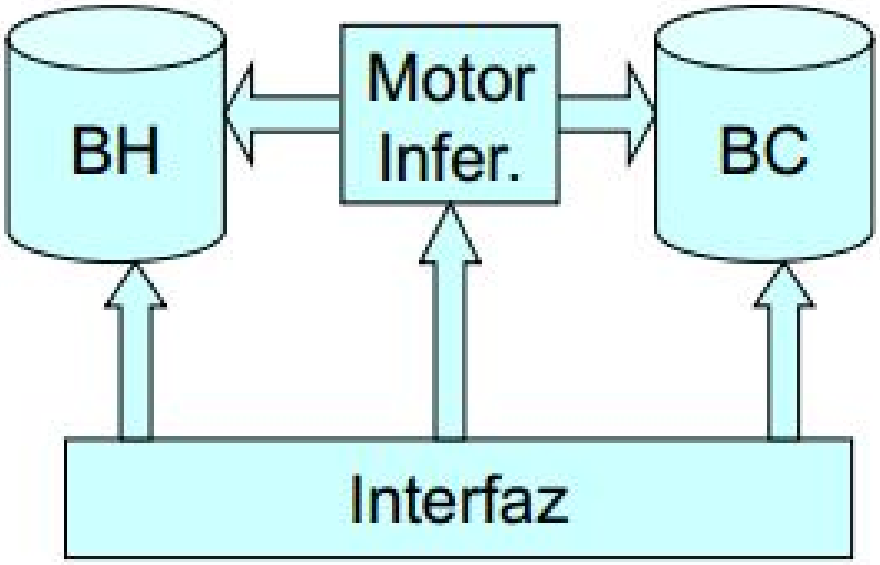
Práctica 2: sistema basado en reglas

Sistemas Inteligentes - 3º Ingeniería Informática - 2016/2017



Jorge Gallego Madrid

Grupo 1.2

[16-12-2016]

Equiparación-Conjunto conflicto **4**

Condición de parada **4**

Aplicación del SBR construido **5**

Situación 1: identificación de frutas. 5

BH-F1.txt 5

BH-F2.txt 6

Situación 2: detección de inundaciones. 7

Config-I.txt 7

BH-I1.txt 9

BH-I2.txt 9

BH-I3.txt 10

Bibliografía **11**

Sistema basado en reglas

Los sistemas basados en reglas (SBR) se inspiran en los sistemas de deducción en lógica proposicional o de primer orden. Utilizan la estructura de inferencia modus ponens para obtener conclusiones lógicas, interpretan la primera premisa como una regla de la forma IF condición THEN acción.

Los tres componentes básicos de un SBR son tres:

- Base de conocimientos (BC): contiene las reglas que codifican todo el conocimiento.
- Base de hechos (BH): contiene los hechos establecidos como verdaderos, tanto datos de entrada como conclusiones inferidas.
- Mecanismo de inferencias (MI): selecciona las reglas que se pueden aplicar y las ejecuta, con el objetivo de obtener alguna conclusión.

Las reglas operan sobre el espacio de trabajo de la BH, la condición de las mismas es referente al contenido de la BH, mientras que, si se verifica ésta, la acción puede modificar la BH.

La inferencia en un SBR la podemos clasificar en dos formas, dependiendo del razonamiento:

- Encadenamiento hacia delante: buscar el conjunto de metas que se verifican a partir de un conjunto de hechos. En este tipo, la inferencia progresa en la red de izquierda a derecha.
- Encadenamiento hacia atrás: determina si se verifica una cierta meta con los hechos disponibles. Aquí, la inferencia progresa en la red de derecha a izquierda.

En el desarrollo de esta práctica se va a desarrollar un motor de inferencia con encadenamiento hacia delante, que sigue este esquema genérico:

```
FUNCION MotorEncaminamientoHaciaDelante
    BH = hechosIniciales
    ConjuntoConflicto = ExtraeCualquierRegla(BC)
    MIENTRAS NoContenida(Meta, BH) Y NoVacio(ConjuntoConflicto) HACER
        ConjuntoConflicto = Equiparar(antecedente(BC), BH)
        SI NoVacio(ConjuntoConflicto) ENTONCES
            R = Resolver(ConjuntoConflicto)
            NuevosHechos = Aplicar(R, BH)
            Actualizar(BH, NuevosHechos)
```

```
        FIN
    FIN
    SI Contenida(Meta, BH) ENTONCES devolver "exito"
FIN
```

Básicamente, el motor ejecuta un bucle donde se busca un conjunto de reglas que sea posible aplicar para los datos almacenados en la base de hechos y las almacena en el llamado conjunto conflicto, de donde se selecciona una de ellas posteriormente para aplicarla. Una vez hecho esto se actualiza la base de hechos y se comprueban las condiciones de parada: que la meta haya sido alcanzada o que ya no se puedan aplicar más reglas a la base de hechos que tenemos.

