Sistemas Inteligentes

Práctica 2

Pablo José Rocamora Zamora G3.2

Convocatoria de Junio

Índice general

Explicación breve y completa de la técnica Sistema Basado en Reglas (SBR)

Introducción

Los sistemas basados en reglas (SBR) se inspiran en los sistemas de deducción en lógica proposicional o de primer orden. Se les suele llamar Sistemas Expertos (SE) porque el conocimiento suele proceder de expertos humanos y los SBR permiten capturarlo bien.

- Utilizan la estructura de inferencia *modus ponens* para obtener conclusiones lógicas.
- Interpretan la primera premisa de un *modus ponens* como una regla de la forma **IF condición THEN acción**.

Componentes Básicos

- Base de conocimiento (BC): Contiene las reglas que codifican todo el conocimiento. Una regla consta de 2 partes: izquierda o antecedente y derecha o consecuente. Ejemplo: *IF antecedente THEN consecuente*.
- Base de Hechos (BH): Son los hechos establecidos como verdaderos; pueden ser tanto datos de entrada como conclusiones inferidas.
- Motor de Inferencias (MI): Selecciona las reglas que se pueden ejecutar y las ejecuta con el objetivo de obtener alguna conclusión.

Una diferencia importante entre la BC y la BH es que la BH almacena información puntual sobre un problema concreto, mientras que la BC almacena porciones de cocimiento sobre cómo resolver el problema genérico.

Inferencia en un SBR

Hay dos formas de razonar un MI:

- Encadenamiento hacia delante o dirigido por datos: se seleccionan las reglas cuyos antecedentes se verifican a partir del contenido de la BH. La particularidad de esta etapa es la equiparación; en ella, se seleccionan las reglas cuyos antecedentes se verifican a partir del contenido de la BH.
- Encadenamiento hacia atrás o dirigido por metas: se especifica una meta u objetivo y se trata de determinar si se verifica o no, teniendo en cuenta el contenido de la BH.

Ejemplo:

Teniendo estas dos reglas:

```
R1: Si A Entonces B
R2: Si B Entonces C
```

Si tengo A, a través de la *R1* podemos inferir B; y teniendo B, a través de *R2* podemos inferir C. Por tanto, teniendo inicialmente A, acabamos obteniendo: A, B y C.

Técnicas de Equiparación

La equiparación del antecedente de las reglas con el estado de la BH no siempre es trivial: puede no describir situaciones particulares sino generales. Otro problema es la necesidad de examinar todas las reglas en cada ciclo de inferencias, ya que puede ser poco eficiente si la BC es extensa. El tiempo de ejecución puede mejorar con indexado de reglas o el algoritmo *RETE*.

Técnicas de resolución de conflictos

Un método de resolución de conflictos selecciona, a partir del conjunto conflicto, la regla que hay que aplicar. Las principales técnicas de resolución son las siguientes:

- Según la BC (Criterios estáticos): Seleccionan las reglas ordenados por un criterio, como puede ser la prioridad de reglas.
- Según la BH (Criterios dinámicos): Reglas que usan elementos mas recientes de la BH.
- Según la ejecución (Criterios dinámicos): Usar reglas no utilizadas previamente.

Explicación clara de los elementos siguientes del motor de inferencia diseñado

Equiparación-Conjunto conflicto

La equiparación: El objetivo de la equiparación es seleccionar reglas válidas para la BH. Se realizará recorriendo regla a regla la BC y comprobando, para cada condición de la regla, si está esa misma condición en la BH; si, además, se cumplen todas las condiciones de la regla, entonces podremos obtener esa regla para ser usada posteriormente.

Conjunto conflicto: La equiparación nos retorna una lista de reglas válidas para usarse. Para averiguar qué regla tenemos que usar en cada caso, cogeremos siempre la primera (ya que la BC está ordenada primero por prioridad y, en caso de empate de prioridad, por número de regla); gracias a eso, la equiparación nos retorna la lista ordenada.

