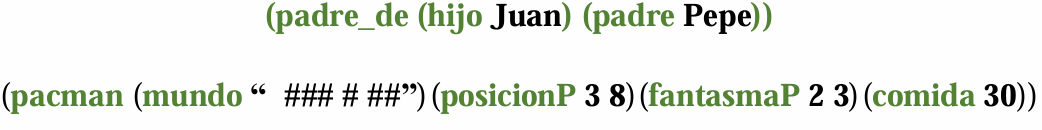
**Hechos**

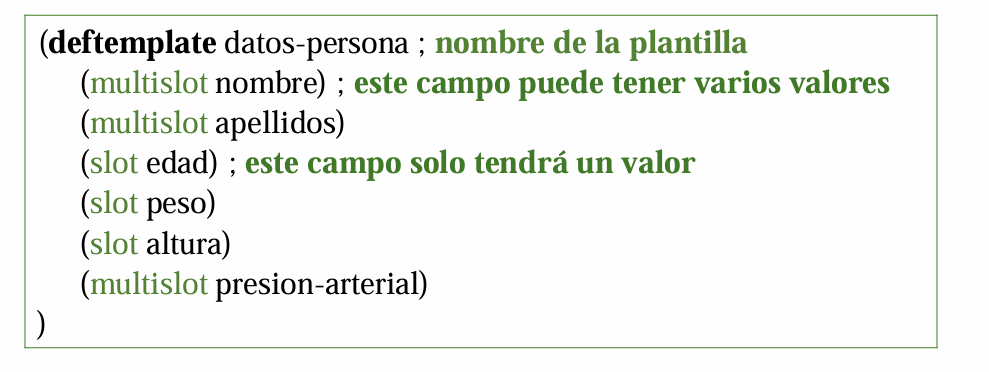
* Un hecho es una forma básica de representar la información. Puede tener un campo o varios,de tipo numerico,simbólico o String
* CLIPS diferencia entre mayúsculas y minúsculas
* Existen dos tipos de hechos **ordenados** y **no ordenados**

**Hechos no ordenados**

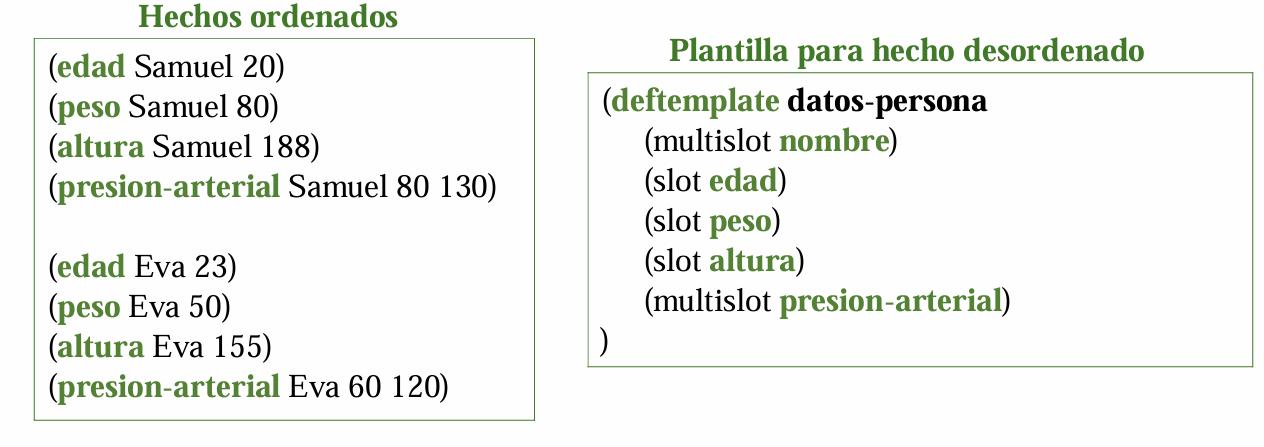
* En los hechos no ordenados cada campo tiene su nombre y valor

