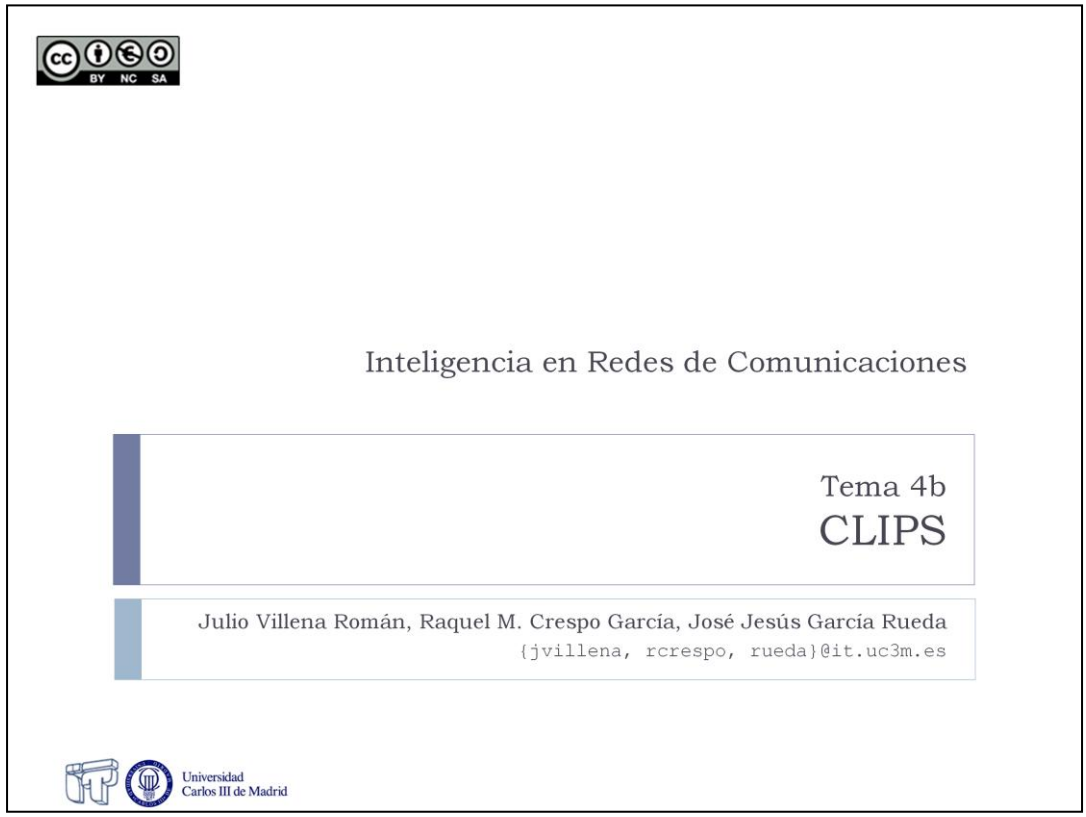
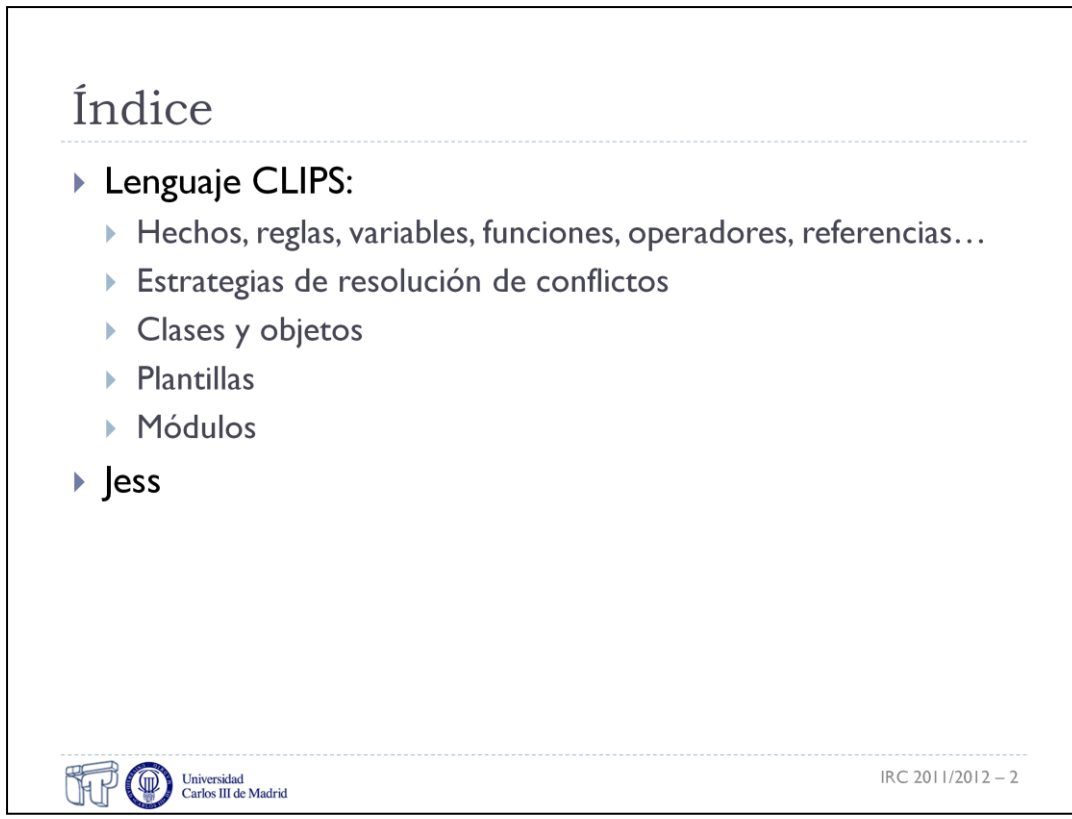
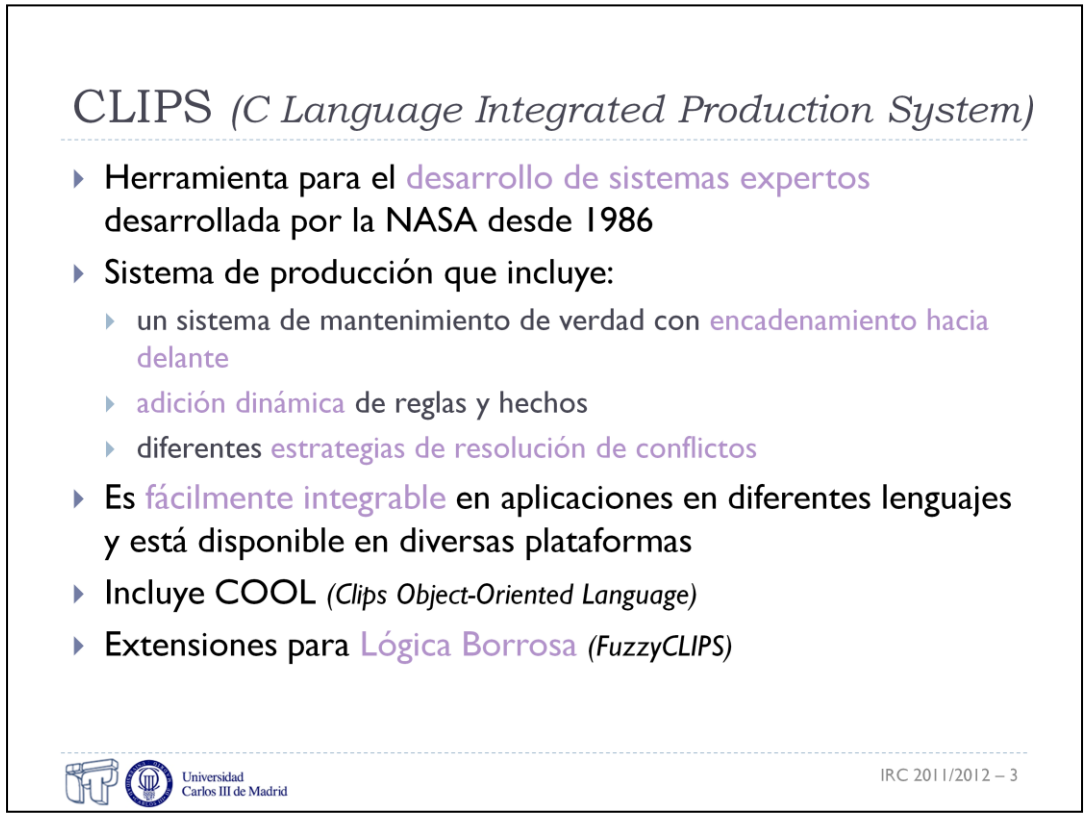


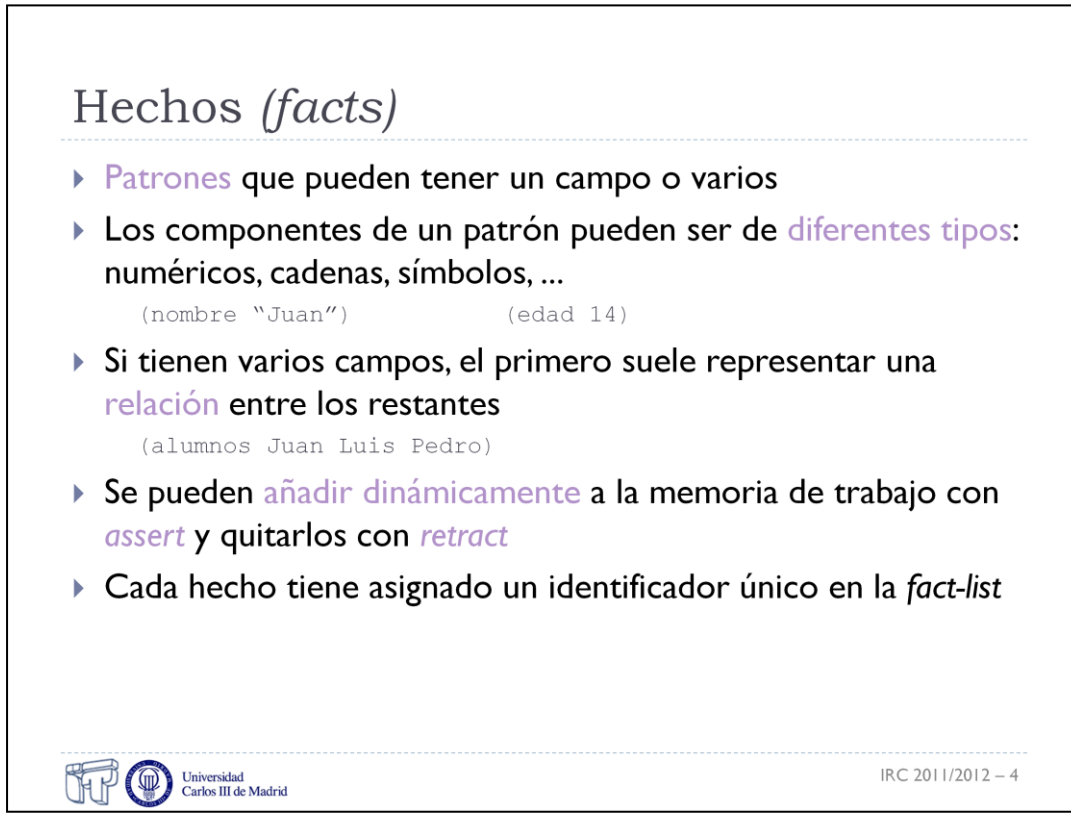
En este Tema 4b se van a estudiar los fundamentos del lenguaje CLIPS para la creación de sistemas basados en reglas.



