Extensión del Modelo de Sistemas Expertos usando Aprendizaje Automático

Sacha A. Bocic David A. Fuller

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Casilla 306
Santiago 22
Chile

Resumen

En este artículo se presenta la incorporación de técnicas de aprendizaje automático al área de sistemas expertos. Para ello, se propone una extensión al modelo tradicional de sistemas expertos que consiste en incorporar una interfaz inteligente. Dicha interfaz se basa en técnicas de aprendizaje automático, con la capacidad de adquirir conocimiento durante la interacción con el usuario. Se presentan dos interfaces, para sistemas expertos con razonamiento con encadenamiento hacia adelante y hacia atrás. Para ambas interfaces se muestran ejemplos e implementaciones en PROLOG.

Palabras Clave: Aprendizaje Automático, Sistemas Expertos, EBG, Interfaz Inteligente.

1. Introducción

Se ha demostrado que los sistemas expertos son herramientas muy útiles para resolver problemas en muchos campos, tales como en planificación y en medicina. Sin embargo, esta tecnología presenta inconvenientes al momento de ser aplicada para resolver problemas de la vida real. Estos inconvenientes los encontramos en las diferentes etapas del ciclo de vida de un sistema experto. Dentro de los problemas más significativos, figuran aquellos relacionados con la etapa de desarrollo y aquellos relacionados con la etapa de utilización. En la etapa de desarrollo, tenemos la dificultad de extraer el conocimiento del experto como una fuerte limitante [7,13,15]. Por otro lado, en la etapa de utilización de un sistema experto, la velocidad de respuesta constituye un serio problema en algunos casos.

Un sistema basado en conocimientos puede efectuar un proceso aprendizaje de dos tipos o niveles [11]. El aprendizaje a nivel del conocimiento consiste en incorporar al sistema, conocimiento externo que no es cubierto por la clausura del conocimiento[1]. El segundo tipo de aprendizaje corresponde a la definición de aprendizaje en [14], es decir, a cambios en el sistemas que son adaptivos en el sentido que permiten al sistema realizar exactamente la misma tarea, sacada de la misma colección de preguntas, en forma más eficiente la próxima vez. En [8] y [12] se presentan trabajos relacionados con la adquisición automática de conocimiento, es decir, con el primer tipo de aprendizaje. En este artículo se presentan técnicas de aprendizaje del segundo tipo, aplicadas al área de sistemas expertos.

Usualmente, en los sistemas expertos encontramos mecanismos de razonamiento que son una mezcla de razonamiento con encadenamiento hacia adelante y hacia atrás[2]. Un ejemplo típico de este tipo de sistema son los de diagnóstico. En efecto, si nos trasladamos al terreno de la medicina, es común que el médico se encuentre con un conjunto de síntomas (antecedentes) a partir de los cuales intenta probar una hipótesis (para lo cual tiene que recurrir a razonamiento con encadenamiento hacia atrás), y así prueba un diagnóstico (conclusión).

Por ejemplo, si se tiene el conjunto de reglas de la figura 1 y se observan los síntomas `pulso_debil` y `taquicardia`, se intenta probar `presion_baja` para concluir `shock`. Así, `presion_baja` será probado con razonamiento con encadenamiento hacia atrás, para luego concluir, usando razonamiento con encadenamiento hacia adelante la condición de `shock`.

1.- La "clausura de conocimientos" corresponde al conocimiento que el sistema puede deducir a partir del conocimiento que se representa en forma explícita en la base de conocimientos.

2.- En inglés, "forward" y "backward chaining", respectivamente.

```
if (pulso_debil and taquicardia and presion_baja) then shock
if (presion_sistolica < 90) then presion_baja
```

Figura 1. Ejemplo de reglas de un sistema de diagnóstico.

Para los efectos de este artículo, hemos decidido tratar los dos tipos de razonamiento por separado para facilitar la comprensión de los distintos aspectos involucrados en cada uno de ellos. Por este motivo, se presentan dos interfaces inteligentes para sistemas expertos, cuyas principales características son la de contar con la capacidad de analizar una consulta, extraer información y almacenarla para su posterior uso. Se asume que las interfaces forman parte de un sistema experto implementado en PROLOG, por lo tanto los ejemplos se presentan en dicho lenguaje. Si el lector no está familiarizado con técnicas de programación y meta-programación en PROLOG, puede consultar [17,1].

