

---

Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

## Tema 3: Hechos II

Aurora Esteban Toscano  
aestebant@uco.es

José Manuel Alcalde Llergo  
i72alllj@uco.es

Marzo de 2023



- Profundizar en los *hechos no ordenados* en CLIPS.
- Profundizar en la afirmación de hechos en CLIPS.
- Conocer los comandos de gestión de hechos en CLIPS.



Los hechos representan la información actual del Sistema Experto: determinan cómo actúa y se actualizan según las condiciones cambian (las distintas reglas se disparan).

Tipos de hechos:

- Ordenados: son una relación de valores. Por ejemplo:  
(lista huevos leche “pimiento rojo”)
- No ordenados: su estructura se determina por una plantilla.

Los hechos no ordenados permiten representar información con la relación entre conceptos y atributos.

Por ejemplo

```
(deftemplate coche “Es un coche”  
  (slot marca)  
  (slot modelo)  
  (slot color))  
(assert (coche (marca Seat)))  
(assert (coche (marca Audi) (modelo A1) (color rojo)))
```



## Constructor

```
(deftemplate <nombre> [comentario]
  (slot|multislot <nombre> [atributos*])+)
)
```

- Las plantillas deben tener nombre único: si se define una plantilla con el mismo nombre que otra ya existente, la sobrescribirá.
- Las plantillas no se pueden redefinir mientras estén en uso: si un hecho o una regla se base en ella.
- Las plantillas se componen de un numero variable de *campos*:
  - Monocampos o campos simples: slot.
  - Multicampos o campos compuestos: multislot.



- 1 Descargar de Moodle el fichero y observar la estructura de la plantilla.

```
(deftemplate persona "datos de una persona"  
  (slot nombre)  
  (slot edad)  
  (multislot direccion)  
)
```

- 2 Cargar la plantilla en CLIPS.

- 3 Afirmar los hechos:

- Persona cualquiera de 34 años.
- Juan, de 26 años.
- Marta, de 21 años y dirección Avd de Cervantes.



- Valor por defecto. (slot <nombre> (default | default-dynamic <por defecto>))
  - default: valor por defecto estático:
    - ?DERIVE: derivado de las restricciones del campo. Comportamiento por defecto.
    - Puede ser un valor primitivo.
    - ?NONE: forzar que no se derive  $\Rightarrow$  campo obligatorio.
  - default-dynamic: valor por defecto dinámico. La expresión es evaluada cada vez que se crea una instancia.
- Tipo del campo. (slot <nombre> (type <tipo>))
  - Se permiten los tipos de datos primitivos: INTEGER, FLOAT, NUMBER, SYMBOL, STRING, LEXEME



- Valores permitidos. (slot <nombre> (allowed-... ))
  - allowed-values
  - allowed-integers
  - allowed-numbers
  - allowed-floats
  - allowed-lexemes
  - allowed-symbols
  - allowed-strings
- Rango de valores permitido. (slot <nombre> (range <min> <max>))
- Cardinalidad permitida en campos múltiples. (multislot <nombre> (cardinality <min> <max>))



- 1 Descargar de Moodle el fichero y observar la estructura de la plantilla.

```
(deftemplate equipo "datos de un equipo de futbol"  
  (slot id (default-dynamic (gensym*)))  
  (slot nombre (type STRING) (default ?NONE))  
  (slot posicion (type INTEGER) (range 1 10) (default 5))  
  (multislot equipacion (type SYMBOL) (allowed-symbols blanco rojo  
    azul verde amarillo))  
  (multislot plantilla (type LEXEME) (cardinality 3 7))  
)
```

- 2 Cargar la plantilla en CLIPS. Cargar los hechos definidos en liga
- 3 ¿Qué pasa con el campo id? ¿Y con el campo nombre?
- 4 ¿Cuál es la posición máxima que puede ocupar un equipo?
- 5 ¿Cuántos jugadores pueden componer un equipo?





- (list-deftemplates): listar las plantillas usadas por el sistema.
- (ppdeftemplate <nombre>): muestra la definición de una plantilla existente.
- (undeftemplate <nombre>): borra una plantilla existente.



## Dos modalidades:

- **assert**: introducir en la base de hechos los hechos dinámicamente. Por ejemplo:  

```
(assert (a b c))  
(assert (a) (b) (c))  
(assert (a b) (c d) (e))
```
- **deffacts**: inicializar la base de hechos al principio. Por ejemplo:  

```
(deffacts h1 "hechos iniciales" (a b) (c d) (e))
```

  - La base de hechos no se inicializará hasta ejecutar `(reset)`
- **OjO**: no se puede insertar un hecho que ya existe.



## Constructor

```
(deffacts <nombre> [comentario]
  <hecho>+
)
```

- Pueden existir múltiples constructores deffacts, cada uno con cualquier número de hechos tanto ordenados como no ordenados.
- Si se carga un nuevo deffacts con el mismo nombre de otro existente, sobrescribe a los hechos que éste definiera.
- Puede contener hechos con expresiones dinámica. El hecho tomará el valor de la expresión resultante.

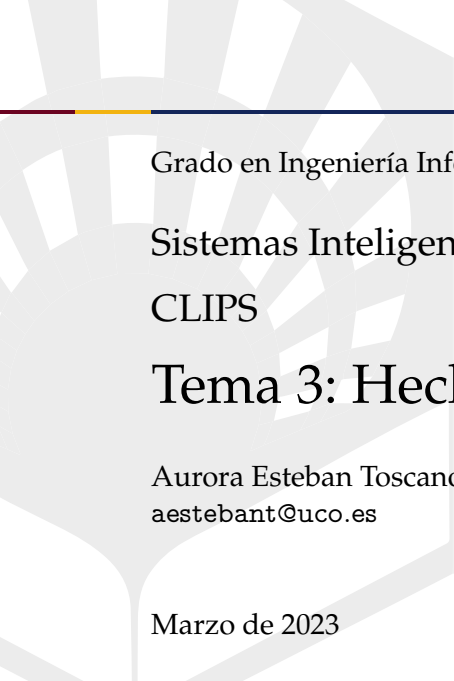


- (list-deffacts): listar los constructores de hechos definidos.
- (ppdeffacts <nombre>): muestra la definición de un constructor de hechos dado.
- (undefacts <nombre>): suprime los hechos afirmados en un constructor de hechos dado.



# Por ejemplo

```
CLIPS> (clear)
CLIPS> (deffacts EstadoCoche (coche motor "a punto") (coche bujias "limpias"))
CLIPS> (assert (coche neumaticos "gastados"))
<Fact-1>
CLIPS> (reset)
CLIPS> (facts)
f-1    (coche motor "a punto")
f-2    (coche bujias "limpias")
CLIPS> (assert (coche luces "funcionan"))
<Fact-3>
CLIPS> (deffacts EstadoCasa (casa salon "ordenado"))
CLIPS> (reset)
CLIPS> (facts)
f-1    (coche motor "a punto")
f-2    (coche bujias "limpias")
f-3    (casa salon "ordenado")
CLIPS> (list-deffacts)
EstadoCoche
EstadoCasa
```



---

Grado en Ingeniería Informática, Universidad de Córdoba

Sistemas Inteligentes

CLIPS

## Tema 3: Hechos II

Aurora Esteban Toscano  
aestebant@uco.es

José Manuel Alcalde Llergo  
i72alllj@uco.es

Marzo de 2023