ALGUNOS ALGORITMOS RÁPIDOS PARA EL PROBLEMA DEL EMPAREJAMIENTO DE PATRONES

F. J. Vigo Bustos
M. Teresa López Bonal

Francisco J. Vigo Bustos y M. Teresa López Bonal están en el Dpto. de Informática. EUPA. Universidad de Castilla-La Mancha.

RESUMEN

El algoritmo de emparejamiento RETE es un método para comparar un conjunto de patrones o reglas con un conjunto de objetos y así determinar todos los posibles emparejamientos. ASIA es una ampliación del algoritmo RETE llevada a cabo para poder trabajar con datos captados de dispositivos externos. En este artículo se realizará una visión general de ambos algoritmos y se desarrollan algunos ejemplos para su mejor comprensión.

1. INTRODUCCIÓN

UN sistema experto ha de ofrecer una respuesta dentro de unos límites razonables de tiempo y esta exigencia se hace especialmente necesaria cuando el sistema experto interactúa con el entorno en el que se encuentra inmerso. Por tanto, es obvia la necesidad de algoritmos que lleven a cabo su labor de la forma más rápida posible.

Una de las tareas principales que ha de llevar a cabo un intérprete de un sistema de producción es el emparejamiento o «matching» que consiste en comparar el antecedente de las reglas (LHS) con los elementos de la memoria de trabajo (EMT) para así calcular el conjunto de reglas que pueden ser activadas.

Una forma posible de realizar la tarea descrita en el párrafo anterior sería comparar todos los antecedentes con todos los EMT, sin embargo, ésta no sería una buena solución en el caso de tener un número elevado de reglas o de EMT por lo que se hace necesario encontrar soluciones alternativas.

2. EL ALGORITMO RETE

RETE es un algoritmo que reduce sensiblemente el tiempo necesario para calcular las reglas que pueden ser activadas evitando iterar sobre la memoria de trabajo y sobre la memoria de producción.

2.1. Tratamiento de la memoria de trabajo

El intérprete simple, como ya se ha descrito, iterará sobre la memoria de trabajo comparando el patrón en cuestión con todos los elementos de la memoria de trabajo. RETE va a evitar esta iteración asociando a cada patrón una lista que contendrá todos aquellos elementos con los que empareja; esta lista será actualizada cuando se produzca algún cambio en la memoria de trabajo los cuales van a poder ser descritos mediante tokens. Un token será un par ordenado de la forma:

<etiqueta, elemento de datos>

en la que la etiqueta nos indicará si el elemento es añadido (+) o suprimido (–).

Ej.:

```
<- (Expresión #Nombre Expr41 #Arg1 Y #Op + #Arg2 Y) >
```

2.2. Tratamiento de la memoria de producción

El algoritmo RETE además de no iterar sobre la memoria de trabajo tampoco itera sobre la memoria de producción. Para ello, el algoritmo va a generar un grafo que represente el conjunto de reglas que será usado para determinar si la LHS de las reglas se verifican.

2.2.1. El proceso de generación del grafo o compilado de patrones

En un elemento de la memoria de trabajo vamos a poder distinguir dos tipos de características:

1. Característica intra-elementos: sólo se tienen en cuenta elementos de la memoria de trabajo. Si consideramos el siguiente patrón,

   ```
   (Expresión #Nombre <N> #Arg1 0 #Op + #Arg2 <X>)
   ```

   el emparejador de patrones intentará encontrar elementos de la memoria de trabajo que posean las siguientes características intra-elementos:

- La clase de elemento ha de ser expresión.
- El valor del atributo Arg1 ha de ser el número 0.
- El valor del atributo Op ha de ser el átomo +.

2. Características inter-elementos: este tipo de características se dan cuando una variable aparece en más de un patrón.
   Si consideramos la LHS siguiente

```
(P Suma0x
   ( Objetivo #Tipo Simplificar #Objeto <N> )
   ( Expresión #Nombre<N> #Arg1 0 #Op + #Arg2 <X> )
---> ...)
```

   podemos extraer la siguiente característica intra-elementos:
   - El valor del atributo Objeto del objetivo ha de ser igual al valor del atributo Nombre de expresión.