2. Modelo Clásico de Sistema Experto

Un modelo sencillo de un sistemas experto consta de cuatro componentes principales: la base de conocimiento, la máquina de inferencia, el generador de explicaciones y la interfaz con el usuario (figura 2).

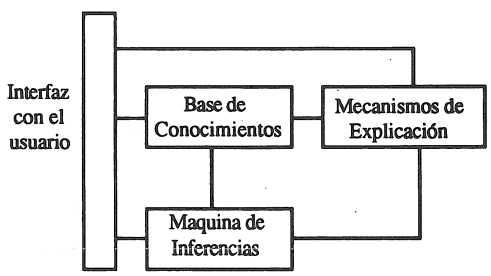

Figura 2. Modelo simple de un Sistema Experto.

La base de conocimientos es el conjunto de información acerca de los objetos y de las relaciones entre estos objetos dentro de un dominio específico y acotado. Existen muchas formas de representar este conocimiento, siendo las más comunes las reglas, los marcos[3] y las redes semánticas. En nuestro caso, lo haremos mediante reglas. La función de la máquina de inferencia es obtener conclusiones a partir de la base de conocimiento. Existen varias estrategias para realizar lo anterior y a modo de ejemplo, podemos mencionar el mecanismo de resolución de PROLOG.

3.- Más conocidos por la palabra en inglés "frames".

Todo sistema experto debe ser capaz de explicar como llega a una conclusión y porqué hace una pregunta al usuario, entregándo información adicional al usuario para que éste entienda lo que se le está preguntando para poder contestar correctamente. Estas tareas las cumple el módulo de mecanismos de explicación. Por último, la interfaz permite comunicar al usuario con la máquina de inferencia y con la base de conocimientos.

3. Extensión al Modelo de Sistemas Expertos

Una de las principales falencias de los sistemas expertos convencionales radica en que éstos no se benefician de la experiencia pasada [15]. Por ejemplo, consideremos un sistemas experto de diagnóstico que frente a un paciente, su historial clínico y síntomas, determina que el paciente sufre de una enfermedad X. Si a este mismo sistema experto se le presenta otro paciente con una historia clínica y síntomas similares, también diagnosticará la enfermedad X porque habrá *repetido* todo el proceso.

Por otro lado, es bastante acéptada la idea que un experto humano es básicamente una persona con mucha experiencia en un área determinada. Por lo tanto, el no considerar este factor en un sistema experto no parece ser razonable. Esto nos hace pensar en la posibilidad de que el sistema se beneficie de la experiencia anterior de tal forma de no tener que repetir todo el proceso de inferencia, logrando un mayor grado de eficiencia. Esto se traduce, en último término, en una mayor velocidad de respuesta del sistema.

Un sistema experto que se beneficia de la experiencia debe ser capaz de analizar casos y extraer información útil para el futuro. Así como también, debe reconocer casos similares a los analizados en el pasado y resolverlos utilizando la información almacenada para tal efecto. En este artículo se propone incorporar al modelo clásico de sistema experto, una interfaz inteligente, cuya función es la descrita. El modelo propuesto se presenta en la figura 3.

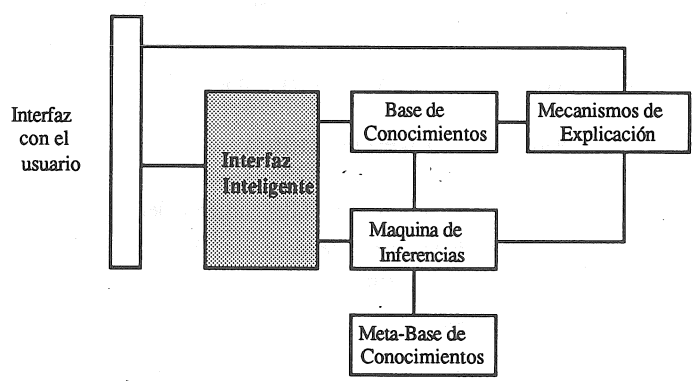

Figura 3. Modelo propuesto de Sistema Experto.