Este es el índice del tema: se estudiarán los diferentes conceptos y fundamentos del lenguaje CLIPS y su reimplementación Jess.



CLIPS es una herramienta (y un lenguaje) desarrollado por la NASA desde 1986 para la construcción de sistemas expertos.



Entre los elementos de CLIPS están los hechos: patrones que representan información en la base de conocimientos.

Agrupación de hechos

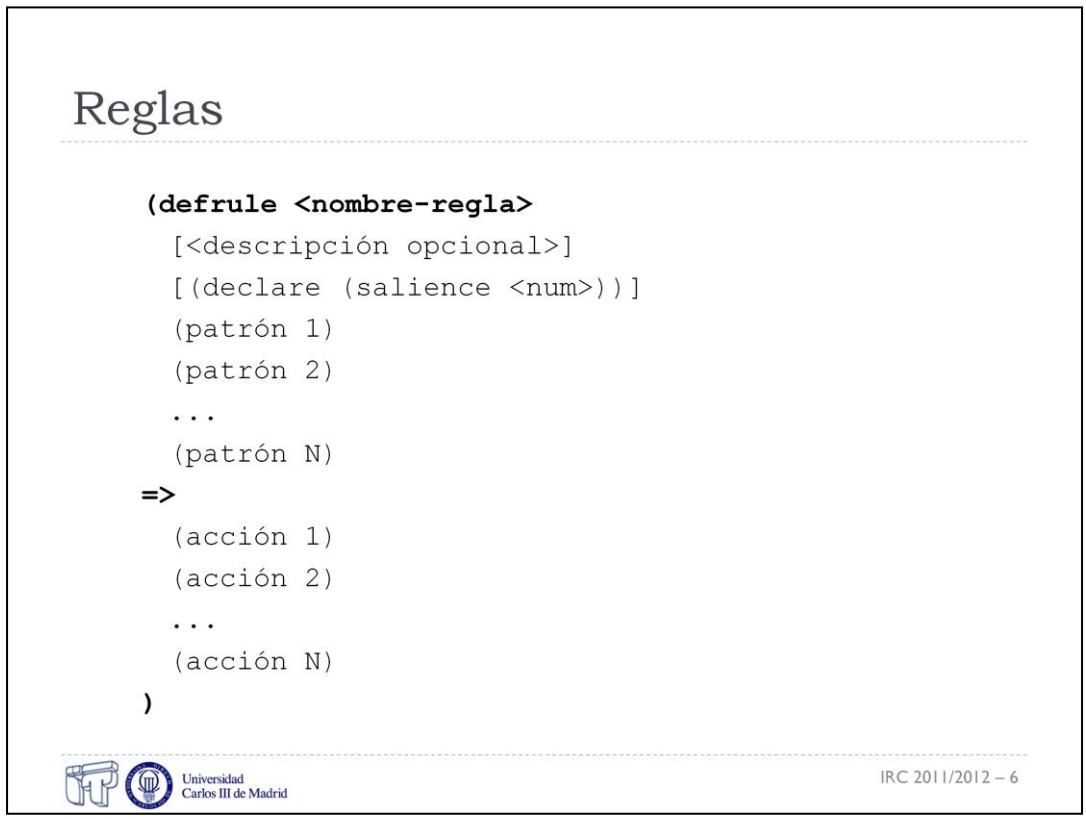
- La estructura *deffacts* se utiliza para agrupar conceptualmente diferentes hechos sobre el mismo objeto
- Ejemplo:

```
(deffacts hechos-del-vehiculo
  "información del vehículo"
  (marca Citroen)
  (modelo Xsara Picasso)
  (puertas 5)
)
```

Universidad Carlos III de Madrid

IRC 2011/2012 – 5

La estructura “deffacts” se utiliza para definir de forma conjunta una serie de hechos, habitualmente referidos a un mismo objeto, en la base de conocimientos.



Las reglas se definen con la estructura “defrule”: “si patrón_1 y patrón_2 y … patrón_N, entonces ejecuta la acción_1 y la acción_2 y … la acción_N”.

Ejemplo (hechos y reglas)

```
;;******************************
;;* EJEMPLO DE HECHOS Y REGLAS *
;;******************************

(deffacts nombres
  (nombre Pedro Perez Perez)
  (nombre Luis Montes Garcia)
  (nombre Carlos Marin Rodriguez)
  (nombre Juan Soler Leal)
)

(defrule apellidos-iguales
  (nombre ?x ?y ?y)
=>
  (printout t
  "D." ?x " tiene los dos apellidos
   iguales" crlf)
)

(defrule buenos-dias
  (nombre ? ?x ?)
=>
  (printout t
  "Buenos días, Sr." ?x crlf)
)

(defrule buenas-tardes
  (nombre $? ?x ?)
=>
  (printout t
  "Buenas tardes, Sr." ?x crlf)
)
```

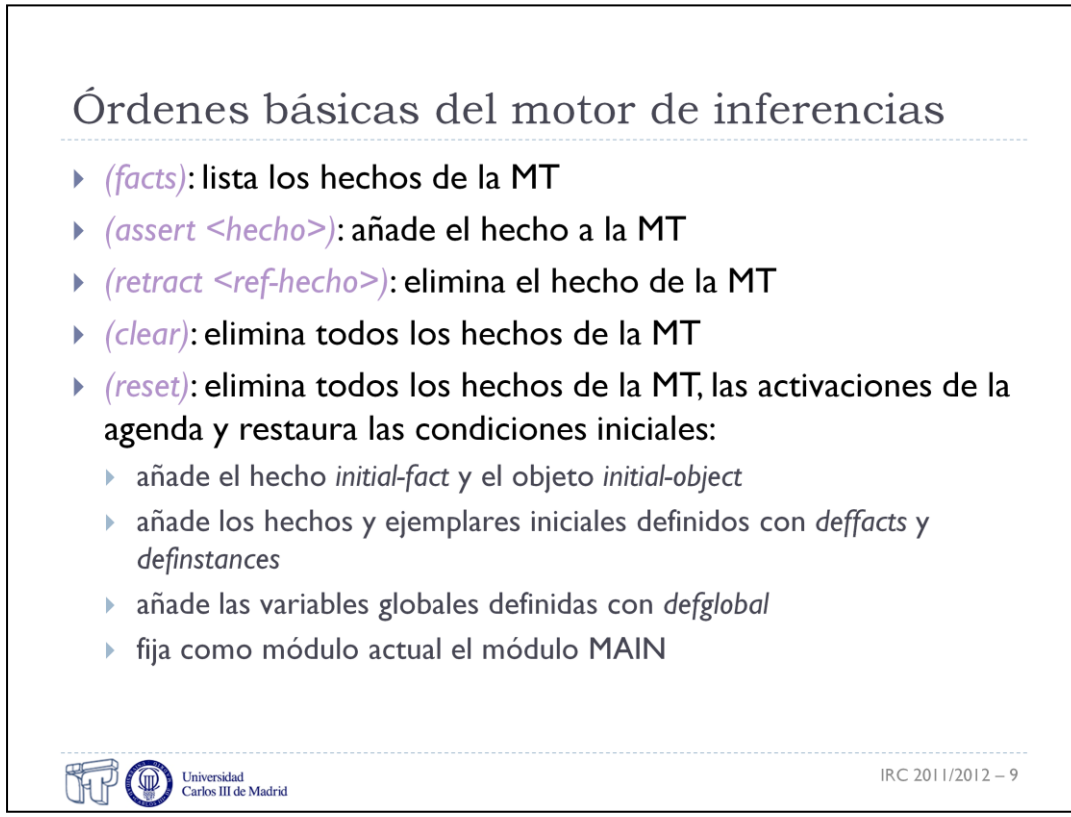
IRC 2011/2012 – 7

Ejemplo ilustrado de hechos y reglas.