Veamos a continuación los pasos de que constaría el proceso de compilado:

1. Determinar la característica intra-elementos que cada patrón requiere y construir la secuencia lineal de nodos correspondientes. Cada nodo, denominado nodo constante, verificará si una determinada característica se cumple.
2. Construcción de nodos para verificar las características inter-elementos (nodos de dos entradas). En un nodo con dos entradas confluirán las secuencias lineales de dos patrones, por lo que será necesario un nuevo tipo de nodo denominado nodo de memoria para almacenar los tokens que lleguen al nodo de dos entradas.
3. Construcción de los nodos terminales que representan a la producción.

Supongamos que tenemos el siguiente conjunto de reglas

```
Regla R1:

   If
      (C1 #A1 1 #A2 <x>)
      (C2 #A1 <x> #A2 25)
      (C3 #A1 <x>)
   Then
      ...

Regla R2:

   If
      (C2 #A1 <y> A2 25)
      (C4 #A1 <y>)
   Then
      ...
```

Si aplicamos el algoritmo que se ha descrito previamente el grafo resultante sería el que aparece en la figura 1.

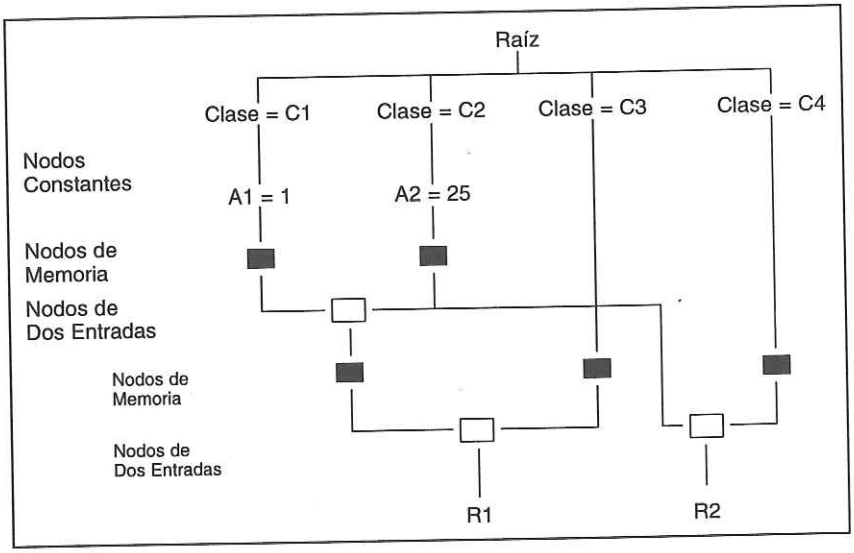

FIGURA 1

2.2.2. *Un ejemplo*

Veamos a continuación qué ocurre en el grafo representado en la figura anterior cuando se introducen en la memoria de trabajo los siguientes elementos

```
(C2 #A1 <10> #A2 25)
(C4 #A1 <10>)
```

En primer lugar, se crea el token <+(C2#A1 <10> #A2 25)> y se envía a la raíz de la red. Este nodo envía el token a todos sus sucesores siendo aceptado tan sólo por el segundo sucesor, el cual a su vez, envía el token a su sucesor siendo de nuevo aceptado por este último. Ahora el token es enviado al nodo de dos entradas y puesto que no ha llegado ningún otro token, queda almacenado ya que no es posible realizar ninguna comparación.

A continuación se pasaría a procesar el token <+(C4 #A1 <10>)>. Ahora sí que se puede llevar a cabo la comparación puesto que ha llegado un nuevo token al nodo de dos entradas. Puesto que la comparación es satisfactoria, esta instanciación de la producción puede ser añadida al conjunto de reglas candidatas a ser disparadas.

3. EL ALGORITMO ASIA

FLEX es un shell especializado en la construcción de sistemas expertos en tiempo real para la resolución de problemas de control. FLEX está compuesto por los siguiente módulos:

- FLEXIE: motor de inferencia.
- FLEXEN: interface con el entorno.

FLEXIE usa un algoritmo denominado ASIA (ASinchronous Information Access) para llevar a cabo el emparejamiento de patrones. ASIA es una extensión del algoritmo RETE hecha para poder incluir datos de sensor y datos de estado en el proceso de inferencia.

