

## Ordenación de la Agenda (I)

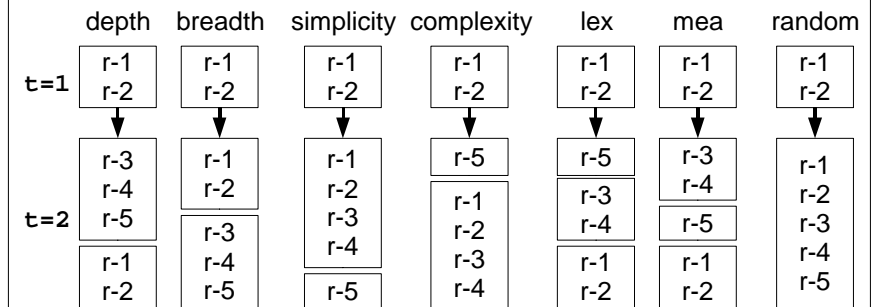
- Prioridad o saliencia de la regla. Entre -10000 y 10000. Por defecto 0.
- Estrategia de resolución de conflictos activa
  - Antigüedad de la regla
    - **depth**: prioridad regla activada más recientemente (por defecto)
    - **breadth**: prioridad regla activada más antiguamente
  - Especificidad de la parte izquierda de la regla (nº comparaciones LHS)
    - **simplicity**: prioridad regla más sencilla (menos comparaciones)
    - **complexity**: prioridad regla más compleja (más comparaciones)
  - Antigüedad de los hechos
    - **lex**: prioridad hechos más recientes reordenando todos en LHS
    - **mea**: prioridad 1<sup>er</sup> hecho en LHS más reciente + reordenación resto LHS
  - **random**: aleatoria, pero repetible en sucesivas sesiones
- Arbitrariamente. No aleatorio. Depende de la implementación.

## Ordenación de la Agenda (II)

```
(defrule r-1 (f-a) =>) (defrule r-2 (f-a) =>)
(defrule r-3 (f-b) =>) (defrule r-4 (f-b) =>)
(defrule r-5 (f-a) (f-b) =>)
```

```
1º t=1 (assert (f-a))
```

```
2º t=2 (assert (f-b))
```



## Ordenación de la Agenda (III)

- **(get-strategy)**  
devuelve el valor de la estrategia actual
- **(set-strategy depth|breadth|simplicity|complexity|lex|mea|random)**  
cambia el valor de la estrategia actual
- Puede cambiarse en el menú *Execution/Options*
- Las reglas son “refractarias”: no se vuelven a disparar con los mismos hechos

## Ejercicio-4

Definidas las siguientes reglas:

```
(defrule r1 (f-1)(f-2)(f-3) =>)
(defrule r2 (f-3)(f-1) =>)
(defrule r3 (f-2)(f-1) =>)
(defrule r4 (f-1)(f-2) =>)
(defrule r5 (f-1)(f-2)(f-3) =>)
(defrule r6 (f-1) (f-4) =>)
```

Evaluar la ordenación de la agenda con cada estrategia después de asertar los siguientes hechos:

```
1º t=1 (assert (f-1))
2º t=2 (assert (f-2))
3º t=3 (assert (f-3))
4º t=4 (assert (f-4))
```

## Hechos (V)

### • Hechos desordenados

- Se definen usando una plantilla:

```
(deftemplate <nom-relación> [<comentario>]
  (slot|multislot <nom-slot> atributo*))
```

- Contienen campos o ranuras (slot):

- slot = ranura sencilla = tiene un único dato
- multislot = ranura múltiple = tiene cero, uno o más datos

- El orden de los campos no es relevante

Ej.: (deftemplate persona (defacts personas-1

```
"datos de una persona"      (persona (nombre ana)
  (slot nombre)              ;nombre      (edad 15)
  (slot edad)                 ;edad        (e-civil soltero)
  (slot e-civil)               ;estado civil (estudios egb))
  (multislot estudios)         ;estudios
```

