Tema 9: Estilo y eficiencia en BC basadas en reglas

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo
Francisco J. Martín Mateos

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Contenido y bibliografía

- **Ordenación de condiciones**
- **Uso de variables múltiples**
- **Test y restricciones**
- **Reglas específicas o generales**
- **Bibligrafía**
  - **Giarratano, J.C. y Riley, G. "Expert Systems Principles and Programming (2nd ed.)" (PWS Pub. Co., 1994)**
    *** Cap. 11: "Efficiency in Rule-Based Languages"**
  - **Giarratano, J.C. "CLIPS 6.0 User's Guide"**

Ordenación de condiciones

- **Ejemplo:** `ej-1.clp`

```
(deffacts informacion
  (letras a e i)
  (letra a)
  (letra e)
  (letra i))

(defrule regla-1
  (letras ?x ?y ?z)
  (letra ?x)
  (letra ?y)
  (letra ?z)
  =>
  (assert (letras-encontradas ?x ?y ?z)))

(defrule regla-2
  (letra ?x)
  (letra ?y)
  (letra ?z)
  (letras ?x ?y ?z)
  =>
  (assert (letras-encontradas ?x ?y ?z)))
```

Ordenación de condiciones

- **Equiparación de condiciones en reglas**

  `(matches <regla>)`

- **Equiparación de condiciones para la `regla-1`**

```
CLIPS> (load "ej-1.clp")
$**
TRUE
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (letras a e i)
f-2     (letra a)
f-3     (letra e)
f-4     (letra i)
For a total of 5 facts.
CLIPS> (matches regla-1)
Matches for Pattern 1: f-1
Matches for Pattern 2: f-2, f-3, f-4
Matches for Pattern 3: f-2, f-3, f-4
Matches for Pattern 4: f-2, f-3, f-4
Partial matches for CEs 1 - 2
 f-1, f-2
Partial matches for CEs 1 - 3
 f-1, f-2, f-3
Partial matches for CEs 1 - 4
 f-1, f-2, f-3, f-4
Activations
 f-1, f-2, f-3, f-4
```

Ordenación de condiciones

- **Equiparación de condiciones para la `regla-2`**

```
CLIPS> (facts)
f-0     (initial-fact)
f-1     (letras a e i)
f-2     (letra a)
f-3     (letra e)
f-4     (letra i)
For a total of 5 facts.
CLIPS> (matches regla-2)
Matches for Pattern 1: f-2, f-3, f-4
Matches for Pattern 2: f-2, f-3, f-4
Matches for Pattern 3: f-2, f-3, f-4
Matches for Pattern 4: f-1
Partial matches for CEs 1 - 2
 [f-4,f-4], [f-4,f-3], [f-4,f-2], [f-2,f-4],
 [f-3,f-4], [f-3,f-3], [f-3,f-2], [f-2,f-3],
 [f-2,f-2]
Partial matches for CEs 1 - 3
 [f-4,f-4,f-4], [f-4,f-4,f-3], [f-4,f-4,f-2],
 [f-4,f-3,f-4], [f-4,f-3,f-3], [f-4,f-3,f-2],
 [f-4,f-2,f-4], [f-4,f-2,f-3], [f-4,f-2,f-2],
 [f-2,f-4,f-4], [f-2,f-4,f-3], [f-2,f-4,f-2],
 [f-3,f-4,f-4], [f-3,f-4,f-3], [f-3,f-4,f-2],
 [f-2,f-2,f-4], [f-2,f-3,f-4], [f-3,f-2,f-4],
 [f-3,f-3,f-4], [f-3,f-3,f-3], [f-3,f-3,f-2],
 [f-3,f-2,f-3], [f-3,f-2,f-2], [f-2,f-3,f-3],
 [f-2,f-3,f-2], [f-2,f-2,f-3], [f-2,f-2,f-2]
Partial matches for CEs 1 - 4
 [f-2,f-3,f-4,f-1]
Activations
 [f-2,f-3,f-4,f-1]
```