Condición de parada

Tenemos 2 condiciones de parada:

- Cuando en la BH aparezca el objetivo que buscamos, lo que significará que hemos resuelto el problema.
- Cuando equiparar no retorna ninguna regla o todas las reglas que retornan ya han sido usadas, lo que significará que no podemos continuar y, por tanto, no se puede resolver el problema por falta de información.

Aplicación del SBR construido a las siguientes situaciones

Situación 1: Identificación de Frutas.

Para la situación 1 tenemos la siguiente BC ya ordenada por criterio estático de orden de prioridad y, en caso de empate, por orden numérico ascendente

```
R2: IF Forma = Larga && Color = Amarillo THEN Fruta = Platano; Priority: 10 ; Use: False
R7: IF Forma = Larga && Color = Verde THEN Fruta = Platano; Priority: 10 ; Use: False
R8: IF ClaseFrutal = Emparrado && Color = Verde THEN Fruta = Sandia; Priority: 10 ; Use:
    ↪ False
R9: IF ClaseFrutal = Emparrado && Superficie = Lisa && Color = Amarillo THEN Fruta = Melon
    ↪ ; Priority: 10 ; Use: False
R10: IF ClaseFrutal = Emparrado && Superficie = Rugosa && Color = Tostado THEN Fruta =
    ↪ Cantalupo; Priority: 10 ; Use: False
R11: IF ClaseFrutal = Arbol && Color = Naranja && TipoSemilla = Hueso THEN Fruta =
    ↪ Albaricoque; Priority: 10 ; Use: False
R12: IF ClaseFrutal = Arbol && Color = Naranja && TipoSemilla = Multiple THEN Fruta =
    ↪ Naranja; Priority: 10 ; Use: False
R13: IF ClaseFrutal = Arbol && Color = Rojo && TipoSemilla = Hueso THEN Fruta = Cereza;
    ↪ Priority: 10 ; Use: False
R14: IF ClaseFrutal = Arbol && Color = Rojo && TipoSemilla = Multiple THEN Fruta = Manzana
    ↪ ; Priority: 10 ; Use: False
R15: IF ClaseFrutal = Arbol && Color = Amarillo && TipoSemilla = Multiple THEN Fruta =
    ↪ Manzana; Priority: 10 ; Use: False
R16: IF ClaseFrutal = Arbol && Color = Verde && TipoSemilla = Multiple THEN Fruta =
    ↪ Manzana; Priority: 10 ; Use: False
R17: IF ClaseFrutal = Arbol && Color = Naranja && TipoSemilla = Hueso THEN Fruta =
    ↪ Melocoton; Priority: 10 ; Use: False
R18: IF ClaseFrutal = Arbol && Color = Morado && TipoSemilla = Hueso THEN Fruta = Ciruela;
    ↪  Priority: 10 ; Use: False
R1: IF NSemillas > 1 THEN TipoSemilla = Multiple; Priority: 0 ; Use: False
R3: IF Forma = Redonda && Diametro >= 10 THEN ClaseFrutal = Emparrado; Priority: 0 ; Use:
    ↪ False
R4: IF Forma = Ensanchada && Diametro >= 10 THEN ClaseFrutal = Emparrado; Priority: 0 ;
    ↪ Use: False
R5: IF Forma = Redonda && Diametro < 10 THEN ClaseFrutal = Arbol; Priority: 0 ; Use: False
R6: IF NSemillas = 1 THEN TipoSemilla = Hueso; Priority: 0 ; Use: False
```

Nuestro objetivo será encontrar el atributo *Fruta*; por eso, todas las reglas con las que se obtiene una *fruta* tienen prioridad 10, y las reglas para obtener atributos intermedios tienen prioridad 0. La condición de parada es Fruta en BH o Conjunto Conflicto vacío. Usaremos Encadenamiento hacia delante para llegar a la solución.