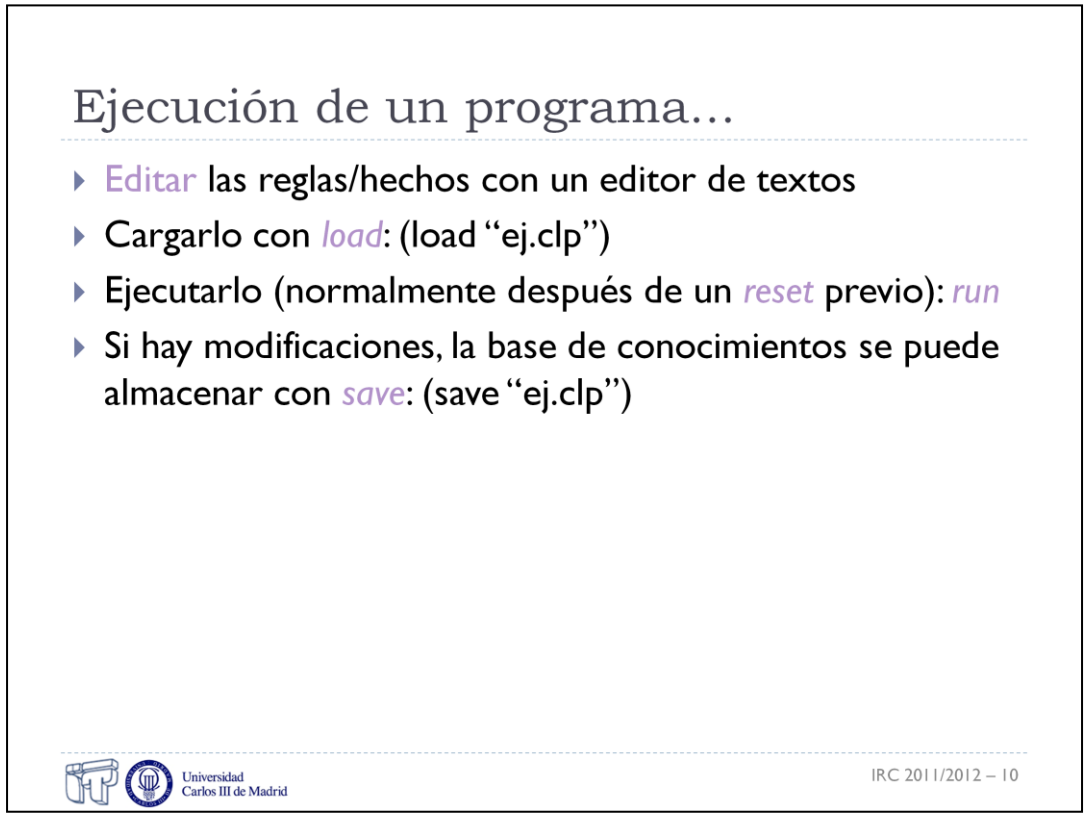
Motor de inferencias

- El motor de inferencias trata de emparejar la lista de hechos con los patrones de las reglas
- Si todos los patrones de una regla están emparejados se dice que dicha regla está activada
- La agenda almacena la lista de activaciones, por orden de prioridad
- Para insertar una activación en la agenda, se siguen las estrategias de resolución de conflictos

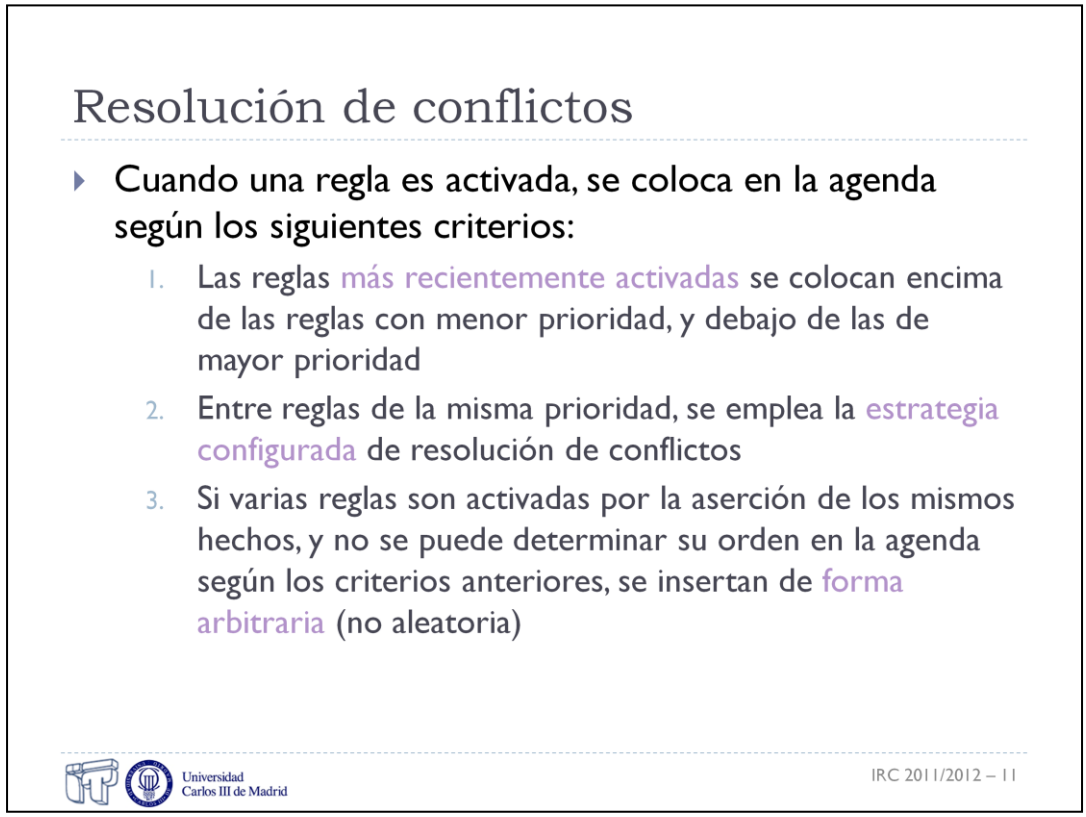
CLIPS tiene un motor de inferencias que continuamente trata de emparejar la lista de hechos de la memoria de trabajo (que es dinámica, pudiendo insertar/extraer hechos de ella en cualquier instante de tiempo), con los patrones de las reglas. Si todos los patrones de una regla existen en la memoria de trabajo, se dice que dicha regla está activada. Si hay más de una regla activada, se decide cuál se ejecuta según una estrategia de resolución de conflictos. La lista de reglas activadas se denomina “agenda”.



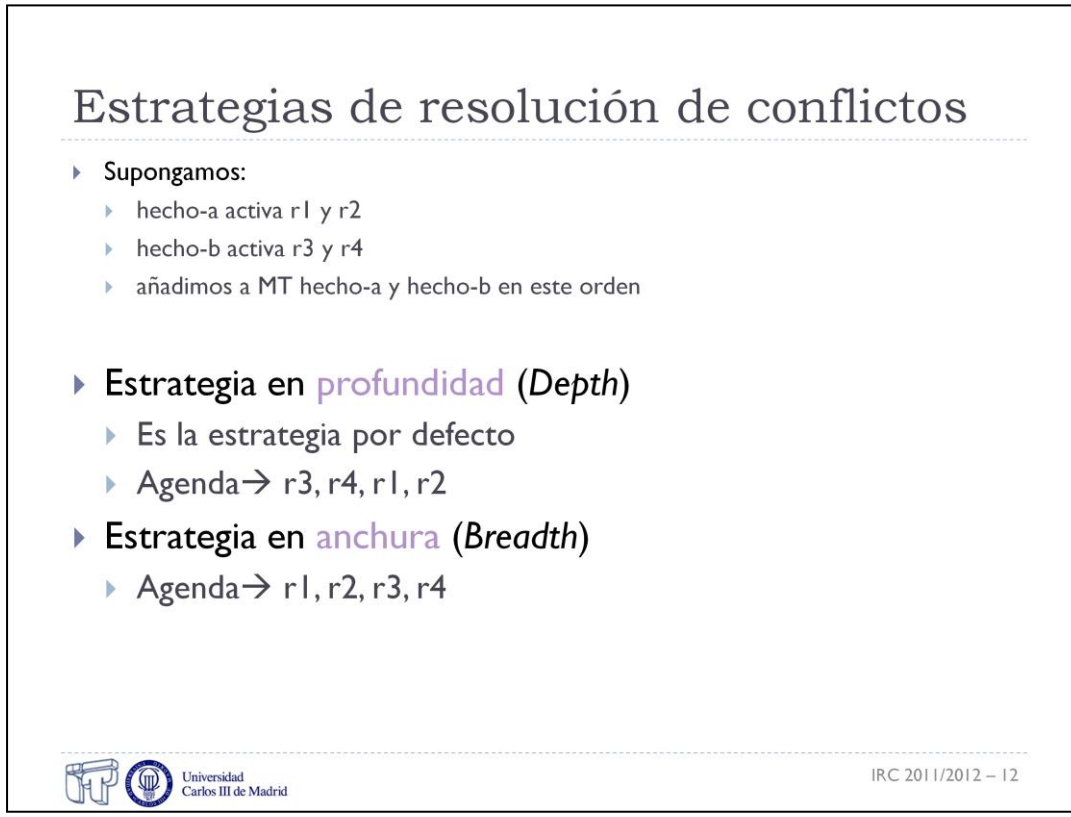
Esta es una lista resumida de las órdenes básicas para el motor de inferencias. “MT” significa “memoria de trabajo”, el estado de la base de conocimientos durante una determinada ejecución del programa CLIPS.



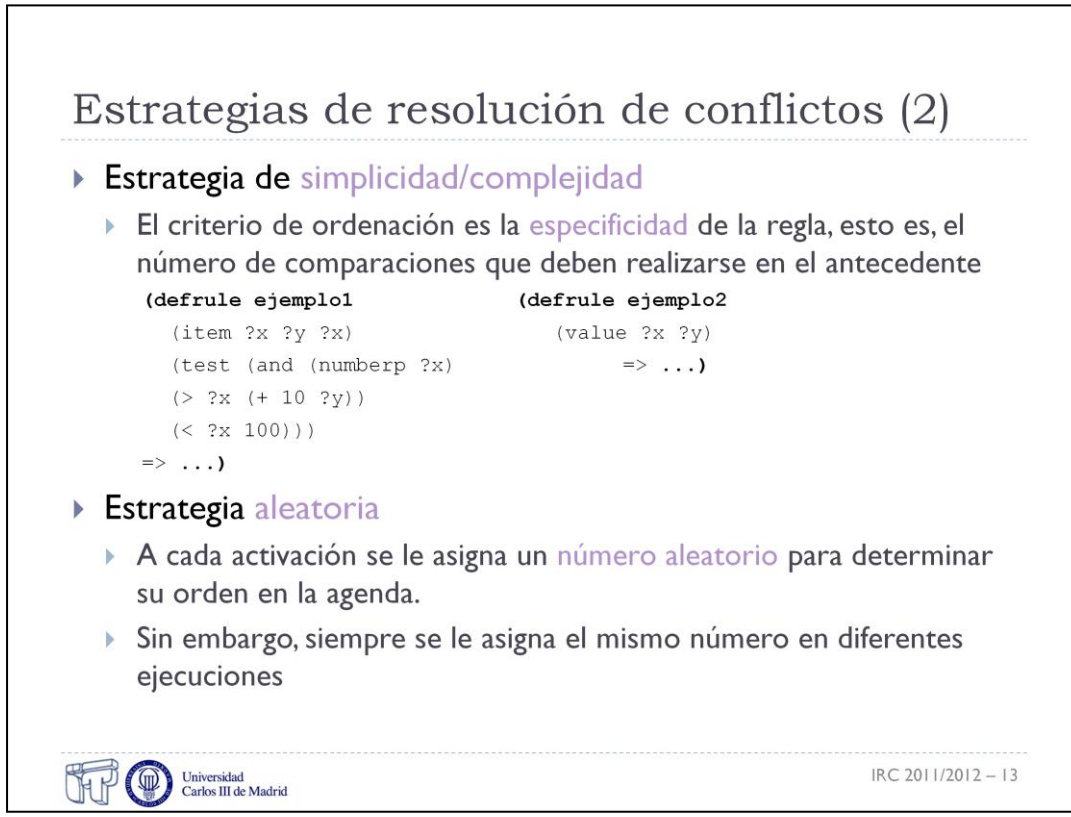
La ejecución habitual de un programa CLIPS es editar las reglas y hechos con un editor de textos, cargar dicho programa en CLIPS y ejecutar "run".



