Clips: Estilo y eficiencia

José A. Alonso y Francisco J. Martín

Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Ordenación de patrones

- **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 patrones

- **Sesión**

```
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> (ppdefrule regla-1)
(defrule MAIN::regla-1
   (letras ?x ?y ?z)
   (letra ?x)
   (letra ?y)
   (letra ?z)
   =>
   (assert (letras-encontradas ?x ?y ?z)))
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 patrones

```
CLIPS> (ppdefrule regla-2)
(defrule MAIN::regla-2
   (letra ?x)
   (letra ?y)
   (letra ?z)
   (letras ?x ?y ?z)
   =>
   (assert (letras-encontradas ?x ?y ?z)))
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 patrones

- **Heurística para ordenación de patrones**
  - **Colocar al principio los más específicos**
  - **Colocar al final los más volátiles**
  - **Colocar al principio los hechos de control**

Multicampos

- **Utilidad de multicampos**

```
(deffacts informacion
  (lista a b c))

(deftemplate particion
  (multislot principio)
  (multislot centro)
  (multislot final))

(defrule particion
  (lista $?principio $?centro $?final)
  =>
  (assert (particion (principio ?principio)
                     (centro ?centro)
                     (final  ?final))))
```

Multicampos

- **Sesión**

```
CLIPS> (clear)
CLIPS> (load "ej-3.clp")
$%*
TRUE
CLIPS> (reset)
CLIPS> (run)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (lista a b c)
f-2     (particion (principio) (centro) (final a b c))
f-3     (particion (principio) (centro a) (final b c))
f-4     (particion (principio) (centro a b) (final c))
f-5     (particion (principio) (centro a b c) (final))
f-6     (particion (principio a) (centro) (final b c))
f-7     (particion (principio a) (centro b) (final c))
f-8     (particion (principio a) (centro b c) (final))
f-9     (particion (principio a b) (centro) (final c))
f-10    (particion (principio a b) (centro c) (final))
f-11    (particion (principio a b c) (centro) (final))
For a total of 12 facts.
```

Multicampos

- **Ineficiencia de multicampos**

```
(deffacts informacion
  (objeto nombre "Objeto 1"
          peso 120 posicion-x 2 posicion-y 3)
  (objeto nombre "Objeto 2"
          peso  70 posicion-x 4 posicion-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 posicion-x
 7      nombre "Objeto 1" peso 120 posicion-x 2
 8      nombre "Objeto 1" peso 120 posicion-x 2 posicion-y
 9      nombre "Objeto 1" peso 120 posicion-x 2 posicion-y
```

Multicampos

- **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     (type STRING))
  (slot peso       (type NUMBER))
  (slot posicion-x (type NUMBER))
  (slot posicion-y (type NUMBER)))

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

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

Localización de los tests

- **Ejemplo:** `ej-4.clp`

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

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

Localización de los tests

- **Sesión:**

```
CLIPS> (clear)
CLIPS> (load "ej-4.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]
```

Test 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)))
```

Primitivas de equiparación

- **Equiparación lógica:**

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

- **Equiparación con primitivas:**

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

Cuadrados mágicos

- **Problema:**

```
ABC          {A,B,C,D,E,F,G,H,I} = {1,2,3,4,5,6,7,8,9}
DEF          A+B+C = D+E+F = G+H+I = A+D+G = B+E+F
GHI                = C+F+I = A+E+I = C+E+G
```

- **Programa** `cuadrado-magico.clp`**:**

```
(deffacts datos
  (numero 1) (numero 2) (numero 3) (numero 4)
  (numero 5) (numero 6) (numero 7) (numero 8)
  (numero 9) (solucion 0))

(deffunction suma-15 (?x ?y ?z)
  (= (+ ?x ?y ?z) 15))

(defrule busca-cuadrado
  (numero ?e)
  (numero ?a&~e)
  (numero ?i&~?e&~?a&:(suma-15 ?a ?e ?i))
  (numero ?b&~?e&~?a&~?i)
  (numero ?c&~?e&~?a&~?i&~b&:(suma-15 ?a ?b ?c))
  (numero ?f&~?e&~?a&~?i&~b&~c&:(suma-15 ?c ?f ?i))
  (numero ?d&~?e&~?a&~?i&~b&~c&~f&:(suma-15 ?d ?e ?f))
  (numero ?g&~?e&~?a&~?i&~b&~c&~f&~d&:(suma-15 ?a ?d ?g)
                                      &:(suma-15 ?c ?e ?g))
  (numero ?h&~?e&~?a&~?i&~b&~c&~f&~d&~g&:(suma-15 ?b ?e ?h)
                                         &:(suma-15 ?g ?h ?i)
  =>
  (assert (escribe-solucion ?a ?b ?c ?d ?e ?f ?g ?h ?i)))
```

Cuadrados mágicos

```
(defrule escribe-solucion
  ?fase <- (escribe-solucion ?a ?b ?c ?d ?e ?f ?g ?h ?i)
  ?solucion <- (solucion ?n)
  =>
  (retract ?fase ?solucion)
  (assert (solucion (+ ?n 1)))
  (printout t "Solucion " (+ ?n 1) ":" crlf)
  (printout t "   " ?a ?b ?c crlf)
  (printout t "   " ?d ?e ?f crlf)
  (printout t "   " ?g ?h ?i crlf)
  (printout t crlf))

(defrule escribe-numero-de-soluciones
  (declare (salience -10))
  ?solucion <- (solucion ?n)
  =>
  (printout t ?n " soluciones encontradas" crlf))
```

- **Sesión:**

```
CLIPS> (clear)
CLIPS> (load "cuadrado-magico.clp")
$!***
TRUE
CLIPS> (reset)
CLIPS> (run)
Solucion 1:
   492
   357
   816
...
12 soluciones encontradas
```

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