BH-F1

Inicialmente tenemos la siguiente BH:

```
Diametro = 3 && Forma = Redonda && NSemillas = 1 && Color = Rojo
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: ClaseFrutal = Arbol)

- Conjunto Conflicto = {R5, R6}
- Resolver Conflicto: R5
- BH = BH + {R5} // Aplicar R5 en BH
- Marcada = {R5}
- Conjunto Conflicto = {~~R5~~, R6} //Marco R5 como usada en BC

Iteración 2 (Obtendré: TipoSemilla = Hueso)

- Resolver Conflicto: R6

- BH = BH + {R6} // Aplicar R6 en BH
- Marcada = {R5, R6}
- Conjunto Conflicto = {~~R5~~, ~~R6~~, R13} //Marco R6 como usada en BC

Iteración 3 (Obtendré: Fruta = Cereza)

- Resolver Conflicto: R13
- BH = BH + {R13} // Aplicar R13 en BH
- Marcada = {R5, R6, R13}
- Conjunto Conflicto = {~~R5~~, ~~R6~~, ~~R13~~} //Marco R13 como usada en BC
- Condición de fin: Fruta en BH (FIN)

Estado final de la BH:

```
Diametro = 3 && Forma = Redonda && NSemillas = 1 && Color = Rojo && ClaseFrutal = Arbol &&
    ↪  TipoSemilla = Hueso && Fruta = Cereza
```

El orden de las reglas usadas sería:

```
R5: IF Forma = Redonda && Diametro < 10 THEN ClaseFrutal = Arbol; Priority: 0 ; Use: True
R6: IF NSemillas = 1 THEN TipoSemilla = Hueso; Priority: 0 ; Use: True
R13: IF ClaseFrutal = Arbol && Color = Rojo && TipoSemilla = Hueso THEN Fruta = Cereza;
    ↪ Priority: 10 ; Use: True
```

El resultado objetivo es: **Fruta = Cereza**

BH-F2

Inicialmente tenemos la siguiente BH:

```
Diametro = 8 && Forma = Redonda && NSemillas = 10 && Color = Verde
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: TipoSemilla = Multiple)

- Conjunto Conflicto = {R1, R5}
- Resolver Conflicto: R1
- BH = BH + {R1} // Aplicar R1 en BH
- Marcada = {R1}
- Conjunto Conflicto = {~~R1~~, R5} //Marco R1 como usada en BC

Iteración 2 (Obtendré: ClaseFrutal = Arbol)

- Resolver Conflicto: R5
- BH = BH + {R5} // Aplicar R5 en BH
- Marcada = {R1, R5}
- Conjunto Conflicto = {~~R1~~, ~~R5~~, R16} //Marco R5 como usada en BC

Iteración 3 (Obtendré: Fruta = Manzana)

- Resolver Conflicto: R16
- BH = BH + {R16} // Aplicar R16 en BH
- Marcada = {R1, R5, R16}
- Conjunto Conflicto = {~~R1~~, ~~R5~~, ~~R16~~} //Marco R16 como usada en BC
- Condición de fin: Fruta en BH (FIN)

Estado final de la BH:

```
Diametro = 8 && Forma = Redonda && NSemillas = 10 && Color = Verde && TipoSemilla =
    ↪ Multiple && ClaseFrutal = Arbol && Fruta = Manzana
```

El orden de las reglas usadas sería:

```
R1: IF NSemillas > 1 THEN TipoSemilla = Multiple; Priority: 0 ; Use: True
R5: IF Forma = Redonda && Diametro < 10 THEN ClaseFrutal = Arbol; Priority: 0 ; Use: True
R16: IF ClaseFrutal = Arbol && Color = Verde && TipoSemilla = Multiple THEN Fruta =
    ↪ Manzana; Priority: 10 ; Use: True
```