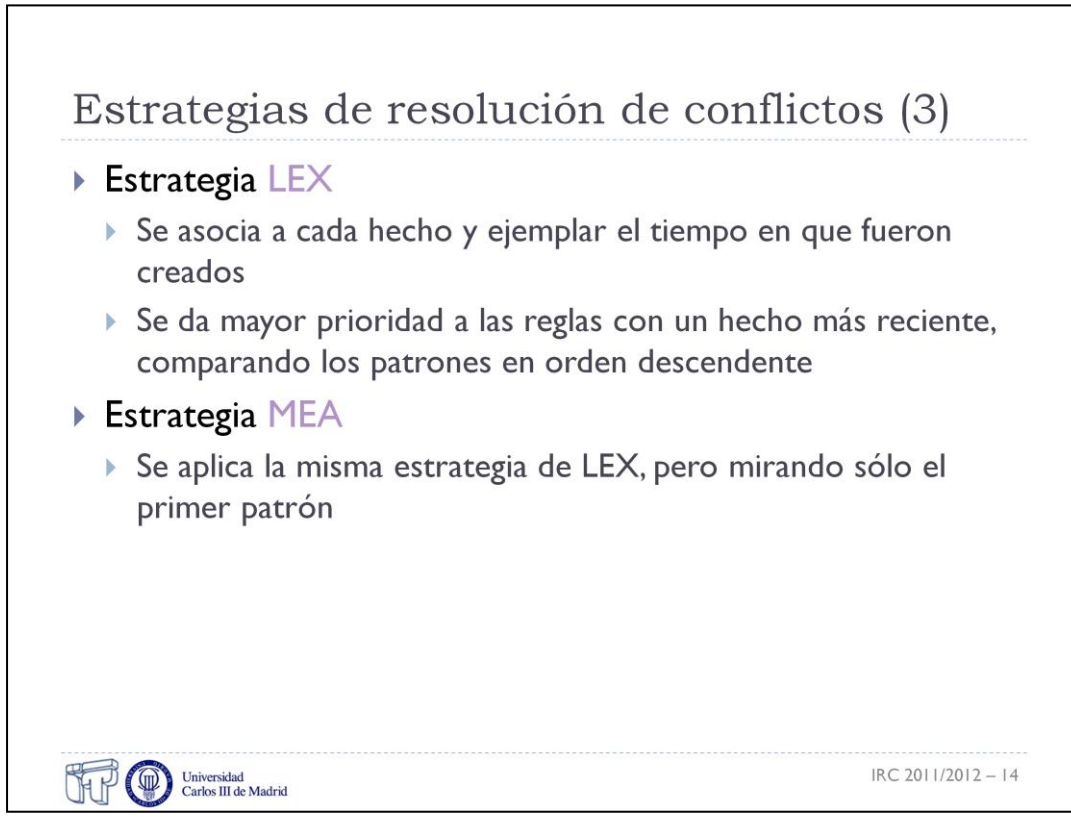
CLIPS tiene diferentes estrategias de resolución de conflictos para decidir en qué orden se ejecutan reglas que están activadas simultáneamente.



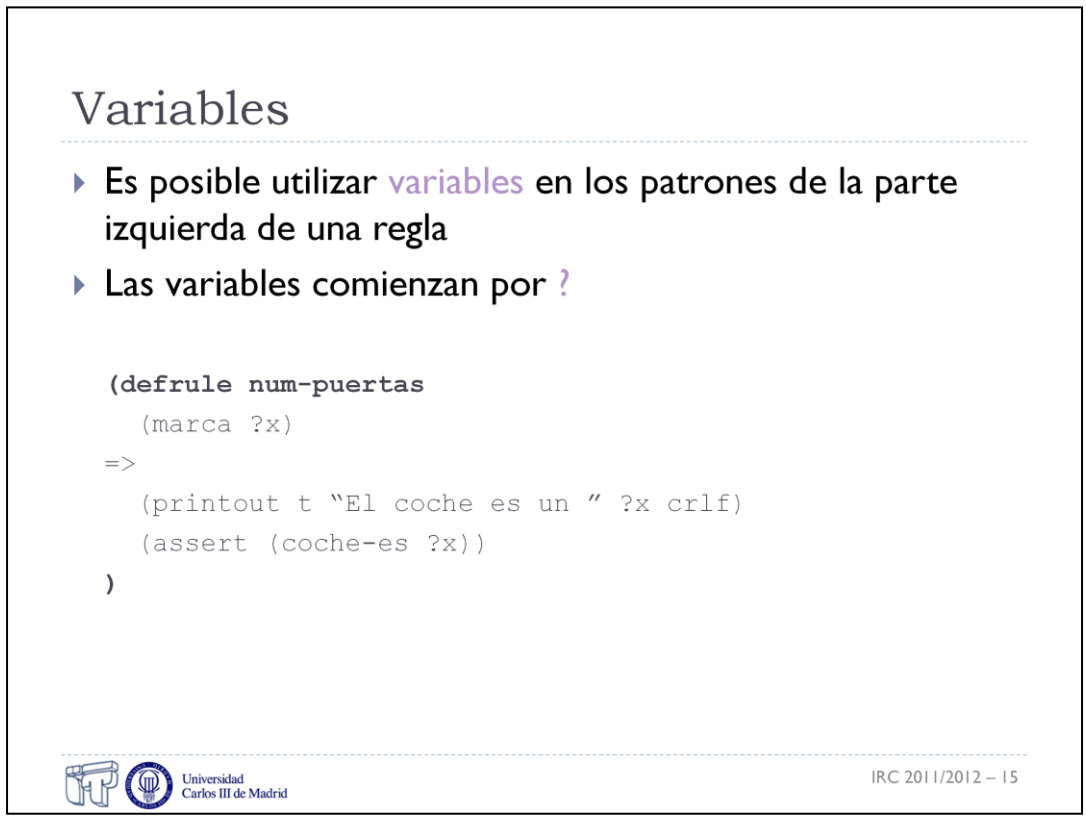
Estrategias de resolución de conflictos en profundidad y en anchura.



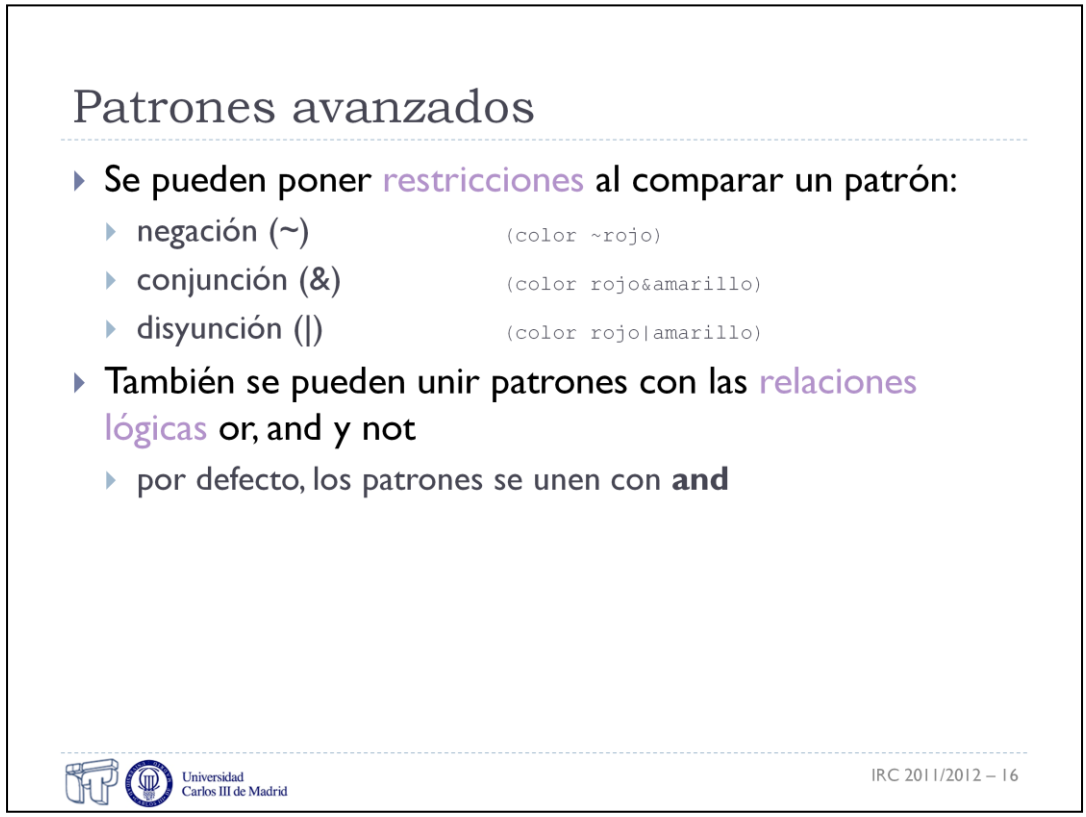
Estrategias de resolución de conflictos por simplicidad/complejidad y estrategia aleatoria.