Equiparación-Conjunto conflicto

Por definición, la equiparación en un SBR es la búsqueda del conjunto de reglas cuyas condiciones o acciones sean compatibles con los datos almacenados. El conjunto de reglas que se obtiene durante el proceso de equiparación se denomina conjunto conflicto.

En el motor de inferencia diseñado durante el desarrollo de esta práctica la equiparación la realizaremos comprobando cuáles de las reglas que poseemos en la base de conocimientos son aplicables en nuestro sistema, teniendo en cuenta los hechos que sabemos. Éstas reglas las iremos almacenando en el conjunto conflicto.

Para llevar a cabo esto, tendremos que recorrer la estructura donde guardemos las reglas. Entonces, para cada una de ellas y comprobando que no han sido usadas anteriormente, lo que haremos será revisar las condiciones que nos impone su antecedente, es decir, si en la base de hechos tenemos el conocimiento necesario (hechos y valores) como para poder aplicarla. Solo si la regla cumple todos estos requisitos será insertada en el conjunto conflicto que se usará para la iteración actual.

~~Es necesario añadir que una vez que una regla perteneciente al conjunto conflicto sea usada, ya no podremos reutilizarla.~~

Condición de parada

Nuestras condiciones de parada serán dos:

- Nos quedamos sin reglas en el conjunto conflicto: esta situación se dará cuando ya no podamos aplicar más reglas sobre los hechos que sabemos.
- Encontramos la solución: la condición más obvia, si hemos encontrado ya la solución que estamos buscando no es necesario continuar.

Aplicación del SBR construido

En este apartado se muestran los resultados de la aplicación del SBR desarrollado a los dos dominios proporcionados por el profesor. En el primero se muestran las salidas para las dos bases de hechos indicadas. En el segundo, se explica la creación de la configuración y de las dos bases de hechos requeridas. ~~tres bases de hechos (se proporciona una adicional para mostrar cómo se comportan las tres situaciones posibles que nos podemos encontrar).~~

~~Con el fin de evitar repetir la misma explicación de la salida en los cinco casos, a continuación se comenta la salida de forma general.~~

El formato de la salida se explica de manera genérica a continuación:

- En la consola se muestra el progreso de la ejecución, así como el dominio y si se ha encontrado una solución.
- Respecto al archivo solución generado:
    - En primer lugar se indica el dominio en el que estamos trabajando, el objetivo que queremos conseguir y la base de hechos de la que partimos.
    - En segundo lugar, se muestra la solución a la que ha llegado el algoritmo. En caso de no haber llegado a ninguna, se muestra "solución no encontrada".
    - Finalmente, se imprime la reconstrucción del camino que ha usado el SBR para llegar a la solución, mostrando las reglas usadas para ello.

Situación 1: identificación de frutas.

BH-F1.txt

Con esta base de hechos, llegamos a "Fruta = Cereza"

```
C:\Users\Jorge\Desktop\MEGASync\UMUGraph\SSII\Practicas\SSII\cmake-build-debug>Motor.exe BC-F.txt Config-F.txt BH-F1.txt
Leyendo los ficheros...
Dominio: IDENTIFICACION DE FRUTAS
Aplicando el algoritmo y buscando una solucion...
Solucion encontrada, detalles en: solucion_BH-F1.txt
```

solucion_BH-F1.txt

```
Buscando una solución en el dominio IDENTIFICACION DE FRUTAS
El objetivo es: Fruta
La base de hechos inicial (BH-F1.txt) es:
   Color = Rojo
   Diametro = 3
```

```
   Forma = Redonda
   NSemillas = 1
-----------------------------------------------

Solución encontrada: Fruta = Cereza

Reglas usadas para llegar a la solucion:
   R5: Si Forma = Redonda y Diametro < 10 Entonces ClaseFrutal = Arbol
   R6: Si NSemillas = 1 Entonces TipoSemilla = Hueso
   R13: Si ClaseFrutal = Arbol y Color = Rojo y TipoSemilla = Hueso
Entonces Fruta = Cereza
```

Como podemos ver, el motor ha conseguido llegar a la meta propuesta por medio de tres reglas, las nº 5, 6 y 13. Las dos primeras se han podido aplicar gracias al conocimiento que se tenía en la base de hechos inicial, mientras que la última se ha inferido a partir de los nuevos hechos. Con la regla nº 13 es con la que alcanzamos la meta.