Ordenación de condiciones

- **Heurística para la ordenación de las condiciones**
  - **Colocar al principio las más específicas**
  - **Colocar al principio los hechos de control**
  - **Colocar al final las más volátiles**

Uso de variables múltiples

- **Ineficiencia de multicampos**

```
(deffacts informacion
  (objeto nombre "Objeto 1"
          peso 120 pos-x 2 pos-y 3)
  (objeto nombre "Objeto 2"
          peso  70 pos-x 4 pos-y 5))

(defrule buscar-pesados
  (objeto $? peso ?peso&:(> ?peso 100) $?)
  =>
  (printout t "Encontrado objeto pesado de peso "
              ?peso crlf))
```

- **Equiparaciones de la primera variable múltiple**

```
Intento Campos
 1
 2      nombre
 3      nombre "Objeto 1"
 4      nombre "Objeto 1" peso
 5      nombre "Objeto 1" peso 120
 6      nombre "Objeto 1" peso 120 pos-x
 7      nombre "Objeto 1" peso 120 pos-x 2
 8      nombre "Objeto 1" peso 120 pos-x 2 pos-y
 9      nombre "Objeto 1" peso 120 pos-x 2 pos-y 3
```

Uso de variables múltiples

- **Mejora con campos simples**

```
(deffacts informacion
  (objeto "Objeto 1" 120 2 3)
  (objeto "Objeto 2"  70 4 5))

(defrule buscar-pesados
  (objeto ? ?peso&:(> ?peso 100) $?)
  =>
  (printout t "Encontrado objeto pesado de peso "
              ?peso crlf))
```

- **Mejora con plantillas**

```
(deftemplate objeto
  (slot nombre)
  (slot peso)
  (slot pos-x)
  (slot pos-y))

(deffacts informacion
  (objeto (nombre "Objeto 1") (peso 120)
          (pos-x 2) (pos-y 3))
  (objeto (nombre "Objeto 2") (peso 70)
          (pos-x 4) (pos-y 5)))

(defrule buscar-pesados
  (objeto (peso ?peso&:(> ?peso 100)))
  =>
  (printout t "Encontrado objeto pesado de peso "
              ?peso crlf))
```

Tests y restricciones

- **Ejemplo:** `ej-3.clp`

```
(defrule puntos-ordenados-1
  (punto ?x1 ?y1)
  (punto ?x2 ?y2)
  (punto ?x3 ?y3)
  (test (and (< ?x1 ?x2 ?x3)
             (< ?y1 ?y2 ?y3)))
  =>
  (assert (ordenados ?x1 ?y1 ?x2 ?y2 ?x3 ?y3)))

(defrule puntos-ordenados-2
  (punto ?x1 ?y1)
  (punto ?x2 ?y2)
  (test (and (< ?x1 ?x2)
             (< ?y1 ?y2)))
  (punto ?x3 ?y3)
  (test (and (< ?x2 ?x3)
             (< ?y2 ?y3)))
  =>
  (assert (ordenados ?x1 ?y1 ?x2 ?y2 ?x3 ?y3)))

(deffacts puntos
  (punto 1 2)
  (punto 4 2)
  (punto 1 1)
  (punto 5 3))
```