El resultado objetivo es: **Fruta = Manzana**

BH-F3

Inicialmente tenemos la siguiente BH:

```
Forma = Redonda && NSemillas = 2 && Diametro = 11 && Color = Verde
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: TipoSemilla = Multiple)

- Conjunto Conflicto = {R1, R3}
- Resolver Conflicto: R1
- BH = BH + {R1} // Aplicar R1 en BH
- Marcada = {R1}
- Conjunto Conflicto = {~~R1~~, R3} //Marco R1 como usada en BC

Iteración 2 (Obtendré: ClaseFrutal = Emparrado)

- Resolver Conflicto: R3
- BH = BH + {R3} // Aplicar R3 en BH
- Marcada = {R1, R3}
- Conjunto Conflicto = {~~R1~~, ~~R3~~, R8} //Marco R3 como usada en BC

Iteración 3 (Obtendré: Fruta = Sandia)

- Resolver Conflicto: R8
- BH = BH + {R8} // Aplicar R8 en BH
- Marcada = {R1, R3, R8}
- Conjunto Conflicto = {~~R1~~, ~~R3~~, ~~R8~~} //Marco R8 como usada en BC
- Condición de fin: Fruta en BH (FIN)

Estado final de la BH:

```
Forma = Redonda && NSemillas = 2 && Diametro = 11 && Color = Verde && TipoSemilla =
    ↪ Multiple && ClaseFrutal = Emparrado && Fruta = Sandia
```

El orden de las reglas usadas sería:

```
R1: IF NSemillas > 1 THEN TipoSemilla = Multiple; Priority: 0 ; Use: True
R3: IF Forma = Redonda && Diametro >= 10 THEN ClaseFrutal = Emparrado; Priority: 0 ; Use:
    ↪ True
R8: IF ClaseFrutal = Emparrado && Color = Verde THEN Fruta = Sandia; Priority: 10 ; Use:
    ↪ True
```

El resultado objetivo es: **Fruta = Sandia**

BH-F4

Inicialmente tenemos la siguiente BH:

```
ClaseFrutal = Arbol && Color = Naranja && Forma = Redonda && NSemillas = 1 && Diametro = 6
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: ClaseFrutal = Arbol)

- Conjunto Conflicto = {R5, R6}
- Resolver Conflicto: R5
- BH = BH + {R5} // Aplicar R5 en BH
- Marcada = {R5}
- Conjunto Conflicto = {~~R5~~, R6} //Marco R5 como usada en BC

Iteración 2 (Obtendré: TipoSemilla = Hueso)

- Resolver Conflicto: R6
- BH = BH + {R6} // Aplicar R6 en BH
- Marcada = {R5, R6}
- Conjunto Conflicto = {~~R5~~, ~~R6~~, R11, R17} //Marco R6 como usada en BC

Iteración 3 (Obtendré: Fruta = Albaricoque)

- Resolver Conflicto: R11
- BH = BH + {R11} // Aplicar R11 en BH
- Marcada = {R5, R6, R11}
- Conjunto Conflicto = {~~R5~~, ~~R6~~, ~~R11~~, R17} //Marco R11 como usada en BC
- Condición de fin: Fruta en BH (FIN)

Estado final de la BH:

```
ClaseFrutal = Arbol && Color = Naranja && Forma = Redonda && NSemillas = 1 && Diametro = 6
    ↪  && ClaseFrutal = Arbol && TipoSemilla = Hueso && Fruta = Albaricoque
```

El orden de las reglas usadas sería:

```
R5: IF Forma = Redonda && Diametro < 10 THEN ClaseFrutal = Arbol; Priority: 0 ; Use: True
R6: IF NSemillas = 1 THEN TipoSemilla = Hueso; Priority: 0 ; Use: True
R11: IF ClaseFrutal = Arbol && Color = Naranja && TipoSemilla = Hueso THEN Fruta =
    ↪ Albaricoque; Priority: 10 ; Use: True
```

El resultado objetivo es: **Fruta = Albaricoque**

> En esta situación cabe destacar que la regla 17 indicaba que el atributo *Fruta = Melocoton*, por lo que, si hubiese tenido mayor prioridad que la regla 11, el resultado habría sido *Fruta = Melocoton*

Situación 2: Detección de Inundaciones.