BH-F2.txt

Con esta base de hechos, llegamos a "Fruta = Manzana".

```
C:\Users\Jorge\Desktop\MEGASync\UMUGraph\SSII\Practicas\SSII\cmake-build-debug>Motor.exe BC-F.txt Config-F.txt BH-F2.txt
Leyendo los ficheros...
Dominio: IDENTIFICACION DE FRUTAS
Aplicando el algoritmo y buscando una solucion...
Solucion encontrada, detalles en: solucion_BH-F2.txt
```

solucion_BH-F2.txt

```
Buscando una solución en el dominio IDENTIFICACION DE FRUTAS
El objetivo es: Fruta
La base de hechos inicial (BH-F2.txt) es:
   Color = Verde
   Diametro = 8
   Forma = Redonda
   NSemillas = 10
-----------------------------------------------

Solución encontrada: Fruta = Manzana

Reglas usadas para llegar a la solucion:
   R5: Si Forma = Redonda y Diametro < 10 Entonces ClaseFrutal = Arbol
   R7: Si NSemillas > 1 Entonces TipoSemilla = Multiple
   R16: Si ClaseFrutal = Arbol y Color = Verde y TipoSemilla = Multiple
Entonces Fruta = Manzana
```

De la misma forma que en el caso anterior, hemos conseguido llegar a la meta por

medio de las reglas nº 5, 7 y 16. Nuevamente, las dos primeras se han podido aplicar gracias a los hechos iniciales y la otra ha sido inferida por los nuevos hechos. Con la regla nº 16 es con la que conseguimos alcanzar la meta.

Situación 2: detección de inundaciones.

Config-I.txt

La realización del fichero de configuración para el dominio DETECCIÓN DE INUNDACIONES se puede dividir en tres partes:

- ATRIBUTOS: a partir de la base de conocimientos proporcionada por el profesor se buscan todos los atributos que aparecen y se incluyen en esta parte, así como los posibles valores que pueden tomar.
- OBJETIVO: debido a que el dominio es DETECCIÓN DE INUNDACIONES, podemos deducir que el objetivo será averiguar "Inundacion".
- PRIORIDADES-REGLAS: para asignar las prioridades a las reglas seguimos las siguientes pautas:
    - Las reglas que más prioridad tendrán serán aquellas mediante las cuales obtengamos una solución, es decir, aquellas cuyo consecuente es el atributo "Inundacion".
    - Dentro de las reglas que nos proporcionan una solución, si encontramos dos reglas que obtienen la misma conclusión y los antecedentes de una son los mismos que de la otra pero añadiendo uno más, ésta última deberá tener mayor prioridad. Esto es debido a que la segunda será capaz de llegar a la solución con una mayor certeza que la otra, al tener más información para obtener una conclusión.
    - El resto de reglas tendrán la misma prioridad con una excepción, las que nos permiten distinguir los dos tipos de reglas comentados en el punto anterior. Examinando la base de conocimientos podemos ver que el factor que distingue estos dos tipos es la inclusión del atributo "Lluvia", por lo que le damos más prioridad a las reglas que lo consiguen que al resto porque va a ser el que nos permita llegar a una solución más acertada.

```
ATRIBUTOS
9
Mes Nom
{Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio,Agosto,Septiembre,Octubre,Nov
```

```
iembre,Diciembre}
Estacion Nom {Seca,Humeda}
Precipitaciones Nom {Ninguna,Ligera,Fuertes}
Cambio Nom {Bajando,Ninguno,Subiendo}
Profundidad NU
Nivel Nom {Bajo,Normal,Alto}
Prediccion Nom {Soleado,Nuboso,Tormenta}
Lluvia Nom {Ninguna,Ligera,Fuerte}
Inundacion Nom {Si,No}
OBJETIVO
Inundacion
PRIORIDADES-REGLAS
32
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
5
5
5
7
7
7
7
7
10
10
10
10
10
```