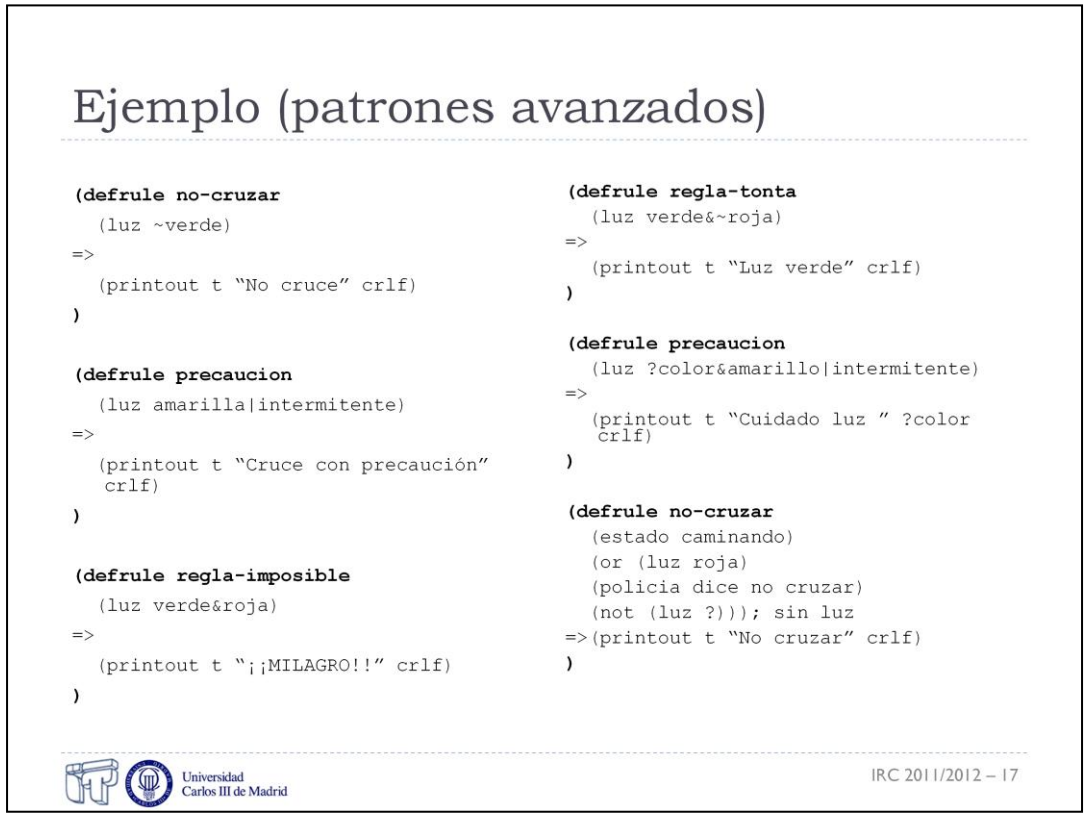
Estrategias de resolución de conflictos LEX y MEA.



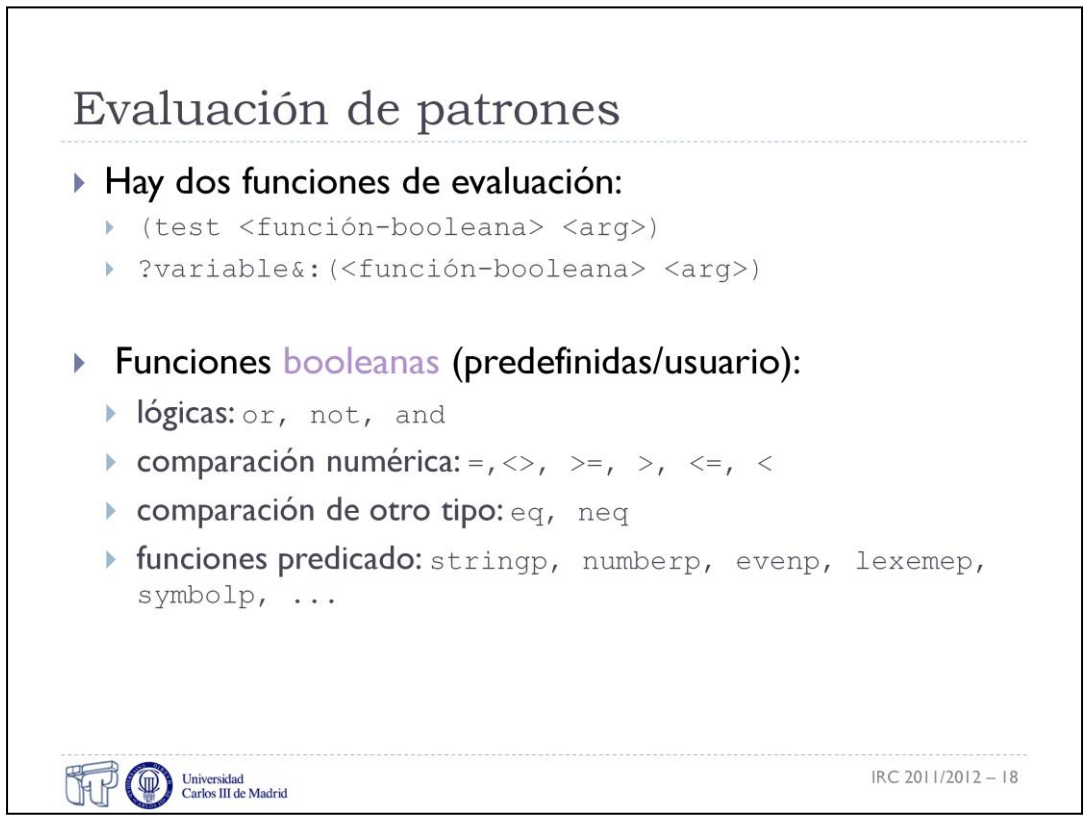
Ejemplo de utilización de variables (que empiezan por ?) en los patrones de la parte izquierda de una regla.



Los patrones avanzados hacen uso de operadores como NOT, AND y OR.



Ejemplo de patrones avanzados.



En los patrones se pueden utilizar diferentes funciones de evaluación y funciones booleanas de comparación numérica, de cadenas, de tipos, etc.

Ejemplo (evaluación de patrones)

```
(defrule mes-valido-1
  (entrada ?numero)
  (test (and (>= ?numero 1) (<= ?numero 12)))
=>
  (printout t "Mes válido" crlf)
)

(defrule mes-valido-2
  (entrada ?numero&:(and (>= ?numero 1) (<= ?numero 12)))
=>
  (printout t "Mes válido" crlf)
)
```

IRC 2011/2012 – 19

Ejemplo de reglas con funciones de evaluación de patrones.

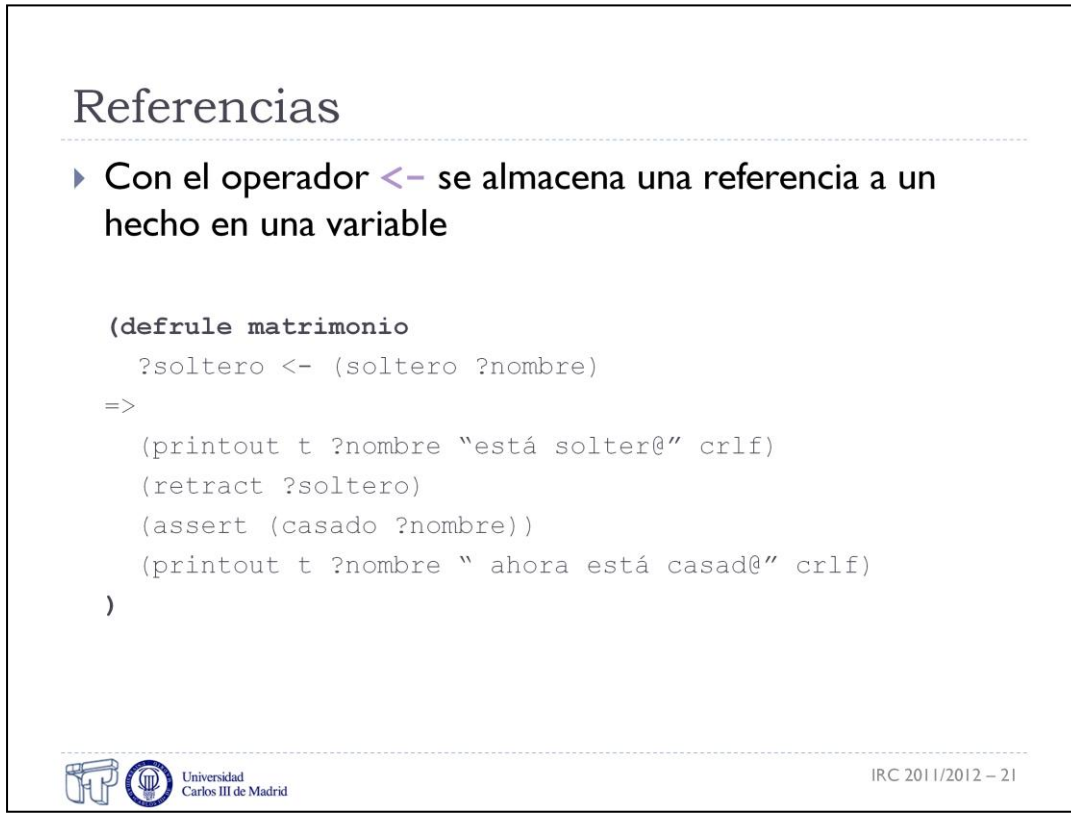
Variables globales *(defglobal)*

- Permiten almacenar valores accesibles en reglas y funciones, muy útiles para resultados