Está claro que la interfaz debe contar con algún tipo de mecanismo de aprendizaje (para extraer y almacenar información) y de razonamiento por analogía (para reconocer casos similares). Existe un gran número de técnicas de aprendizaje automático que pueden ser aplicadas en el área de sistemas expertos [13], sin embargo, al seleccionar el mecanismo de aprendizaje, se debe tener en cuenta el tipo de razonamiento que hace el sistema (con encadenamiento hacia adelante, hacia atrás o combinaciones de ellos) y la forma en que se representa el conocimiento.

Considerando que el conocimiento lo representamos por medio de reglas, es posible recurrir a una técnica de aprendizaje deductivo conocida como Generalización Basada en Ejemplos o más conocida por su sigla en inglés EBG [10]. Esta técnica es muy apropiada para aplicarla en razonamiento con encadenamiento hacia atrás. Para razonamiento con encadenamiento hacia adelante se propone una técnica basada en la anterior. El mecanismo de razonamiento por analogía es bastante primitivo y consiste en considerar similar dos consultas si una de ellas es subsumible a otra. En las siguientes secciones se presentan implementaciones y ejemplos de dos interfaces inteligentes.

4. Interfaz Inteligente para Razonamiento con Encadenamiento hacia Atrás

El razonamiento con encadenamiento hacia atrás parte desde una hipótesis que queremos demostrar y llega a las condiciones necesarias para que ésta sea verdadera, que pueden ser hechos de la base de conocimientos o reglas cuya validez debe ser demostrada. En términos de un árbol de prueba, esto lo podemos describir como la construcción del árbol desde la raíz hacia las hojas.

Ejemplos típicos de sistemas expertos con este tipo de razonamiento son los de planificación. En este tipo de sistema experto, un problema se modela con un estado inicial, un estado final y un conjunto de acciones u operadores. Cada operador o acción se define en términos de sus precondiciones y efectos. Un plan es una secuencia de operadores, que al ser aplicados en orden, permiten alcanzar el estado final a partir del estado inicial [5].

4.1. Generalización Basada en Ejemplos (EBG)

EBG es definido en [10] como "un método independiente del dominio para guiar la generalización usando conocimiento dependiente del dominio". Para los efectos de este artículo, se presenta una descripción de un sistema EBG como parte de un sistema experto con razonamiento con encadenamiento hacia atrás.

Bajo esta perspectiva, podemos decir que los cuatro parámetros de entrada que recibe todo sistema EBG los "toma" del sistema experto. El primer parámetro, el **concepto ob-**

jetivo o lo que se desea aprender, es una hipótesis que el sistema experto recibe para intentar demostrar, pero usando variables libres. El segundo parámetro es el llamado **ejemplo de entrenamiento**. En términos de sistemas expertos, corresponde a toda la información necesaria para que el sistema experto pueda demostrar una instancia del concepto objetivo, es decir, pueda demostrar que la hipótesis que recibe el sistema experto con las variables instanciadas es verdadera.

Para probar que el ejemplo de entrenamiento es una instancia del concepto objetivo, EBG construye un árbol de prueba [9] que tiene como nodo raíz a la hipótesis instanciada. Este árbol se llama "árbol del ejemplo de entrenamiento". Paralelamente se construye el "árbol generalizado", que es un árbol de prueba que tiene como nodo raíz al concepto objetivo. Para construir ambos árboles de prueba se utiliza el conjunto de reglas y hechos que provee la base de conocimientos del sistema experto. La base de conocimientos corresponde a la **teoría de dominio**, es decir, al tercer parámetro de entrada a EBG.

Ambos árboles se construyen idénticos (para cada nodo que se expande, ambos árboles seleccionan la misma regla) hasta alcanzar los predicados "operacionales". El "**criterio operacional**" determina si un predicado es o no operacional. La definición de este criterio es dependiente del dominio, por lo tanto, depende del sistema experto. Este criterio corresponde al cuarto parámetro de entrada a EBG. Usualmente se definen como operacionales, a los predicados que representan una pregunta al usuario, los hechos (facts) y aquellos predicados fácil de calcular.

