Sistemas Inteligentes

Practica 2

Pablo José Rocamora Zamora G3.2

Convocatoria de Junio

Índice general

acercándonos a la solución final [1].

acercándonos a la solución final [2].

acercándonos a la solución final [3].

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 se 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 como 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, dado el contenido de la BH. La particularidad de esta etapa es la equiparación, donde se seleccionan las reglas cuyos antecedentes se verifican, dado el contenido de la BH.
- Encadenamiento hacia atrás o dirigido por metas, se especifica una meta objetivo y se trata de determinar su la meta se verifica o no, teniendo en cuenta el contenido de la BH.

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, 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 a aplicar. Las principales técnicas de resolución son las siguientes:

- Según la BC (Criterios estáticos): Seleccionan las reglas ordenados por un criterio, como puede ser 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 (dado que el diseño es anterior a la implementación, no se debe hacer mención a aspectos de código):

Equiparación-Conjunto conflicto

La equiparación: El objetivo de la equiparación es seleccionar reglas validas para la BH, se realizara recorriendo regla a regla la BC y comprobando para cada condición de la regla si esta esa misma condición en la BH, si esta y 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 validas para usarse, para resolver que regla tenemos que usar en cada caso cogeremos siempre la primera, ya que la BC esta ordenada primero por prioridad y en caso de empate de prioridad por numero de regla, gracias a eso la equiparación nos retorna la lista ordenada y resolver el conjunto conflicto es una tarea sencilla.

Condición de parada

Tenemos 2 condiciones de parada:

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

Aplicación del SBR construido a las siguientes situaciones.

Incluir y explicar el razonamiento seguido en la resolución de cada base de hechos (fichero Salida1.txt que se indica en el apartado e)) y la solución obtenida y que proporcionaremos al usuario del SBR (fichero Salida2.txt que se indica en el apartado e)). Además, para la Situación 2, explicar claramente todas las decisiones tomadas para la definición del fichero de configuración.

Situación 1: Identificación de Frutas – Se proporcionan (recursos del Aula Virtual) la BC-F, Config-F, y cuatro bases de hechos (BH-F1, BH-F2, BH-F3 y BH-F4).

Para la situacion 1 tenemos la siguiente BC ya ordenada por criterio estatico de orden de prioridad y en caso de empate por orden numerico 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 sera encontrar el atributo Fruta, por eso mismo todas las reglas con las que se obtiene una fruta tienen prioridad 10 y las reglas para obtener atributos intermedios tienen prioridad 0. Por tanto la condicion de parada es Fruta en BH. Usaremos Encadenamiento hacia delante

BH-F1

Inicialmente tenemos la siguiente BH

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

Iteracion 1

Conjunto Conflicto = {R5, R6}

Resolver Conflicto: R5

BH = BH + {R5} // Aplicar R5 eb BH

Marcada = {R5}

Conjunto Conflicto = {~~R5~~, R6}

BH-F2

BH-F3

BH-F4

Situación 2: Detección de Inundaciones – Se proporciona BC-I (recursos del Aula Virtual). Para la aplicación, deben definirse el fichero de configuración y cuatro bases de hechos.

BH-I1

BH-I2

BH-I3

BH-I4

Tanto las BH como las BC proporcionadas no podrán ser modificadas.

Bibliografía

La referencia:

- 1: Usado para convertir un string en un int, ya que la funcion stoi de c++ parece que no esta implementada correctamente en MinGW
- : Usado para ver como partir un string a partir de un delimitador especifico
- : Usado para convertir un string en un operador y realizar su respectiva operacion
- : Usado para ver la forma mas correcta de hacer un return null de un struct cuando nunca deberia de ejecutarse, en c++ lo correcto seria enviar una excepcion

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

[2] 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

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