****

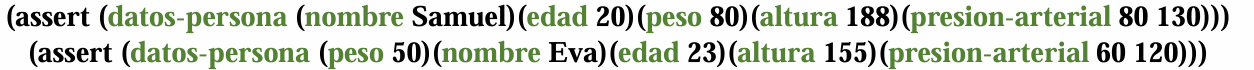
* Los hechos no ordenados **proporcionan al usuario la habilidad de abstraerse de la estructura del hecho**, asignando un nombre a cada campo del mismo
* Para los **hechos no ordenados** CLIPS **requiere la definición** previa de su **plantilla,**por medio del constructor **deftemplate**,para especificar el modelo del hecho
* El constructor **deftemplate** crea una plantilla o patron que se usa para acceder por su nombre a los campos (slots/multislots) de un hecho no ordenado

****

* **Comandos:**
  + (list-deftemplates) listamos los **nombres de las plantillas** definidas
  + (ppdeftemplate <nombre>) muestra el **contenido de la plantilla**
  + (undeftemplate <nombre>) permite eliminar la definición,siempre que no existan hechos en la base de hechos que sigan esa plantilla
* Los hechos ordenados permiten juntar trozos de información.Un ejemplo podría ser el estado de forma de una persona

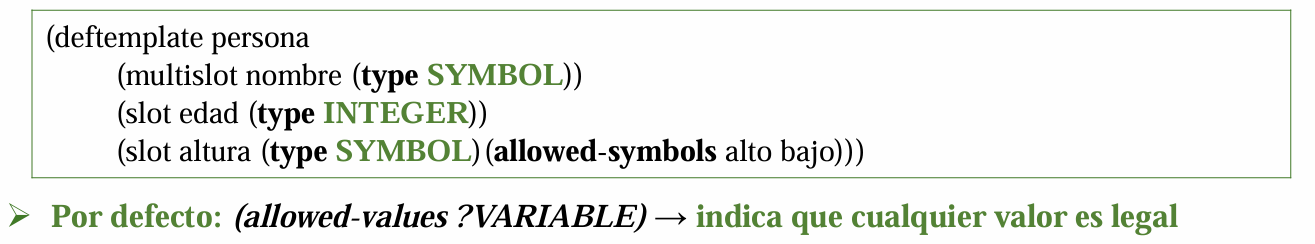
****

* Cuando hay muchos datos relacionados, es mas cómodo utilizar la plantilla

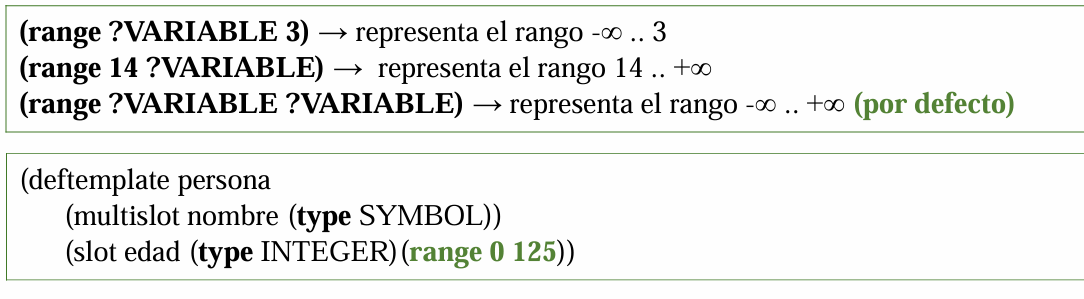
****

**Hechos no ordenados atributos de las plantillas**

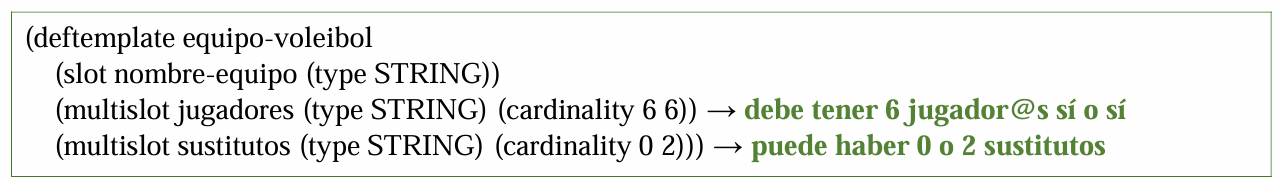
* Una plantilla **permite definir** además del nombre, el tipo, los valores por defecto y el rango de sus slots o multislots
* Los tipos posibles son **SYMBOL,STRING,LEXEME,NUMBER,INTEGER** y **FLOAT**
* Para definir el tipo de dato **que admite** un slot/multislot se usa type
  + **LEXEME** Equivale a especificar SYMBOL y STRING conjuntamente
  + **NUMBER** equivale a especificar INTEGER y FLOAT
  + **?VARIABLE** puede tomar cualquier tipo de dato(por defecto)
* **Allowed** Especifica los valores concretos permitidos allowed-symbols,allowed-strings,allowed-lexemes,allowed-integers,allowed-floats,allowed-numbers y allowed-values