Cuando en una rama del árbol del ejemplo de entrenamiento se alcanza un predicado operacional, se suspende la expansión de la rama correspondiente al árbol generalizado y se expande completamente la que corresponde al árbol del ejemplo de entrenamiento.

Por último, el proceso de reformulación no es más que la conjunción de todos los nodos hoja del árbol generalizado, es decir, de todos los nodos operacionales del árbol generalizado. De esta forma se obtiene una nueva definición para el concepto objetivo pero que es más fácil de calcular que la original (por lo tanto más eficiente).

A modo de ejemplo, consideremos el conjunto de reglas de la figura 4. Para este caso, definimos como operacional a los predicados: `u(X)`, `t(X)` y `v(X)` y supondremos que además se tiene la suficiente información para demostrar `p(a,b)`, es decir, es posible construir un árbol de prueba completo para `p(a,b)`.

```
p(X,Y):- q(X),r(Y).          u(X):- ...
q(X):- s(X),t(X).            v(X):- ...
s(X):- u(X).                 t(a).
r(Y):- v(Y).                 t(b).
.....
```

Figura 4. Ejemplo de reglas para EBG.

En la figura 5, el árbol de la izquierda corresponde al del ejemplo de entrenamiento, es decir, `p(a,b)` y el de la derecha es el árbol generalizado para el concepto objetivo `p(X,Y)`. La reformulación que se obtiene es `p(X,Y):- u(X),t(X),v(Y)`, que corresponde a la regla aprendida por el sistema.

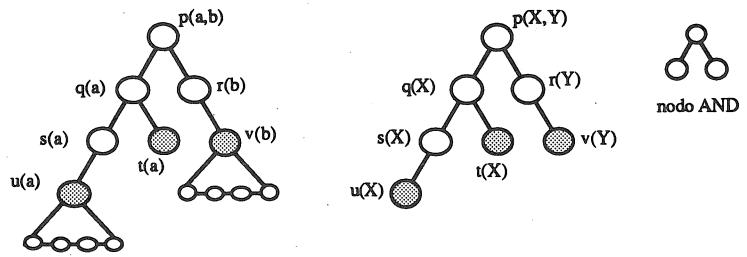

Figura 5. Arboles de prueba para las reglas de la figura 4.

4.2. Interfaz Para Razonamiento con Encadenamiento Hacia Atrás

EBG recibe como entrada un concepto objetivo, una teoría de dominio, criterio operacional y ejemplo de entrenamiento y entrega como salida una reformulación para el concepto objetivo en términos de predicados operacionales.

La idea más útil que aporta EBG es la de reformular un predicado en términos de otros más fácil de calcular. Es así como podemos pensar en una interfaz inteligente que ante cada hipótesis sugerida, chequea si ha sido procesada una hipótesis similar en el pasado, de tal forma de usar la información que se tiene almacenada para tal efecto. Si no ha sido procesada, la interfaz debe tomar la hipótesis sugerida (reemplazando las constantes por variables libres) como concepto objetivo y generar una reformulación para dicha consulta, de tal forma que en el futuro se pueda contestar la misma consulta en forma más eficiente. El ejemplo de entrenamiento lo constituye, en este caso, el conjunto de predicados operacionales instanciados que se utilizan para la demostración de la hipótesis. Se asume que el criterio operacional ha sido definido. En [2] se presenta una implementación de una interfaz inteligente con las características mencionadas pero basada en una extensión a EBG [2] que permite generar reformulaciones más eficientes.