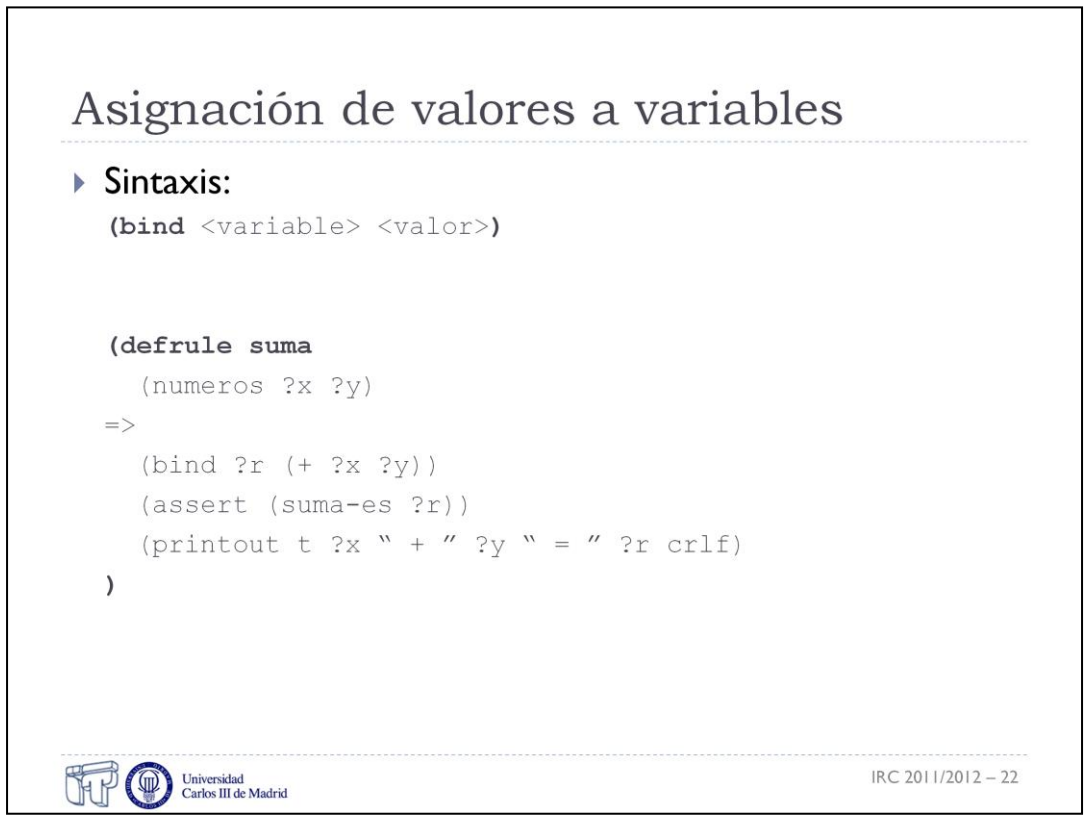
```
(defglobal
  ?*num* = 3
  ?*suma* = (+ ?*num* 2)
  ?*cadena* = "hola"
  ?*lista* = (create$ a b c)
)
```

- Pueden aparecer en la parte izquierda de las reglas si no son utilizadas para asignar un valor y su cambio no activa las reglas, pero no pueden ser parámetros de funciones ni métodos

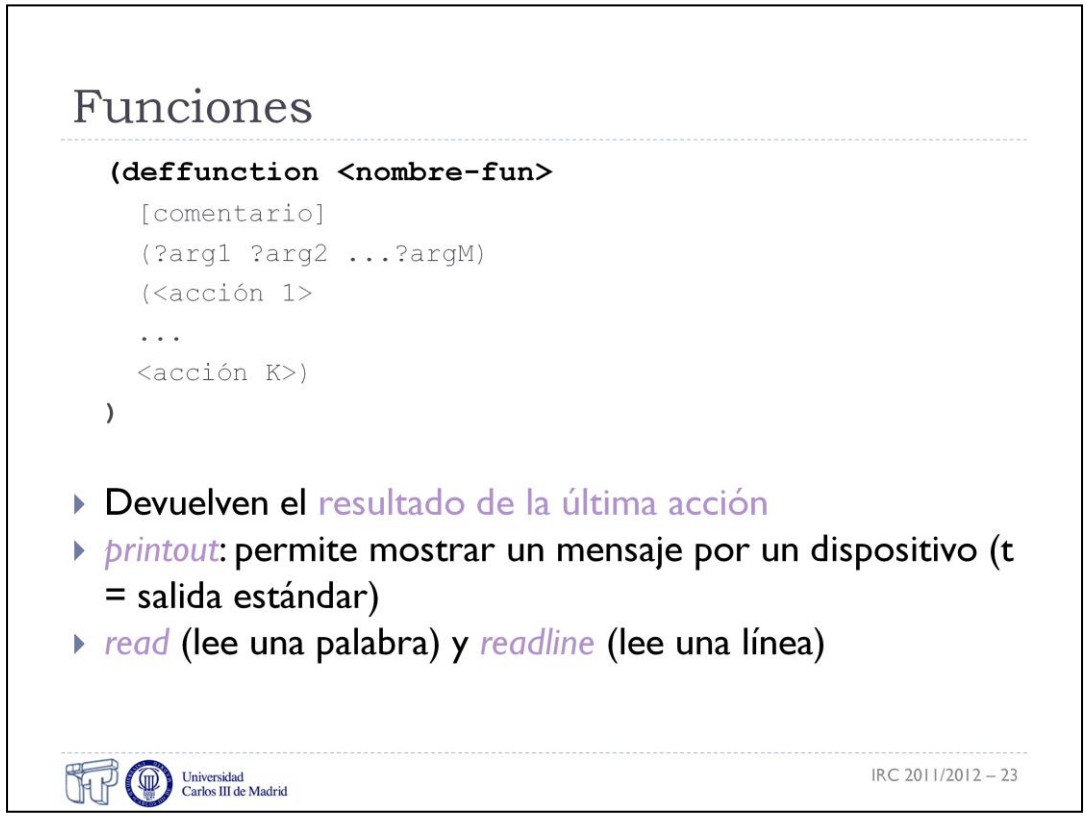
La estructura "defglobal" permite almacenar constantes/funciones globales.



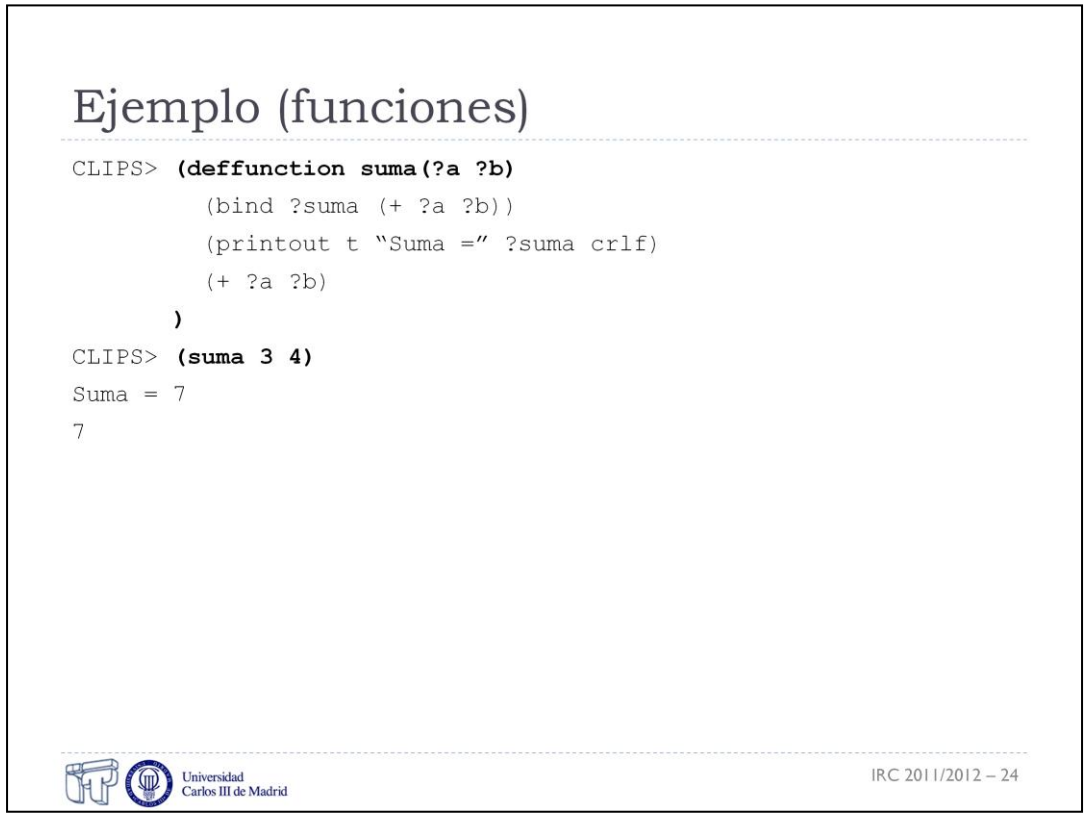
El operador "<-" sirve para almacenar una referencia a un hecho en una variable, para utilizarla más adelante.