Para la situación 2 tenemos la siguiente BC ya ordenada por criterio estático de orden de prioridad y, en caso de empate, por orden numérico ascendente

```
R23: IF Nivel = Bajo THEN Inundacion = No; Priority: 10 ; Use: False
R24: IF Cambio = Ninguno && Nivel = Normal THEN Inundacion = No; Priority: 10 ; Use: False
R25: IF Cambio = Ninguno && Nivel = Bajo THEN Inundacion = No; Priority: 10 ; Use: False
R26: IF Cambio = Subiendo && Nivel = Normal THEN Inundacion = No; Priority: 10 ; Use:
    ↪ False
R27: IF Cambio = Subiendo && Nivel = Bajo THEN Inundacion = No; Priority: 10 ; Use: False
R28: IF Cambio = Subiendo && Nivel = Normal && Lluvia = Fuerte THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: False
R29: IF Cambio = Subiendo && Nivel = Normal && Lluvia = Ligera THEN Inundacion = No;
    ↪ Priority: 10 ; Use: False
R30: IF Cambio = Subiendo && Nivel = Alto && Lluvia = Ninguna THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: False
R31: IF Cambio = Subiendo && Nivel = Alto && Lluvia = Ligera THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: False
R32: IF Cambio = Subiendo && Nivel = Alto && Lluvia = Fuerte THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: False
R1: IF Mes = Junio THEN Estacion = Seca; Priority: 0 ; Use: False
R2: IF Mes = Julio THEN Estacion = Seca; Priority: 0 ; Use: False
R3: IF Mes = Agosto THEN Estacion = Seca; Priority: 0 ; Use: False
R4: IF Mes = Septiembre THEN Estacion = Humeda; Priority: 0 ; Use: False
R5: IF Mes = Octubre THEN Estacion = Humeda; Priority: 0 ; Use: False
R6: IF Mes = Noviembre THEN Estacion = Humeda; Priority: 0 ; Use: False
R7: IF Mes = Diciembre THEN Estacion = Humeda; Priority: 0 ; Use: False
R8: IF Mes = Enero THEN Estacion = Humeda; Priority: 0 ; Use: False
R9: IF Mes = Febrero THEN Estacion = Humeda; Priority: 0 ; Use: False
R10: IF Mes = Marzo THEN Estacion = Humeda; Priority: 0 ; Use: False
R11: IF Mes = Abril THEN Estacion = Humeda; Priority: 0 ; Use: False
R12: IF Mes = Mayo THEN Estacion = Humeda; Priority: 0 ; Use: False
R13: IF Precipitaciones = Ninguna && Estacion = Seca THEN Cambio = Bajando; Priority: 0 ;
    ↪ Use: False
R14: IF Precipitaciones = Ninguna && Estacion = Humeda THEN Cambio = Ninguno; Priority: 0
    ↪ ; Use: False
R15: IF Precipitaciones = Ligera THEN Cambio = Ninguno; Priority: 0 ; Use: False
R16: IF Precipitaciones = Fuertes THEN Cambio = Subiendo; Priority: 0 ; Use: False
R17: IF Profundidad < 3 THEN Nivel = Bajo; Priority: 0 ; Use: False
R18: IF Profundidad >= 3 && Profundidad <= 5 THEN Nivel = Normal; Priority: 0 ; Use: False
R19: IF Profundidad > 5 THEN Nivel = Alto; Priority: 0 ; Use: False
R20: IF Prediccion = Soleado THEN Lluvia = Ninguna; Priority: 0 ; Use: False
R21: IF Prediccion = Nuboso THEN Lluvia = Ligera; Priority: 0 ; Use: False
R22: IF Prediccion = Tormenta THEN Lluvia = Fuerte; Priority: 0 ; Use: False
```