4.2. Ejemplo

A modo de ejemplo consideremos el clásico problema de planificación del *mono y las bananas* . Este problema consiste en que se tiene un mono en una pieza y una banana colgada del techo. El mono debe alcanzar la banana, para ello cuenta con una serie de objetos (mesa, silla, garrote, etc.) con los cuales se puede ayudar. El objetivo es generar una secuencia de acciones para que el mono logre su objetivo. La solución se presenta en la figura 6 en forma de un programa en PROLOG, la que consideraremos (para los efectos de este artículo) como un sistema experto de planificación.

```
plan(A,B,Plan):- sameXY(A,B,P), sameZ(A,B,[],P,Plan).

sameXY(A,B,P):- position(A,From), position(B,To),
                P=['move ',A,'from:',From, 'to:', To].

sameZ(A,B,OldObjs,Pi,P):- height(Obj,Hp), height([B],Hb), Hp<Hb,
                          new(OldObjs,NewObjs,Obj),NewP=['Get:',Obj],
                          append(Pi,NewP,Po),sameZ(A,B,NewObjs,Po,P).

sameZ(_,B,OldObjs,P,P):- height(OldObjs,Hp), height(B,Hb), Hp>=Hb.

height([],0).

height(Target,H):- target(Target,H).

height([X|Xs],N):- h(X,N0), height(Xs,N1), N is N1 + N0.

new(Olds,News,Obj):- object(Obj,_),\+ member(Obj,Olds)[4],
                     append([Obj],Olds,News)[5].
```

Figura 6. Ejemplo del mono y las bananas.

En esta solución, el objeto que se instancia en el primer argumento logra alcanzar al objeto instanciado en el segundo, si ejecuta la secuencia de planes que se instancia en el tercer argumento. Luego, si consideremos el conjunto de hechos de la figura 7 como un ejemplo de entrenamiento y como concepto objetivo la consulta `plan(A,B,Plan)`, para utilizar la interfaz inteligente sólo falta definir el criterio operacional.

```
object(mono,5).        position(mono,(0,0)).
target(banana,16).     position(banana,(12,21)).
object(caja,8).        object(silla,6).
```

Figura 7. Ejemplo de entrenamiento.

Antes de definir el criterio operacional, es necesario aclarar que para un mismo sistema es posible definir más de un criterio operacional. También es importante notar que la definición

---

4.- El predicado `member/2` es verdadero si el término que se instancia en el priemer parámetro pertenecea a la lista que se instancia en el segundo parámetro.

5.- El predicado `append/3` es verdadero si el tercer argumento es sintácticamente igual a la concatenación de los dos primeros.

del criterio es una decisión de diseño que tiene directa repercusión sobre la calidad de la reformulación y a la sobrecarga asociada a la interfaz inteligente. Una vez aclarado lo anterior, definimos como operacional para la teoría de dominio de la figura 6 a los predicados: `pos/2`, `height/2`, `new/3` y los operadores aritméticos '<' y '>='.

Luego podemos ocupar la interfaz inteligente en el ambiente PROLOG y hacer la consulta `interfaz(plan(mono,banana,Plan))`, (el predicado `interfaz/1` define la interfaz) con lo que se obtiene como resultado la instanciación de la variable `Plan` (en el plan requerido) y como efecto lateral se le agrega a la base de conocimientos la regla de la figura 8. Además se agrega a la meta-base de conocimientos información que indica que existe información para la consulta recién procesada.

```
plan(A,B,['move:',A,' from:',P1,' to:',P2,' get:',Obj1,' get:',Obj2]):-
        pos(A,P1), pos(B,P2), height([A],H1), height(B,H2), H1 < H2,
        new([A],Objs1,Obj1), height(Objs1,H3), height(B,H4), H3 < H4,
        new(Objs1,Objs2,Obj2), height(Objs2,H5), height(B,H6),H5 >= H6.
```

Figura 8. Reformulación de una regla utilizando la Interfaz Inteligente.

De esta forma, la siguiente vez que se quiera obtener un plan para un objeto "A", primero se intentará utilizar la nueva reformulación, si esta falla, se efectuará un proceso de aprendizaje basado en la nueva consulta.

5. Interfaz Inteligente para Razonamiento con Encadenamiento hacia Adelante

El razonamiento con encadenamiento hacia adelante consiste en derivar una conclusión a partir de los antecedentes. En términos de árboles de prueba, esto consiste en partir la construcción desde las hojas (antecedentes) para llegar a la raíz (conclusión). El mecanismo de aprendizaje que se propone para esta interfaz está basado en EBG, pero con la diferencia que construiremos el árbol de prueba desde las hojas hacia la raíz. Este mecanismo, que llamamos EBG_BU (por EBG Bottom Up), recibe como parámetros de entrada una **teoría de dominio** y un **ejemplo de entrenamiento** en forma de un conjunto de antecedentes (que tienen las variables instanciadas). Estos dos parámetros son comparables a la teoría de dominio y ejemplo de entrenamiento de EBG. El otro parámetro que recibe EBG_BU le llamamos **antecedentes generalizados**, que corresponde al mismo conjunto de literales del ejemplo de entrenamiento, pero con la sustitución de las variables instanciadas por variables libres. Este parámetro es análogo al concepto objetivo de EBG.