## Hechos (VI)

- Ej.: (clear)

```
(deftemplate persona "datos de una persona"
  (slot nombre)(slot edad)(slot e-civil)(multislot estudios))
(defacts personas-1
  (persona (nombre luci) (edad 15)
    (e-civil soltero) (estudios egb))
  (persona (nombre pepe)
    (e-civil soltero)(estudios) (edad 23))
  (persona (edad 44) (e-civil casado)
    (nombre mari)(estudios derecho economicas)))

(reset)
(retract 3)
(assert (persona (nombre juan) (edad 75)
  (e-civil viudo) (estudios ats)))
(modify 4 (edad 85) (e-civil casado))
(modify 1 (nombre lucia))
(duplicate 2 (nombre paco))
```

## Reglas (III)

0.(initial-fact)

2.(persona (nombre pepe)(e-civil soltero)(estudios)(edad 23))

5.(persona (nombre juan)(edad 85)(e-civil casado)(estudios ats))

6.(persona (nombre lucia)(edad 15)(e-civil soltero)(estudios egb))

7.(persona (nombre paco)(e-civil soltero)(estudios)(edad 23))

```
(defrule prueba slots
  (persona (e-civil soltero))
=> ... )
```

2  
6  
7

```
(defrule prueba slots
  (persona (e-civil soltero)(estudios))
=> ... )
```

2  
7

```
(defrule prueba slots
  (persona (e-civil soltero)(edad 34)(estudios))
=> ... )
```

## Ranuras y atributos (I)

- Tipo: (type <tipo-permitido>+)

- NUMBER

- INTEGER
- FLOAT

- LEXEME

- SYMBOL
- STRING

```
(deftemplate t1 (slot s1 (type FLOAT)))
```

```
(deftemplate t1 (slot s1 (type INTEGER LEXEME)))
```

- Valores permitidos: (allowed-values <valor>\*)

```
(deftemplate t1 (slot s1 (allowed-values 1 2 3)))
```

- Constantes permitidas: (allowed-<tipo>s <valor>\*)+

```
(deftemplate t1 (slot s1 (allowed-symbols pepe juan)))
```

## Ranuras y atributos (II)

- Rango: (`range <lim-inf> <lim-sup>`)

```
(deftemplate t1 (slot s1 (range 0 99)))
(deftemplate t1 (slot s1 (range 0 ?VARIABLE)))
```

- Cardinalidad: (`cardinality <lim-inf> <lim-sup>`)

```
(deftemplate t1 (multislot s1 (cardinality 0 4)))
(deftemplate t1 (multislot s1 (cardinality 0 ?VARIABLE)))
```

- Valor por defecto: (`default <valor-por-defecto>`)

```
(deftemplate t1 (slot s1 (default soltero)))
(deftemplate t1 (slot s1 (default ?NONE)))
(deftemplate t1 (slot s1 (default ?DERIVE)))
```

- Valor por defecto dinámico: (`default-dynamic <expresión>*`)

## Ranuras y atributos (III)

- Control de restricciones en los atributos
  - (`get-static-constraint-checking`)
  - (`set-static-constraint-checking FALSE|TRUE`)
  - (`get-dynamic-constraint-checking`)
  - (`get-dynamic-constraint-checking FALSE|TRUE`)
  - En el menú *Execution/Options*

- Ejemplo:

```
(deftemplate medico "datos de un medico"
  (slot nombre (type STRING) (default ?NONE))
  (slot edad (type INTEGER)(range 0 ?VARIABLE))
  (slot sexo (allowed-values hombre mujer))
  (multislot guardias (type INTEGER) (range 1 31)
    (cardinality 0 4)))
```

## Ejercicio-5

Dada la siguiente plantilla:

```
(deftemplate medico "datos de un medico"
  (slot nombre (type STRING) (default ?NONE))
  (slot edad (type INTEGER)(range 0 ?VARIABLE))
  (slot sexo (allowed-values V H) (default H))
  (multislot guardias (type INTEGER) (range 1 31)
    (cardinality 0 4)))
```