BH-I1.txt

Esta primera base de hechos está diseñada para obtener “Inundacion = Si”.

```
4
Prediccion = Nuboso
Mes = Mayo
Profundidad = 7
Precipitaciones = Fuertes
```

```
C:\Users\Jorge\Desktop\MEGASync\UMUGraph\SSII\Practicas\SSII\cmake-build-debug>Motor.exe BC-I.txt Config-I.txt BH-I1.txt
Leyendo los ficheros...
Dominio: DETECCION DE INUNDACIONES
Aplicando el algoritmo y buscando una solucion...
Solucion encontrada, detalles en: solucion_BH-I1.txt
```

solucion_BH-I1.txt

```
Buscando una solución en el dominio DETECCION DE INUNDACIONES
El objetivo es: Inundacion
La base de hechos inicial (BH-I1.txt) es:
  Mes = Mayo
  Precipitaciones = Fuertes
  Prediccion = Nuboso
  Profundidad = 7
-----------------------------------------------

Solución encontrada: Inundacion = Si

Reglas usadas para llegar a la solucion:
  R21: Si Prediccion = Nuboso Entonces Lluvia = Ligera
  R16: Si Precipitaciones = Fuertes Entonces Cambio = Subiendo
  R19: Si Profundidad > 5 Entonces Nivel = Alto
  R31: Si Cambio = Subiendo y Nivel = Alto y Lluvia = Ligera Entonces
Inundacion = Si
```

Como se puede ver en el resultado de la ejecución, con esta base de hechos alcanzamos el objetivo Inundación. Esto se hace a partir de las reglas nº 21, 16, 19 y 31. Las tres primeras reglas pueden aplicarse partiendo de la base de hechos inicial, y gracias a ellas podemos llegar a la meta, mediante la regla nº 31.

~~BH-I2.txt~~

~~Esta base de hechos está diseñada para que la solución esté incluida en la base de hechos inicial.~~

```
1
Inundacion = Si
```

solucion_BH-I2.txt

```
Buscando una solución en el dominio DETECCION DE INUNDACIONES
El objetivo es: Inundacion
La base de hechos inicial (BH-I2.txt) es:
   Inundacion = Si
-----------------------------------------------

Solución encontrada: Inundacion = Si
```

BH-I3.txt

Esta base de hechos está diseñada para que no se encuentre una solución.

```
4
Mes = Enero
Precipitaciones = Fuertes
Profundidad = 10
```

```
C:\Users\Jorge\Desktop\MEGASync\UMUGraph\SSII\Practicas\SSII\cmake-build-debug>Motor.exe BC-I.txt Config-I.txt BH-I3.txt
Leyendo los ficheros...
Dominio: DETECCION DE INUNDACIONES
Aplicando el algoritmo y buscando una solucion...
Solucion no encontrada, detalles en: solucion_BH-I3.txt
```

solucion_BH-I3.txt

```
Buscando una solución en el dominio DETECCION DE INUNDACIONES
El objetivo es: Inundacion
La base de hechos inicial (BH-I3.txt) es:
   Mes = Enero
   Precipitaciones = Fuertes
   Profundidad = 10
-----------------------------------------------
Solución no encontrada
```

Con esta base de datos observamos que no es posible llegar a la meta, a pesar de que con esos datos iniciales sí que es posible aplicar algunas reglas, pero no las necesarias para llegar a una conclusión.

Bibliografía

[1] Imagen de portada. Fundamentos teóricos SBR. Diapositivas de la asignatura de Sistemas Inteligentes, 3º Ingeniería Informática en la UMU.

[2] Sistemas Inteligentes (2016-2017). Fundamentos teóricos SBR. Diapositivas de la asignatura de Sistemas Inteligentes, 3º Ingeniería Informática en la UMU. Usadas para responder a algunas cuestiones teóricas.

[3] std::ifstream [Online]. http://www.cplusplus.com/reference/fstream/ifstream/. Usado en la lectura de los ficheros.

[4] std::match_results [Online]. http://www.cplusplus.com/reference/regex/match_results/. Usado para las expresiones regulares que se usan para parsear las reglas.

[5] std::map [Online]. http://www.cplusplus.com/reference/map/map/. Estructura usada para almacenar los atributos, simula la creación de variables en tiempo de ejecución.

[6] std::list [Online]. http://www.cplusplus.com/reference/list/list/. Estructura usada para almacenar reglas y literales.

[7] std::set [Online].http://www.cplusplus.com/reference/set/set/. Estructura usada para el conjunto conflicto.

[8] <climits> (limits.h) [Online]. http://www.cplusplus.com/reference/climits/. Usado para calcular la regla con más prioridad.