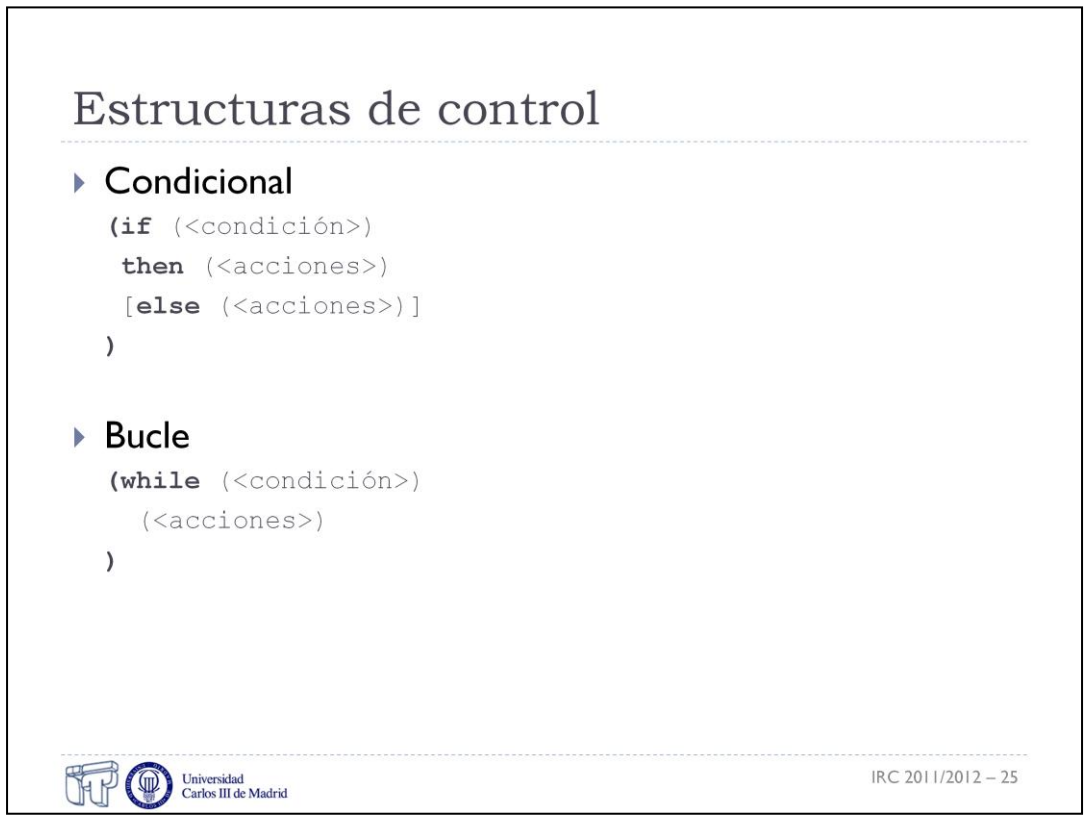
La función “bind” se utiliza para asignar valores a variables, habitualmente con una expresión matemática o lógica.



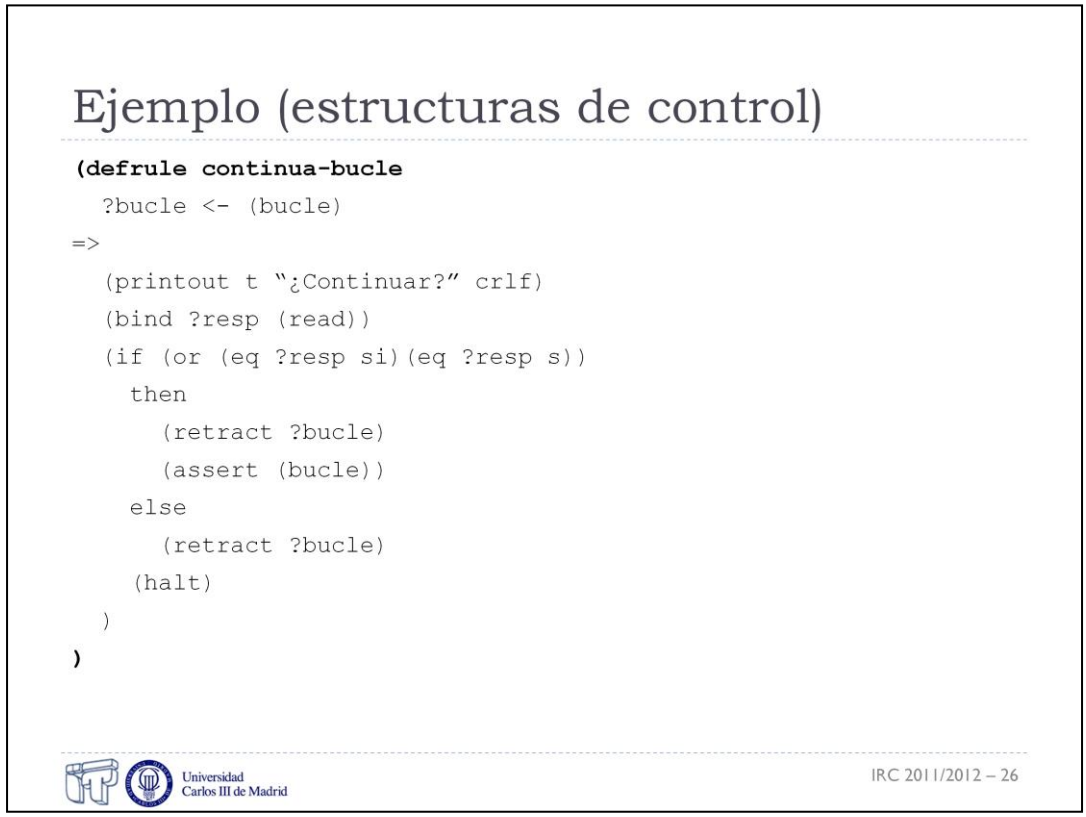
Las funciones se define con la estructura “deffunction”, y se componen de un patrón base con variables y una serie de acciones a ejecutar.



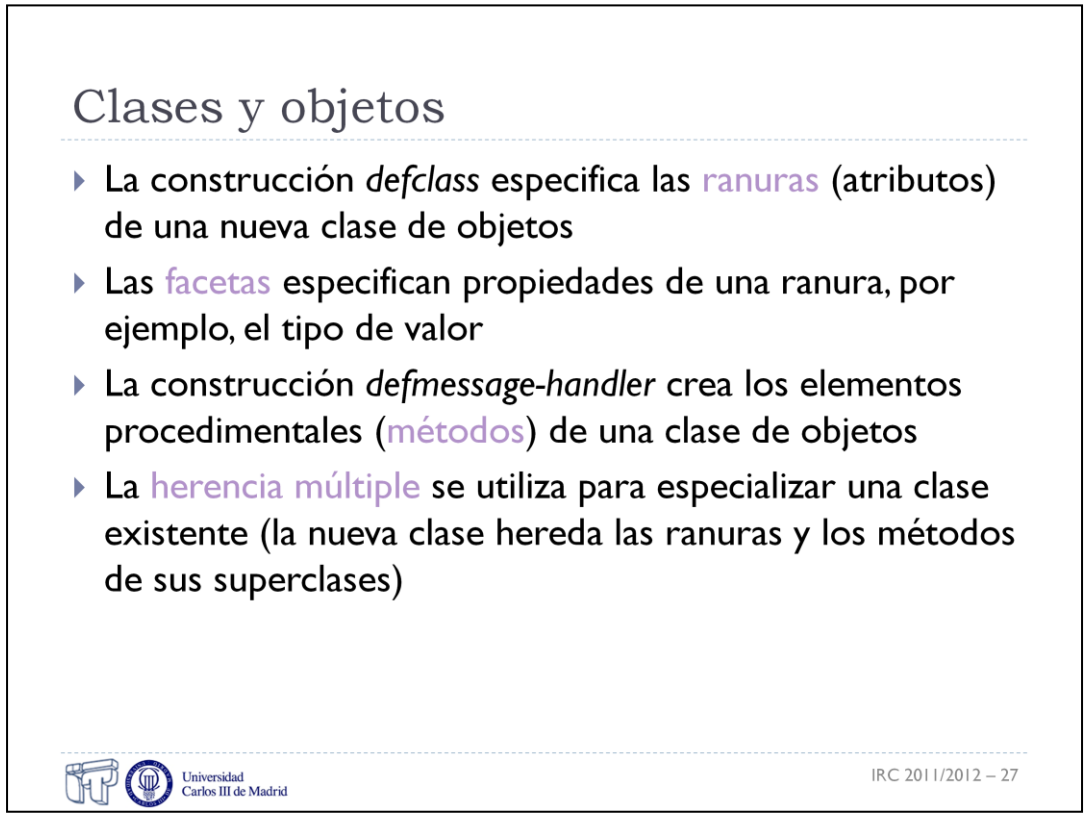
Ejemplo de definición y llamada a la función “suma” en CLIPS.



En CLIPS además existen las estructuras clásicas de control con “if” y bucles con “while”.



Ejemplo de utilización de estructuras de control en CLIPS.



CLIPS soporta un conocimiento orientado a objetos, definiendo clases con “defclass”. Los atributos se denominan “ranuras” (slots), las “facetas” son los valores aceptados en cada ranura, y los “message-handler” serían equivalente a los métodos de acceso a dichos atributos.

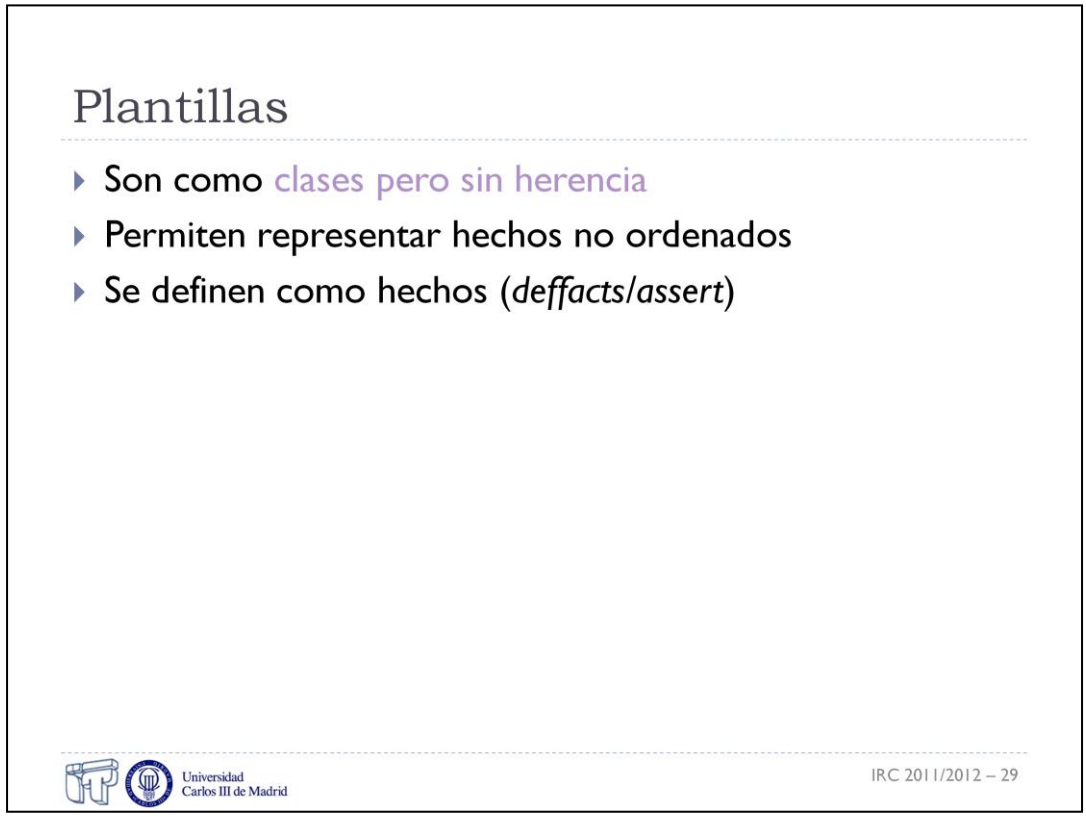
Ejemplo (clases)

```
(defclass perro
  (is-a USER)
  (role concrete)
  (pattern-match reactive)
  (slot patas (type INTEGER)
    (default 4)(range 0 4)
    (create-accessor read-write))
  (slot raza (type SYMBOL)
    (allowed-symbols caniche dogo)
    (default caniche)
    (create-accessor read-write))
  (multislot dueno (type STRING)
    (cardinality 1 2)
    (default "Juan")
    (create-accessor read-write))
  (slot ident(type SYMBOL)
    (default-dynamic (gensym)))
)
```

```
CLIPS> (make-instance Lulu of perro)
[Lulu]
CLIPS> (send [Lulu] print)
[Lulu] of perro
(patas 4)
(raza caniche)
(dueno "Juan")
(ident gen1)
CLIPS> (send [Milu] get-patas)
4
CLIPS> (send [Milu] put-patas 3)
3
```