El mecanismo comienza seleccionando reglas en cuyos cuerpos aparecen los literales del ejemplo de entrenamiento. Cada vez que encuentra una regla en cuyo cuerpo aparecen solamente literales del ejemplo de entrenamiento, se efectúa un proceso de reducción, que consiste en eliminar del ejemplo de entrenamiento los literales que aparecen en el cuerpo de la regla y

agregar a dicho conjunto la cabeza de la regla. Es así como se genera el árbol de prueba en forma ascendente. Paralelamente, se mantiene una copia del árbol que se genera, en el cual las variables que aparecen instanciadas en el ejemplo de entrenamiento se reemplazan por variables libres (árbol generalizado).

El último parámetro que recibe es el **criterio de detención**, que es análogo al criterio operacional de EBG, es decir, determina hasta que punto se construye el árbol generalizado. Para los efecto de este artículo, se considera que el árbol de prueba se construye hasta la raíz, es decir, completo. Una vez que se termina la construcción del árbol, se genera la reformulación que no es más que una regla que tiene el nodo raíz como consecuente y los nodos hoja como antecedente.

A modo de ejemplo, consideremos el conjunto de reglas de la figura 4 y supongamos que el criterio de detención nos dice que la construcción se detiene al llegar al nodo raíz y que tenemos como punto de partida `u(a),t(a)` y `v(b)`. La reformulación es: `p(X,Y):- u(X),t(X),v(Y)`. Esto se muestra en la figura 9.

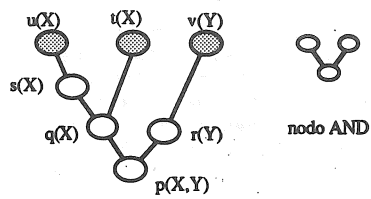

Figura 9. Ejemplo de reformulación bottom-up.

5.1. Ejemplo

A modo de ejemplo, consideremos el conjunto de reglas de la figura 10. Estas reglas representan una parte de un sistema experto para el diagnóstico de la metrorragia (sangramiento de los genitales) durante la segunda mitad del embarazo. Una implementación de una interfaz inteligente para razonamiento con encadenamiento hacia adelante se presenta en [2].

```
diagnostico(Paciente,ru):- sintomas(Paciente,ru),
                           historiaClinica(Paciente,ru),
                           examenMedico(Paciente,ru).

sintomas(Paciente,ru):- she[6](Paciente,positivo),
                        inicioSangramiento(Paciente,brusco).

historiaClinica(Paciente,ru):- cicatriz(Paciente,presente).
```

6.- "she" es una abreviación de Síndrome Hipertenso del Embarazo.

```
examenMedico(Paciente,ru):- hemorragia(Paciente,mixta),
                            shock(Paciente,positivo),
                            dolor(Paciente,presente),
                            feto(Paciente,Palpable).

hemorragia(Paciente,mixta):- hemorragia(Paciente,externa),
                             hemorragia(Paciente,interna).

shock(Paciente,positivo):- presion(Paciente,baja),
                           taquicardia(Paciente,presente),
                           colorPiel(Paciente,palido),
                           pulso(Paciente,debil).

presion(Paciente,baja):- presionSistolica(menorQue90).

taquicardia(Paciente,presente):- frecuenciaPulso(mayorQue100).
```

Figura 10. Reglas de Sistema Experto de Diagnóstico de la Metrorragia de la Segunda Mitad del Embarazo.