****

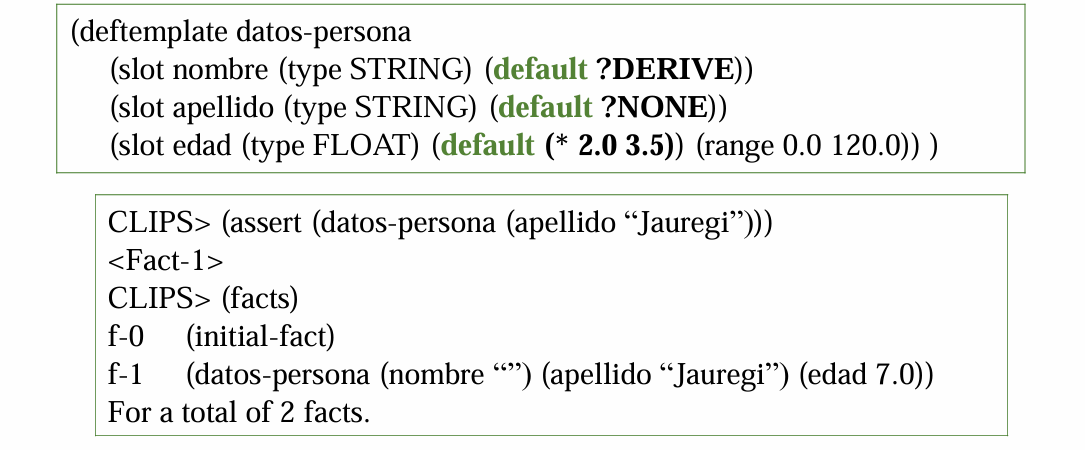
* **Range** permite restringir los valores de un tipo **numerico** a un rango determinado.Los limites pueden ser un valor numerico o ?variable (infinito)

****

* **Cardinality** permite especificar el numero mínimo y máximo de valores que un multislot puede contener.Ambos limites pueden ser entero positivo o ?variable.Por defecto se supone ?variable para ambos limites



* **Default** permite especificar un valor por defecto (defualt <especificacion>)
* La <especificacion> puede ser **?DERIVE,?NONE** o una **expresión(valor):**
  + **?DERIVE** Se deriva un valor para el slot que satisfaga todos los atributos del slot.Si en un slot no se especifica nada acerca de default se supondrá default ?DERIVE
    - Para **String -> “”,Integer->0,Symbol-> nil,float -> 0.0**
  + **?NONE** hay que darle un valor obligatoriamente