IRC 2011/2012 – 28

Ejemplo de definición de clases en CLIPS.



Las plantillas en CLIPS son como clases pero sin herencia, y permiten representar hechos no ordenados.

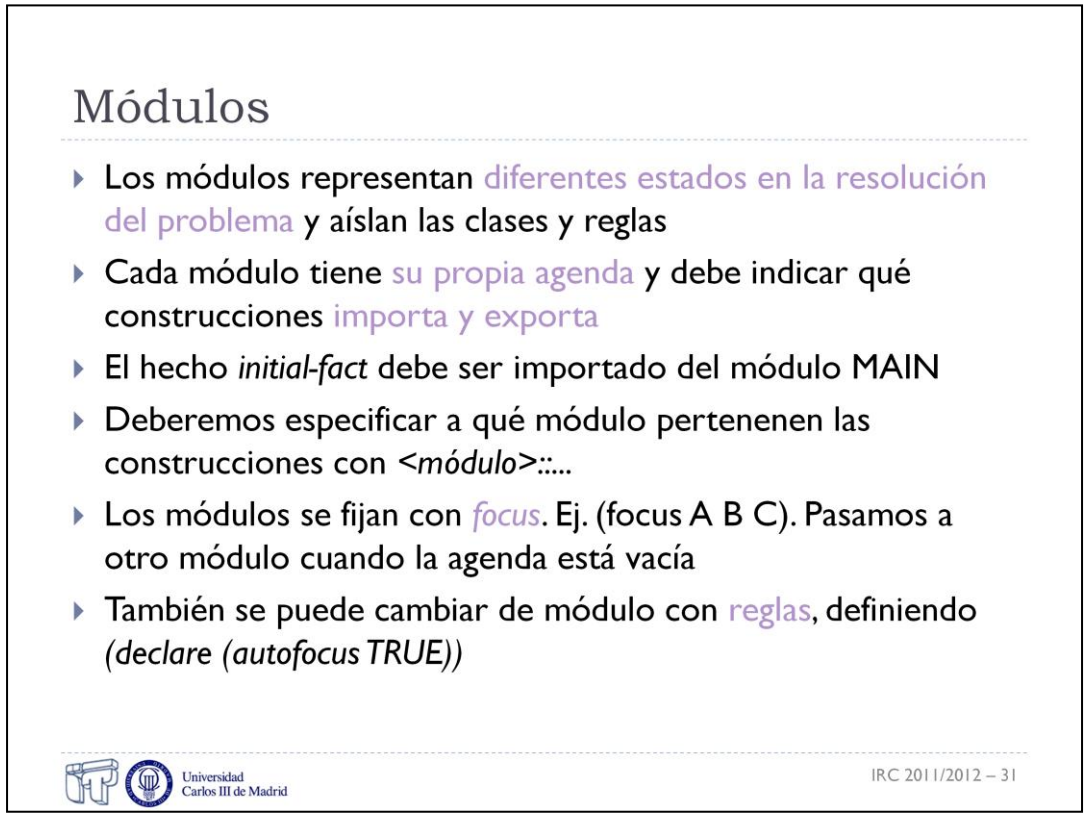
Ejemplo (plantillas)

```
(deftemplate persona
  (slot nombre (type SYMBOL))
  (slot edad (type NUMBER)
    (range 0 99)(default 20))
  (slot estado (type SYMBOL)
    (allowed-symbols soltero casado
   viudo)
    (default soltero))
)
(deffacts personas
  (persona (nombre Pepe))
  (persona (nombre Juan)(edad 25))
)
```

```
(defrule casa-mayores-25
  ?p <-(persona
    (nombre ?nombre)
    (edad ?edad)
    (estado soltero))
  (test (>= ?edad 25))
=>
  (modify ?p (estado casado))
  (printout t ?nombre " tiene "
  ?edad " anios" crlf)
)

CLIPS> (reset)
Juan tiene 25 anios
CLIPS> (facts)
f-0 (initial-fact) CF 1.00
f-1 (persona (nombre Pepe) (edad
20) (estado soltero)) CF 1.00
f-3 (persona (nombre Juan) (edad
25) (estado casado)) CF 1.00
```

Ejemplo de plantillas en CLIPS.



El código en CLIPS se puede estructurar en módulos, que sirven para encapsular de forma lógica el conocimiento almacenado y simplificar así el desarrollo de sistemas expertos complejos.

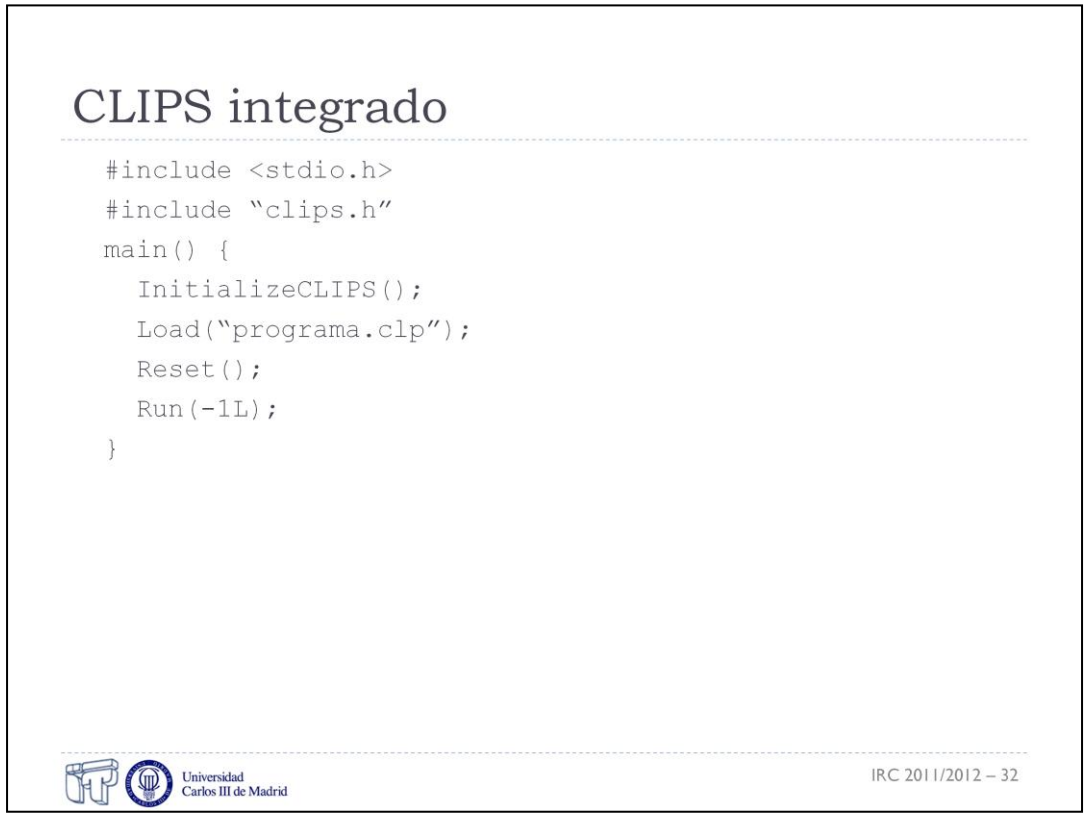
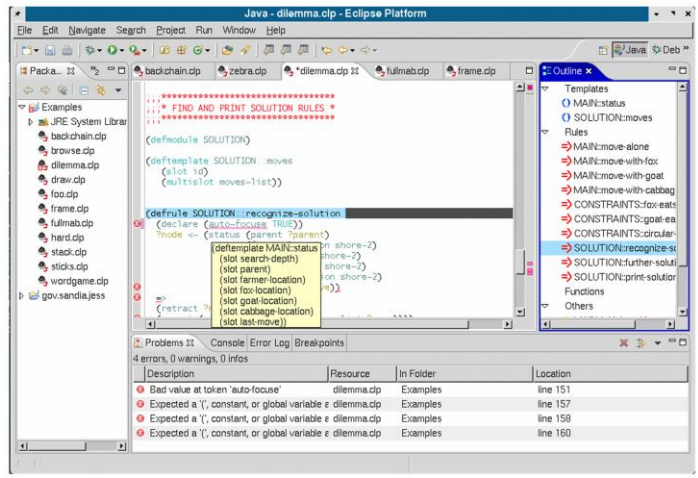


Ilustración de la integración de CLIPS en código C.

Jess *(Java Expert System Shell)*

- CLIPS reprogramado en Java, con ligeras variaciones, e integrado en el lenguaje



Universidad Carlos III de Madrid

IRC 2011/2012 – 33

Jess es la reimplementación de CLIPS en Java, renovado y mejorado unos cuantos años después de la implementación original.