En la figura, sólo es posible derivar un diagnóstico (rompimiento uterino o "ru"), el cual depende de los síntomas e historia clínica de la paciente, además del examen médico que se le efectúe. Ahora, podemos utilizar la interfaz inteligente (definida por el predicado `interfaz_bu`) de la siguiente forma:

```
?- interfaz_bu((colorPiel(ana,palido), pulso(ana,debil), she(ana,positivo),
                hemorragia(ana,externa),feto(ana,palpable),
                hemorragia(ana,interna), frecuenciaPulso(ana,mayorQue100),
                inicioSangramiento(ana,brusco),
                cicatriz(ana,presente),dolor(ana,presente),
                presionSistolica(ana,menorQue90)),Head).
```

con lo que se obtiene como solución la instanciación del segundo argumento de `interfaz_bu/2` en `diagnostico(ana,ru)`. Como efecto lateral se agrega a la meta-base de conocimientos información que indica que existe información para la consulta recién procesada y la regla recientemente aprendida (que se muestra en la figura 11) se agrega a la base de conocimientos.

```
diagnostico(P,ru):- colorPiel(P,palido), pulso(P,debil), she(P,positivo),
                    hemorragia(P,externa), feto(P,palpable),
                    hemorragia(P,interna), frecuenciaPulso(P,mayorQue100),
                    inicioSangramiento(P,brusco), cicatriz(P,presente),
                    presionSistolica(P,menorQue90), dolor(P,presente).
```

Figura 11. Resultado de la reformulación usando EBG_BU.

6. Conclusiones

En este artículo hemos presentado dos interfaces inteligentes, una para razonamiento con encadenamiento hacia adelante y otra para razonamiento con encadenamiento hacia atrás. La utilidad y funcionalidad de ambas interfaces se mostró mediante ejemplos simplificados de sistemas expertos de las áreas de diagnóstico y planificación.

La ventaja de la utilización de estas interfaces radica en el incremento en la velocidad de respuesta que obtiene el sistema. Este incremento se produce como consecuencia de efectuar un proceso de aprendizaje (según la definición de [14]), al no tener que construir todo un árbol de prueba (cuando se tiene información sobre la consulta), ya que se utiliza una regla que es equivalente, pero más fácil de calcular. Además, se evita intentar probar ramas fallidas en el proceso de construcción del árbol de prueba La magnitud de este beneficio depende de la forma del espacio de búsqueda que define el sistema experto. Así por ejemplo, en un espacio profundo y angosto, el beneficio es más significativo que en uno ancho y bajo, como se aprecia en la figura 12.

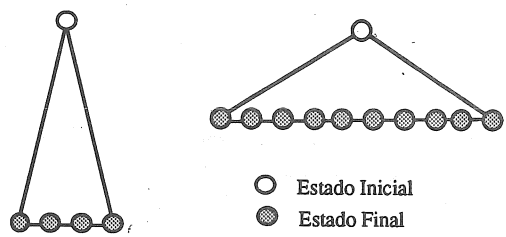

Figura 12. Tipos de espacios de búsqueda.

Un caso que no se ha tratado es el de las múltiples reformulaciones. En efecto, para una consulta es posible obtener más de una reformulación, lo que supone generar más de una regla. En [6] se muestra como EBG puede llevarnos a un caso de "explosión disjuntiva", en el cual la eficiencia del sistema se degrada al agregar nuevas reglas. Distintas formas de atacar este problema se presentan en [2].

Un punto que hay que considerar es que la incorporación de una interfaz inteligente implica una sobrecarga al sistema. En efecto, en términos de meta-programación, se habla de agregar una capa más al sistema y como se sabe, cada capa degrada el rendimiento en un orden de magnitud [17]. En las dos interfaces mostradas, la sobrecarga afecta al sistema durante el proceso de aprendizaje, es decir, cuando no hay información acerca de la consulta. Cuando se tiene información, la consulta es procesada por el intérprete de PROLOG, por lo tanto, la sobrecarga es eliminada. En consecuencia, la utilización de interfaces inteligentes es adecuada para sistemas en los cuales el número de procesos de aprendizaje es mucho menor que el de procesamiento de consultas sobre las cuales se tiene información.

Por otro lado, vemos que el proceso más importante que realizan ambas interfaces consiste en la construcción de un árbol de prueba. En la medida que este proceso sea eficiente, la interfaz inteligente significará menos sobrecarga al sistema total. En [2,3,4] se presentan extensiones a los algoritmos clásicos de EBG y evaluación parcial, cuya incorporación puede mejorar considerablemente la eficiencia de las interfaces.