Evaluar las siguientes órdenes:

```
(assert (medico))
(assert (medico (nombre "Pepe") (edad -5)))
(assert (medico (nombre "Maria")
  (guardias 7 10 20 24 28)))
(assert (medico (nombre "Juan") (sexo)
  (guardias 7 10)))
(assert (medico (nombre "Ana")
  (guardias 7 10)))
```

## Ejercicio-6

Modificaciones al ejercicio 3 (II):

Recuérdese que un alumno habrá realizado dos exámenes y dos prácticas, con las posibles calificaciones "suspense", "aprobado", "notable" y "sobresaliente"; y también es posible que haya realizado un trabajo complementario.

Escribir un `deftemplate` adecuado para representar la información inicial de un alumno, y escribir de nuevo el correspondiente programa CLIPS.

Nota:

Los valores calculados por el programa no formarán parte de la plantilla, sino que se almacenarán en hechos ordenados:

```
(oav practica <valor>),
(oav teoria <valor>) y
(oav asignatura <valor>)
```

## Ejercicio-7

La encargada de controlar el funcionamiento de la STT actúa sobre las palancas A1 y A2 y la conexión a la red eléctrica.

Entrevista entre una hábil ingeniera del conocimiento y la encargada de controlar el funcionamiento de la STT:

C.- Observo que a veces usted apaga la STT. Es una medida extrema, ¿no?

M.- Sí. Apago la STT cuando la temperatura y la presión son demasiado elevadas.

C.- ¿Y cómo sabe cuándo la temperatura es demasiado elevada?

M.- Para eso está el sensor S2. Cuando está hacia la izquierda, sé que la temperatura es excesiva.

C.- ¿Y la presión?

M.- Compruebo si el sensor S1 está indicando a la derecha. Cuando ocurre esto, la presión aún no es muy elevada.

C.- Creo que está claro. Otras veces he observado que usted actúa sobre la palanca A1.

M.- Sí, lo hago para bajar la presión. Cuando la presión es alta, acciono A1.

C.- ¿Pero no hemos dicho que lo que hacemos es apagar la STT?

M.- No, eso lo hacemos cuando son a la vez excesivas la presión y la temperatura.

C.- Ya entiendo. Entonces, cuando sólo la presión es excesiva, no hace falta apagarla.

M.- Eso es.

C.- Entonces, se actúa sobre A1 sólo cuando S1 apunta a la izquierda, ¿no?

M.- Sí... Bueno, cuando S1 está casi vertical, considero que la presión es un poco elevada y acciono la palanca A2.

C.- Entonces A2 sólo se toca en este caso, cuando S1 está casi vertical.

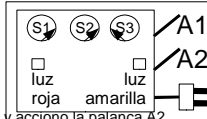
M.- No. A2 también me sirve para controlar la temperatura. Cuando es excesiva, la acciono.

C.- ¿Y para qué sirve la luz amarilla?

M.- Se me olvidaba: siempre que la luz amarilla está encendida, la presión debe considerarse demasiado alta.

C.- Queda por saber qué pasa cuando se enciende la luz roja.

M.- Nada. La verdad es que nunca la miro, ni tampoco el sensor S3.



Se pide:

1. Definir las plantillas adecuadas para representar sensores, luces y acciones, y plasmar el conocimiento extraído en la anterior entrevista en un conjunto de reglas CLIPS.

2. Ejecutar el programa CLIPS con los hechos iniciales correspondientes a la situación en que los sensores S1 y S3 apuntan completamente a la derecha, el S2 completamente a la izquierda y las luces están apagadas.

## Ficheros

- Construcciones

- guarda construcciones (**def...**)

- CLP

- Lotes (Batch)

- guarda construcciones (**def...**)

- guarda órdenes y funciones (**assert, reset, ...**)

- BAT

- Binarios

- guarda las construcciones presentes en el entorno actual

- BIN

- Forma elegante: construcciones en \*.clp y órdenes en \*.bat