- *Datos de Sensor:* Son datos de entrada y se usan para representar las señales de los sensores externos y de los diversos dispositivos de medida.
- *Datos de Estado:* Se usan para representar información de realimentación que será intercambiada con otros módulos de control cuando un sistema experto construido con FLEX opera como una parte de un sistema de control más complejo.

Como consecuencia de la necesidad de manejar los dos nuevos tipos de datos descritos, la red RETE incluirá dos nuevos tipos de nodos debido a que:

1. Se debe permitir hacer referencia a datos de sensor y de estado en la posición valor de los pares <atributo,valor>.
   Existirá en consecuencia un nodo, denominado nodo ASIA, que leerá los datos de sensor o de estado antes de realizar la comparación.
2. Se pueden producir situaciones de alarma por lo que hemos de disponer de un nuevo tipo de nodo terminal, denominado nodo terminal ASIA, que verificará si la RHS de la regla en cuestión genera una condición de alarma. Si esto es así, la regla es disparada dejando de lado el resto de la fase de emparejamiento y de resolución de conflictos, y en caso contrario actúa como un nodo terminal normal.

Veamos a continuación un ejemplo de red modificada con nodos ASIA. Supongamos el siguiente conjunto de reglas:

```
Regla R1:

   If
      (C1 #A1 1 #A2 <x>)
      (C2 #A1 <x> #A2 25)
      (C3 #A1 <x>)
```

```
    Then
        ...
Regla R2:
        If
        (C2 #A1 <y> A2 25)
        (C4 #A1 <y>)
    Then
        (OutStat2 1)     /* Situación de Alarma */
```

La red resultante sería la de la figura 2.

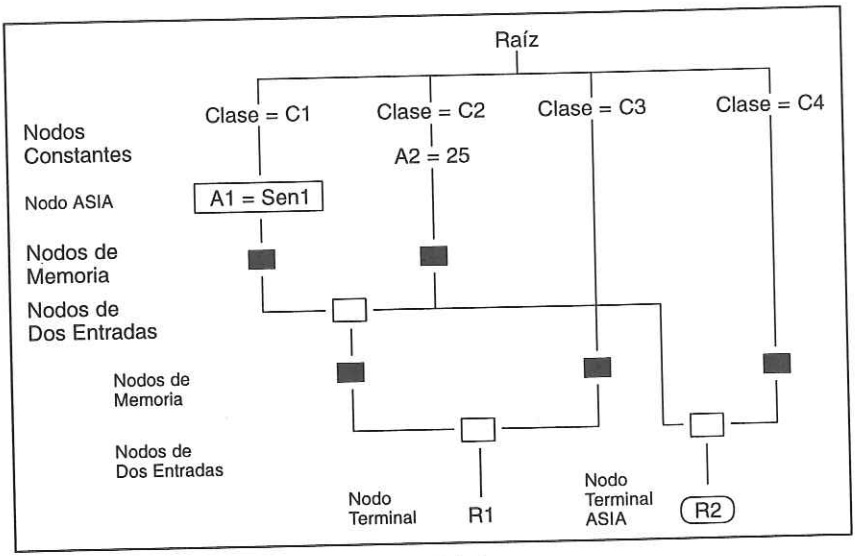

FIGURA 2

4. CONCLUSIONES

El algoritmo de emparejamiento RETE es eficiente incluso cuando tanto el número de patrones como de objetos es muy elevado. Sin embargo pueden darse situaciones en las que es preferible su no aplicación; puede ocurrir que los objetos de la memoria de trabajo cambien tan rápidamente que se produzcan situaciones de inconsistencia resultando por tanto el algoritmo tan inútil como ineficiente.

El algoritmo ASIA es una extensión del algoritmo RETE para el cual es relativamente fácil diseñar una arquitectura para un emparejador de patrones hardware. Esta implementación hardware del algoritmo resulta especialmente interesante para las aplicaciones de tiempo real las cuales imponen fuertes restricciones temporales.

6. BIBLIOGRAFÍA

[1] DEVEDZIC, V.; VELASEVIC, D. (1992): «Fast Reasoning with External Data on Personal Computers». *Engns. Applic. Artif. Intell.* (5), pp. 87-99.

[2] FORGY, C. L. (1982): «Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem». *Artificial Intelligence* (19), pp. 17-37.

[3] NILSSON, N. J. (1987): *Principios de Inteligencia Artificial*. Díaz de Santos.