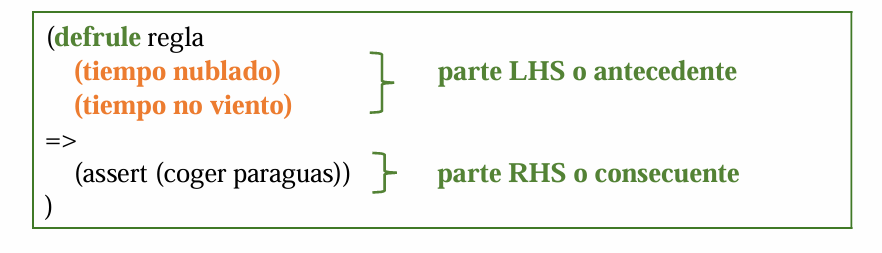


**Hechos operaciones**

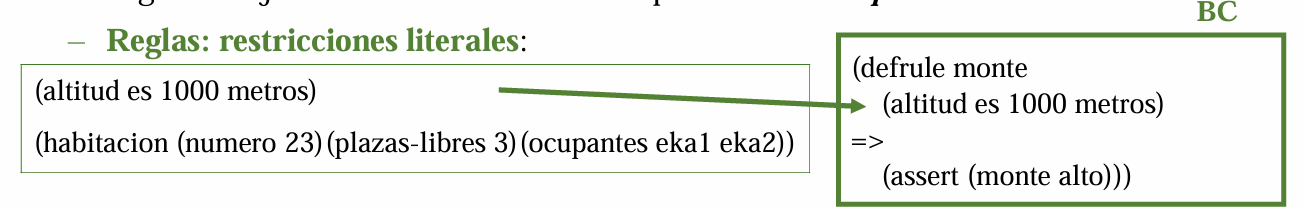
* Las acciones que se pueden realizar sobre los hechos
  + Acciones que modifican la MT Lista de hechos
    - Para insertar hechos
      * (assert <hecho>)
      * (duplicate <especificador-hecho><slot>\*)
        + Solo con hechos no ordenados
    - Para borrar hechos
      * (retract<especificador-hecho>)
        + Puede utilizarse el símbolo \* con el comando retract para eliminar todos los hechos
    - Donde <especificador-hecho> puede ser
      * Una variable previamente ligada a la dirección del hecho a duplicar, borrar o modificar
      * Un índice de hecho

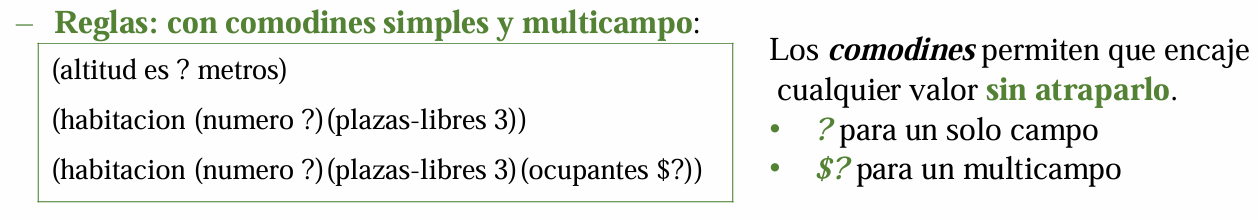
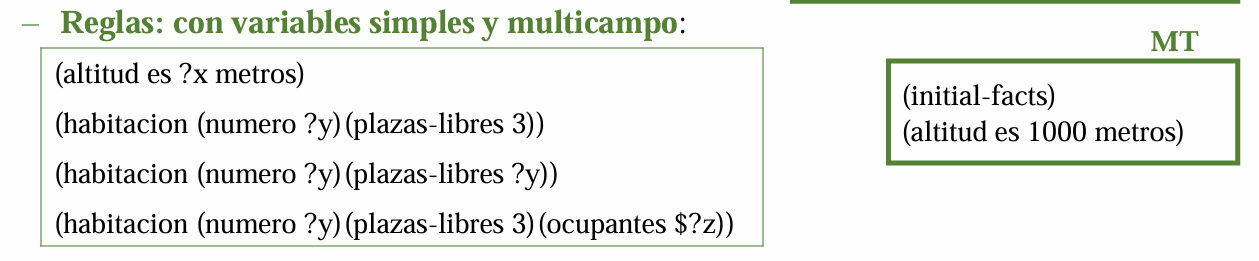
**Reglas**

* Las reglas permiten operar con los hechos
* Una regla consta de un **antecedente** también denominado parte si o parte izquierda de la regla (LHS) y de un **consecuente** también denominado parte entonces o parte derecha de la regla (RHS)
* El **antecedente** esta formado por un conjunto de condiciones -tambien denominadas **elementos condicionales (EC)** que deben satisfacerse para que la regla se aplicable
* El consecuente de una regla es un conjunto de acciones a ser ejecutadas cuando la regla sea aplicable

****

* **Pattern** Colección de restricciones de campos,comodines y variables que se usan para restringir el conjunto de hechos o instancias que satisfacen el pattern

