Tema 12
Clips: Técnicas de control

Técnicas de control

- Ejemplo de fases de un problema
  - Detección
  - Aislamiento
  - Recuperación
- Técnicas de control
  - Control empotrado en las reglas
  - Prioridades
  - Reglas de control
  - Módulos

Control empotrado en las reglas

- Fases en el Nim
  - Elección del jugador
  - Elección del número de piezas
  - Turno del humano
  - Turno de la computadora
- Hechos de control
  - `(fase elige-jugador)`
  - `(fase elige-numero-de-piezas)`
  - `(turno h)`
  - `(turno c)`
- Inconvenientes del control empotrado en las reglas
  - Dificultad para entenderse
  - Dificultad para precisar la conclusión de cada fase

Prioridades

- Ejemplo: `ej-1.clp`

```
(deffacts inicio
  (prioridad primera)
  (prioridad segunda)
  (prioridad tercera))

(defrule regla-1
  (prioridad primera)
  =>
  (printout t "Escribe primera" crlf))

(defrule regla-2
  (prioridad segunda)
  =>
  (printout t "Escribe segunda" crlf))

(defrule regla-3
  (prioridad tercera)
  =>
  (printout t "Escribe tercera" crlf))
```

Prioridades

- **Sesión**

```
CLISP> (load "ej-1.clp")
CLIPS> $***
TRUE
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (prioridad primera)
f-2     (prioridad segunda)
f-3     (prioridad tercera)
For a total of 4 facts.
CLIPS> (rules)
regla-1
regla-2
regla-3
For a total of 3 defrules.
CLIPS> (agenda)
0      regla-3: f-3
0      regla-2: f-2
0      regla-1: f-1
For a total of 3 activations.
CLIPS> (run)
Escribe tercera
Escribe segunda
Escribe primera
CLIPS>
```

Prioridades

- Ejemplo: `ej-2.clp`

```
(deffacts inicio
  (prioridad primera)
  (prioridad segunda)
  (prioridad tercera))

(defrule regla-1
  (declare (salience 30))
  (prioridad primera)
  =>
  (printout t "Escribe primera" crlf))

(defrule regla-2
  (declare (salience 20))
  (prioridad segunda)
  =>
  (printout t "Escribe segunda" crlf))

(defrule regla-3
  (declare (salience 10))
  (prioridad tercera)
  =>
  (printout t "Escribe tercera" crlf))
```

Prioridades

- **Sesión**

```
CLIPS> (load "ej-2.clp")
CLIPS> $***
TRUE
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (prioridad primera)
f-2     (prioridad segunda)
f-3     (prioridad tercera)
For a total of 4 facts.
CLIPS> (rules)
regla-1
regla-2
regla-3
For a total of 3 defrules.
CLIPS> (agenda)
30     regla-1: f-1
20     regla-2: f-2
10     regla-3: f-3
For a total of 3 activations.
CLIPS> (run)
Escribe primera
Escribe segunda
Escribe tercera
CLIPS>
```

Prioridades

- Sintaxis
  - `(declare (salience <numero>))`
- Valores
  - Mínimo: -10000
  - Máximo: 10000
  - Defecto: 0
- Inconveniente
  - Abuso
  - Contradicción con el objetivo de los sistemas basados en reglas

Reglas de control

- Reglas de cambio de fases `ej-3.clp`

```
(defrule deteccion-a-aislamiento
  (declare (salience -10))
  ?fase <- (fase deteccion)
  =>
  (retract ?fase)
  (assert (fase aislamiento)))

(defrule aislamiento-a-recuperacion
  (declare (salience -10))
  ?fase <- (fase aislamiento)
  =>
  (retract ?fase)
  (assert (fase recuperacion)))

(defrule recuperacion-a-deteccion
  (declare (salience -10))
  ?fase <- (fase recuperacion)
  =>
  (retract ?fase)
  (assert (fase deteccion)))

(defrule detecta-fuego
  (fase deteccion)
  (luz-a roja)
  =>
  (assert (problema fuego)))

(deffacts inicio
  (fase deteccion) (luz-a roja))
```

Reglas de control

- **Sesión**

```
CLIPS> (load "ej-3.clp")
CLIPS> ****$
TRUE
CLIPS> (watch rules)
CLIPS> (reset)
CLIPS> (run 7)
FIRE    1 detecta-fuego: f-1,f-2
FIRE    2 deteccion-a-aislamiento: f-1
FIRE    3 aislamiento-a-recuperacion: f-4
FIRE    4 recuperacion-a-deteccion: f-5
FIRE    5 detecta-fuego: f-6,f-2
FIRE    6 deteccion-a-aislamiento: f-6
FIRE    7 aislamiento-a-recuperacion: f-7
CLIPS>
```

Reglas de control

- Cambio de fase mediante una regla

```
(deffacts control
  (fase deteccion)
  (siguiente-fase deteccion aislamiento)
  (siguiente-fase aislamiento recuperacion)
  (siguiente-fase recuperacion deteccion))

(defrule cambio-de-fase
  (declare (salience -10))
  ?fase <- (fase ?actual)
  (siguiente-fase ?actual ?siguiente)
  =>
  (retract ?fase)
  (assert (fase ?siguiente)))
```

Reglas de control

- Cambio de fase mediante una regla y sucesión de fases

```
(deffacts control
  (fase deteccion)
  (sucesion-de-fases aislamiento recuperacion deteccion))

(defrule cambio-de-fase
  (declare (salience -10))
  ?fase <- (fase ?actual)
  (sucesion-de-fases ?siguiente $?resto)
  =>
  (retract ?fase)
  (assert (fase ?siguiente))
  (assert (sucesion-de-fases ?resto ?siguiente)))
```