Tests y restricciones

- **Sesión:**

```
CLIPS> (clear)
CLIPS> (load "ej-3.clp")
**$
TRUE
CLIPS> (reset)
CLIPS> (matches puntos-ordenados-1)
Matches for Pattern 1: f-1, f-2, f-3, f-4
Matches for Pattern 2: f-1, f-2, f-3, f-4
Matches for Pattern 3: f-1, f-2, f-3, f-4
Partial matches for CEs 1 - 2
 [f-4,f-4], [f-4,f-3], [f-4,f-2], [f-4,f-1],
 [f-1,f-4], [f-2,f-4], [f-3,f-4], [f-3,f-3],
 [f-3,f-2], [f-3,f-1], [f-1,f-3], [f-2,f-3],
 [f-2,f-2], [f-2,f-1], [f-1,f-2], [f-1,f-1]
Partial matches for CEs 1 - 3
 [f-3,f-2,f-4]
Activations
 [f-3,f-2,f-4]
CLIPS> (matches puntos-ordenados-2)
Matches for Pattern 1: f-1, f-2, f-3, f-4
Matches for Pattern 2: f-1, f-2, f-3, f-4
Matches for Pattern 3: f-1, f-2, f-3, f-4
Partial matches for CEs 1 - 2
 [f-1,f-4], [f-2,f-4], [f-3,f-4], [f-3,f-2]
Partial matches for CEs 1 - 3
 [f-3,f-2,f-4]
Activations
 [f-3,f-2,f-4]
```

Tests y restricciones

- **Con test:**

```
(defrule paralela-a-eje-1
  (punto ?x1 ?y1)
  (punto ?x2 ?y2)
  (test (or (= ?x1 ?x2) (= ?y1 ?y2)))
  =>
  (assert (paralela ?x1 ?y1 ?x2 ?y2)))
```

- **Con restricciones:**

```
(defrule paralela-a-eje-2
  (punto ?x1 ?y1)
  (punto ?x2 ?y2&:(or (= ?x1 ?x2) (= ?y1 ?y2)))
  =>
  (assert (paralela ?x1 ?y1 ?x2 ?y2)))
```

Tests y restricciones

- **Con elementos condicionales:**

```
(defrule color-basico-1
  (color ?x&:(or (eq ?x rojo)
                 (eq ?x amarillo)
                 (eq ?x azul)))
  =>
  (assert (color-basico ?x)))
```

- **Con restriciones:**

```
(defrule color-basico-2
  (color ?x&rojo|amarillo|azul)
  =>
  (assert (color-basico ?x)))
```

Reglas específicas o generales

- **Reglas específicas:**

```
(deftemplate posicion (slot x) (slot y))

(defrule mover-al-norte
  ?h <- (movimiento norte)
  ?posicion-actual <- (posicion (y ?y-actual))
  =>
  (retract ?h)
  (modify ?posicion-actual (y (+ ?y-actual 1))))

(defrule mover-al-sur
  ?h <- (movimiento sur)
  ?posicion-actual <- (posicion (y ?y-actual))
  =>
  (retract ?h)
  (modify ?posicion-actual (y (- ?y-actual 1))))

(defrule mover-al-este
  ?h <- (movimiento este)
  ?posicion-actual <- (posicion (x ?x-actual))
  =>
  (retract ?h)
  (modify ?posicion-actual (x (+ ?x-actual 1))))

(defrule mover-al-oeste
  ?h <- (movimiento oeste)
  ?posicion-actual <- (posicion (x ?x-actual))
  =>
  (retract ?h)
  (modify ?posicion-actual (x (- ?x-actual 1))))
```

Reglas específicas o generales

- **Reglas generales:**

```
(deftemplate posicion (slot x) (slot y))

(deftemplate variacion
  (slot direccion)
  (slot variacion-x (default 0))
  (slot variacion-y (default 0)))

(deffacts informacion-de-variacion
  (variacion (direccion norte) (variacion-y 1))
  (variacion (direccion sur)   (variacion-y -1))
  (variacion (direccion este)  (variacion-x 1))
  (variacion (direccion oeste) (variacion-x -1)))

(defrule mover
  ?h <- (movimiento ?d)
  ?p-actual <- (posicion (x ?x-actual) (y ?y-actual))
  (variacion (direccion ?d)
             (variacion-x ?vx)
             (variacion-y ?vy))
  =>
  (retract ?h)
  (modify ?p-actual (x (+ ?x-actual ?vx))
                    (y (+ ?y-actual ?vy))))
```