Dado que es posible definir los criterios operacional, de reflectabilidad y de detención de muchas formas para un mismo sistema experto y considerando que estas definiciones repercuten en la calidad de las reformulaciones y en la sobrecarga del sistema, se propone como trabajo futuro el estudio de algún mecanismo que permita generar dichas definiciones en forma automática.

También se propone como trabajo futuro el diseño e implementación de una interfaz inteligente para razonamiento mixto (hacia adelante y atrás) ya que en la práctica, casos en los cuales sólo hay un tipo de razonamiento es difícil de encontrar. Por último, se propone como trabajo futuro la incorporación de aprendizaje inductivo a interfaces inteligentes de sistemas expertos.

Agradecimientos

Deseamos manifestar nuestro agradecimiento al Dr. Andrés Kychentall por su valiosa colaboración en el sistema experto de diagnóstico de la metrorragia de la segunda mitad del embarazo.

Este trabajo ha sido financiado gracias al apoyo del Fondo Nacional de Ciencia y Tecnología, FONDECYT, proyectos 90-0688 y 89-0591 y a los Proyectos de Fomento a la Investigación 91/21 y 90/8 de la Dirección de Investigación de la Pontificia Universidad Católica de Chile.

Referencias

[1] I. Bratko, Prolog Programming for Artificial Intelligence, Wokingham et al.: Addison Wesley (2nd Edition) (1990).

[2] S. Bocic, Manipulación Simbólica en Sistemas Basados en Conocimiento, Tesis de Magister, P. Universidad Católica de Chile, Santiago, (1992).

[3] D. Fuller, S. Bocic, "Un Experimento en Aprendizaje Automático y Evaluación Parcial". En actas de la XVII Centro Latinoamericano de Estudios en Informática (CLEI), Caracas, Venezuela, (1991).

[4] D. Fuller, S. Bocic, "Extending Partial Evaluation in Logic Programming", COMPUTER SCIENCE: Research and Applications, pp. 95-107, Plenum Press, N.Y., (1992).

[5] H. Hendler, A. Tate, M. Drummond, "AI Planning: Systems and Techniques". AI Magazine, pp. 61-77, Verano (1990).

[6] H. Hoppe, "An Analysis of EBG and its Relation to Partial Evaluation: Lessons Learned", Reporte técnico No. 572, GMD - IPSI, Alemania, Sept. (1991).

[7] P. Jackson, Introduction to Expert Systems, Addison Wesley, (1986).

[8] J. Kornell, "Automated Knowledge Acquisition and Representation", Reporte técnico, General Research Corporation, Santa Barbara, CA, Estados Unidos, (1984).

[9] J. Lloyd, Foundations of Logic Programming, Springer-Verlag, (1984).

[10] T.M. Mitchell, R.M. Keller, S.T. Kedar-Cabelli, "Explanation-based Generalization: A unifying view", Machine Learning Vol I, (1986).

[11] A. Newell, "The Knowledge Level", Artificial Intelligence, No. 18, pp. 87-127, (1982) .

[12] M.A. Newstead, R. Pettipher, "Knowledge Acquisition for Expert Systems", Electrical Communication, Vol. 60, No 2, pp. 115-121, (1986) .

[13] R. Reich, S. Fenves, "The Potential of Machine Learning Techniques for Expert Systems". AI EDAM, pp. 175-193, (1989).

[14]. H.A. Simon, Why Should Machines Learn?, en Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell y T.M. Mitchell (Eds), (1983).

[15] S. Slade, "Case-Based Reasoning: A Research Paradigm", AI Magazine, pp. 42-55, Primavera (1991).

[16] L. Sterling, R.D. Beer, "Incremental Flavor-Mixing of Meta-Interpreters for Expert Systems Construction", en Proc. 3rd Symposium on Logic Programming, pp.20-27, Salt Lake City, Estados Unidos, (1986).

[17] L. Sterling, E. Shapiro, The Art of Prolog. The MIT Press, Cambridge, Massachusetts, Londres, Inglaterra, (1986).