -	(3,0) -	(a,d) -	(c,c) -
(a,3) +	(0,3) -	(1,c) -	(3,1) -	(b,2) -	(c,d) -
(b,0) +	(0,a) -	(1,d) -	(3,2) -	(b,3) -	(d,1) -
(b,1) +	(0,b) -	(2,0) -	(3,3) -	(b,a) -	(d,a) -
(c,0) +	(0,c) -	(2,1) -	(3,a) -	(b,b) -	(d,b) -
(c,2) +	(0,d) -	(2,2) -	(3,b) -	(b,c) -	(d,c) -
(d,0) +	(1,0) -	(2,3) -	(3,c) -	(b,d) -	(d,d) -
(d,2) +	(1,1) -	(2,a) -	(3,d) -	(c,1) -	
(d,3) +	(1,2) -	(2,b) -	(a,a) -	(c,3) -	

El conocimiento aprendido por FOIL en este caso fue:

hoja_de(x,y) ← subárbol(y,x,z), ~subárbol(x,u,v).

hoja_de(x,y) ← subárbol(y,z,x), ~subárbol(x,u,v).

hoja_de(x,y) ← subárbol(y,z,u), hoja_de(x,z).

hoja_de(x,y) ← subárbol(y,z,u), hoja_de(x,u).

Supóngase que la primera regla que construye el sistema en la tercera de las expuestas. Una vez añadido el literal *subárbol(y,z,u)*, p. e. en la construcción de esa regla, se expandió el predicado *hoja_de*(x,y) a *hoja_de*(x,y,z,u), e igualmente el conjunto de entrenamiento quedó modificado de acuerdo con esta expansión de la siguiente manera:

(a,0,1,2) +	(d,0,1,2) +	(1,1,a,b) -	(3,1,a,b) -
(a,1,a,b) +	(d,2,c,3) +	...	...
(a,2,c,3) +	(d,3,d,a) +		
(a,3,d,a) +	(0,0,1,2) -		
(b,0,1,2) +	(0,1,a.b) -		
(b,1,a,b) +	...		
(c,0,1,2) +		...	
(c,2,c,3) +			...

Obsérvese como para el primer par se verifica que (a,0) es un par de la relación hoja, dado en el conjunto de entrenamiento, y la terna (0,1,2) está en la relación subárbol. Por tanto se incorpora al conjunto de entrenamiento con los ejemplos ampliados la cuádrupla (a,0,1,2) relativa a la combinación de variables: (x,y,z,u), como un ejemplo positivo ya que de ese tipo era el ejemplo sin ampliar.

De la misma forma, el par (0,0), que constituye un ejemplo negativo de entrenamiento, junto con la terna (0,1,2), que pertenece a la relación subárbol, generan el ejemplo negativo ampliado (0,0,1,2).

Por el contrario, el par (0,a), que aparece en el conjunto de entrenamiento como ejemplo negativo, no genera un ejemplo negativo ampliado tras la incorporación del literal *suárbol(y,z,u)* a la regla en construcción. Ello se debe a que ninguna terna de la forma (a,z,u) puede ser verdadera para la relación subárbol, por lo que al incorporar ese literal, la regla ya excluye de la definición de *hoja* (en lo que se refiere a la regla en construcción) cualquier ejemplo negativo ampliado que comience por (0,a,_,_); de ahí que no haga falta seguir llevándolos en cuenta como elementos del conjunto de ejemplos ampliados de entrenamiento.

- **Sobre el bias y otras propiedades.**
  - FOIL aprende descripciones generales maximales que diferencian los ejemplos positivos de los negativos.
  - Es incapaz de aprender de ejemplos positivos solamente. Requiere ejemplos negativos y la hipótesis del mundo cerrado.
  - Para aprender definiciones recursivas tiene necesidad de conjuntos de ejemplos positivos *casi* completos (es decir, que contenga a *casi* todos los ejemplos positivos).
  - Es incapaz de aprender nuevos predicados y la carencia de funciones en las cláusulas Horn le restringen la eficiencia en algunos casos.
  - La validez de una cláusula la determinan los ejemplos negativos. No tienen la noción de *demostración* como se encuentra en otros sistemas.
  - Es, quizás, excesivamente sensible a la información irrelevante.
  - no trabaja con valores numéricos, ni es incremental.
  - Es eficiente y capaz de desarrollar definiciones recursivas, reglas inexactas que son útiles y sin oráculo.
  - Es un sistema en desarrollo como su propio autor explica.