Para crear el fichero de configuración he tenido que observar inicialmente la cantidad de atributos que había, si eran numéricos o nominales y, en caso de ser nominales, los posibles valores que había en la BC. Posteriormente, he tenido que especificar cuál es el atributo objetivo. Ese dato lo he sacado del dominio de la aplicación, que se llama *Detección de inundaciones* y, por último, había que asignar prioridades a las reglas; como nuestro objetivo es encontrar el atributo *Inundacion*, todas las reglas con las que se obtiene *Inundacion* tienen prioridad 10 y las reglas para obtener atributos intermedios tienen prioridad 0. La condición de parada es *Inundacion* en BH o Conjunto Conflicto vacío. Usaremos Encadenamiento hacia delante para llegar a la solución.

Para crear las siguientes 4 BH he usado encadenamiento hacia atrás, especificaba la meta objetivo e iba calculando los consecuentes necesarios para llegar a ese resultado; esos consecuentes los guardaba en la BH.

BH-I1

Inicialmente tenemos la siguiente BH:

```
Mes = Septiembre && Prediccion = Tormenta && Precipitaciones = Fuertes && Profundidad = 10
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: Estacion = Humeda)

- Conjunto Conflicto = {R4, R16, R19, R22}
- Resolver Conflicto: R4
- BH = BH + {R4} // Aplicar R4 en BH
- Marcada = {R4}
- Conjunto Conflicto = {~~R4~~, R16, R19, R22} //Marco R4 como usada en BC

Iteración 2 (Obtendré: Cambio = Subiendo)

- Resolver Conflicto: R16
- BH = BH + {R16} // Aplicar R16 en BH
- Marcada = {R4, R16}
- Conjunto Conflicto = {~~R4~~, ~~R16~~, R19, R22} //Marco R16 como usada en BC

Iteración 3 (Obtendré: Nivel = Alto)

- Resolver Conflicto: R19
- BH = BH + {R19} // Aplicar R19 en BH
- Marcada = {R4, R16, R19}
- Conjunto Conflicto = {~~R4~~, ~~R16~~, ~~R19~~, R22} //Marco R19 como usada en BC

Iteración 4 (Obtendré: Lluvia = Fuerte)

- Resolver Conflicto: R22
- BH = BH + {R22} // Aplicar R22 en BH
- Marcada = {R4, R16, R19, R22}
- Conjunto Conflicto = {~~R4~~, ~~R16~~, ~~R19~~, ~~R22~~, R32} //Marco R22 como usada en BC

Iteración 5 (Obtendré: Inundacion = Si)

- Resolver Conflicto: R32
- BH = BH + {R32} // Aplicar R32 en BH
- Marcada = {R4, R16, R19, R22, R32}
- Conjunto Conflicto = {~~R4~~, ~~R16~~, ~~R19~~, ~~R22~~, ~~R32~~} //Marco R32 como usada en BC
- Condición de fin: Inundacion en BH (FIN)

Estado final de la BH:

```
Mes = Septiembre && Prediccion = Tormenta && Precipitaciones = Fuertes && Profundidad = 10
    ↪  && Estacion = Humeda && Cambio = Subiendo && Nivel = Alto && Lluvia = Fuerte &&
    ↪ Inundacion = Si
```

El orden de las reglas usadas sería:

```
R4: IF Mes = Septiembre THEN Estacion = Humeda; Priority: 0 ; Use: True
R16: IF Precipitaciones = Fuertes THEN Cambio = Subiendo; Priority: 0 ; Use: True
R19: IF Profundidad > 5 THEN Nivel = Alto; Priority: 0 ; Use: True
R22: IF Prediccion = Tormenta THEN Lluvia = Fuerte; Priority: 0 ; Use: True
R32: IF Cambio = Subiendo && Nivel = Alto && Lluvia = Fuerte THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: True
```

El resultado objetivo es: **Inundacion = Si**

BH-I2

Inicialmente tenemos la siguiente BH:

```
Precipitaciones = Fuertes && Profundidad = 4 && Prediccion = Nuboso
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: Cambio = Subiendo)

- Conjunto Conflicto = {R16, R18, R21}
- Resolver Conflicto: R16
- BH = BH + {R16} // Aplicar R16 en BH
- Marcada = {R16}
- Conjunto Conflicto = {~~R16~~, R18, R21} //Marco R16 como usada en BC

Iteración 2 (Obtendré: Nivel = Normal)

- Resolver Conflicto: R18
- BH = BH + {R18} // Aplicar R18 en BH
- Marcada = {R16, R18}
- Conjunto Conflicto = {~~R16~~, ~~R18~~, R26, R21} //Marco R18 como usada en BC

  Al resolver R18 obtengo la precondición que le faltaba a la regla R26 para poder resolverla; al tener mayor prioridad que R21, R26 se ejecutará antes, dando la solución del problema.

Iteración 3 (Obtendré: Inundacion = No)

- Resolver Conflicto: R26
- BH = BH + {R26} // Aplicar R26 en BH
- Marcada = {R16, R18, R26}
- Conjunto Conflicto = {~~R16~~, ~~R18~~, ~~R26~~, R21} //Marco R26 como usada en BC
- Condición de fin: Inundacion en BH (FIN)

Estado final de la BH:

```
Precipitaciones = Fuertes && Profundidad = 4 && Prediccion = Nuboso && Cambio = Subiendo
    ↪ && Nivel = Normal && Inundacion = No
```

El orden de las reglas usadas sería:

```
R16: IF Precipitaciones = Fuertes THEN Cambio = Subiendo; Priority: 0 ; Use: True
R18: IF Profundidad >= 3 && Profundidad <= 5 THEN Nivel = Normal; Priority: 0 ; Use: True
R26: IF Cambio = Subiendo && Nivel = Normal THEN Inundacion = No; Priority: 10 ; Use: True
```

El resultado objetivo es: **Inundacion = No**

BH-I3

Inicialmente tenemos la siguiente BH:

```
Precipitaciones = Fuertes && Profundidad = 12 && Prediccion = Soleado
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: Cambio = Subiendo)

- Conjunto Conflicto = {R16, R19, R20}
- Resolver Conflicto: R16
- BH = BH + {R16} // Aplicar R16 en BH
- Marcada = {R16}
- Conjunto Conflicto = {~~R16~~, R19, R20} //Marco R16 como usada en BC

Iteración 2 (Obtendré: Nivel = Alto)

- Resolver Conflicto: R19
- BH = BH + {R19} // Aplicar R19 en BH
- Marcada = {R19}
- Conjunto Conflicto = {~~R16~~, ~~R19~~, R20} //Marco R19 como usada en BC

Iteración 3 (Obtendré: Lluvia = Ninguna)

- Resolver Conflicto: R20
- BH = BH + {R20} // Aplicar R20 en BH
- Marcada = {R20}
- Conjunto Conflicto = {~~R16~~, ~~R19~~, ~~R20~~, R30} //Marco R20 como usada en BC

Iteración 3 (Obtendré: Inundacion = Si)

- Resolver Conflicto: R30
- BH = BH + {R30} // Aplicar R30 en BH
- Marcada = {R30}
- Conjunto Conflicto = {~~R16~~, ~~R19~~, ~~R20~~, ~~R30~~} //Marco R30 como usada en BC
- Condición de fin: Inundacion en BH (FIN)

Estado final de la BH:

```
Precipitaciones = Fuertes && Profundidad = 12 && Prediccion = Soleado && Cambio = Subiendo
    ↪  && Nivel = Alto && Lluvia = Ninguna && Inundacion = Si
```

El orden de las reglas usadas sería:

```
R16: IF Precipitaciones = Fuertes THEN Cambio = Subiendo; Priority: 0 ; Use: True
R19: IF Profundidad > 5 THEN Nivel = Alto; Priority: 0 ; Use: True
R20: IF Prediccion = Soleado THEN Lluvia = Ninguna; Priority: 0 ; Use: True
R30: IF Cambio = Subiendo && Nivel = Alto && Lluvia = Ninguna THEN Inundacion = Si;
    ↪ Priority: 10 ; Use: True
```

El resultado objetivo es: **Inundacion = Si**

BH-I4

Inicialmente tenemos la siguiente BH:

```
Precipitaciones = Ninguna && Estacion = Humeda
```

Las iteraciones del motor de inferencia serán:

Iteración 1 (Obtendré: Cambio = Ninguno)

- Conjunto Conflicto = {R14}
- Resolver Conflicto: R14
- BH = BH + {R14} // Aplicar R14 en BH
- Marcada = {R14}
- Conjunto Conflicto = {~~R14~~} //Marco R14 como usada en BC
- Condición de fin: Conjunto Conflicto esta vacío (FIN)

Estado final de la BH:

```
Precipitaciones = Ninguna && Estacion = Humeda && Cambio = Ninguno
```

El orden de las reglas usadas sería:

```
R14: IF Precipitaciones = Ninguna && Estacion = Humeda THEN Cambio = Ninguno; Priority: 0
    ↪ ; Use: True
```

El resultado objetivo **no ha podido ser encontrado**.

> Como podemos observar en esta BH no se han proporcionado suficientes datos para el SBR y, por lo tanto, no ha sido capaz de hallar una solución a ese problema.

Bibliografía

- [1] : Referencia general de C++, usado para repaso de multitud de referencias sobre el uso de C++.
- [2] : Método para convertir un string en un int, en Linux no es necesario pero parece que en Windows con MinGw si por un bug que tiene.
- [3] : Usado para partir un string según un delimitador que puede ser una expresión regular, lo he usado para parsear las condiciones de las reglas.
- [4] : Lo he usado para ver la mejor forma de usar los operadores de las condiciones que están guardados como string.
- [5] : Referencia usada para ver cual es la forma correcta de hacer un return de un struct cuando nunca se debería llegar a esa situación y si se llega es por un error, buscaba un sinónimo del return null de Java.
- [6] : Referencia usada para ver como usar expresiones regulares en C++ de forma correcta, lo he usado para parsear las reglas.
- [7] : Referencia usada para ver la forma de ordenar un vector por una serie de criterios estáticos, usado para ordenar el vector de reglas por prioridades.

Referencias:

[1] «Referencia de C++» [Online]. Disponible en: http://es.cppreference.com/w/cpp

[2] boredcircuits, «GCC 4.9.2 stoi not a member» [Online]. Disponible en: https://www.reddit.com/r/cpp_questions/comments/2s7y35/gcc_492_stoi_not_a_member/

[3] bayusetiaji, «Split a String» [Online]. Disponible en: http://www.cplusplus.com/articles/2wA0RXSz/

[4] Unglued, «Convert string to operator» [Online]. Disponible en: https://stackoverflow.com/

questions/24716453/convert-string-to-operator

[5] tenfour, «C++ return null struct from function» [Online]. Disponible en: https://stackoverflow.com/questions/22972265/c-return-null-struct-from-function?noredirect=1&lq=1

[6] Praetorian, «Regex grouping matches with C++ 11 regex library» [Online]. Disponible en: https://stackoverflow.com/questions/29321249/regex-grouping-matches-with-c-11-regex-library

[7] Alan, «Sorting a vector of custom objects» [Online]. Disponible en: https://stackoverflow.com/questions/1380463/sorting-a-vector-of-custom-objects