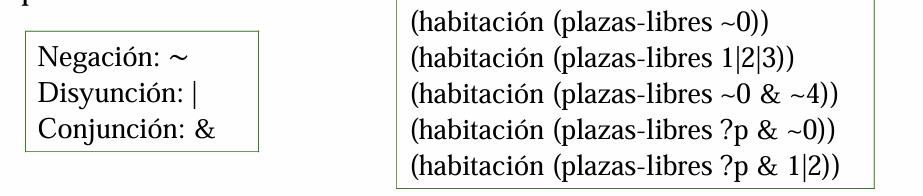


****

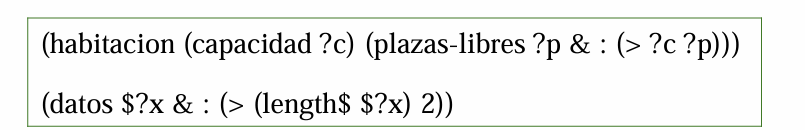


**Tipos de elementos condicionales(LHS)**

* Se pueden especificar restricciones al comparar un patron (operadores logicos).El orden de prioridad es el presentado a continuacion

****

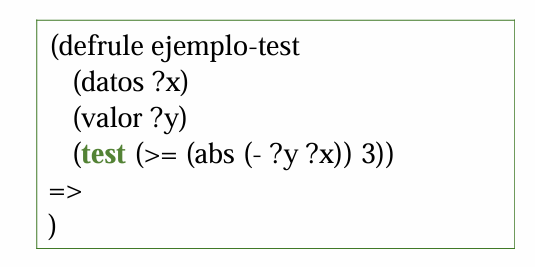
* Con predicados o llamadas a funciones.En este caso deben ir predicados del signo “:”

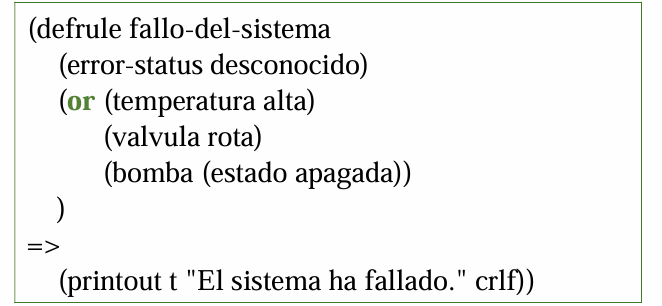
****

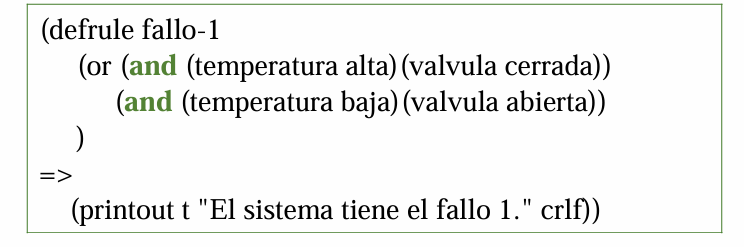
* Predicados
  + De tipo (todos terminan en p): numberp,floatp,symbolp,…
  + Comparaciones numéricas: =,<>,<,<=,>,>=
  + Igualdad (desigualdad) en tipo y valor eq (noeq)
  + Predicados definidos por el usuario

**Tipos de elementos condicionales**

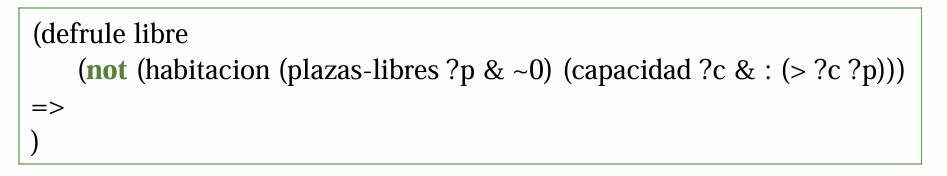
* **Test** Se usa para evaluar expresiones en la parte izquierda de una regla ,interviniendo en el proceso de pattern-matching.El elemento condicional test se satisface si la llamada a la función que aparezca dentro de el devuelve cualquier valor distinto de FALSE

****

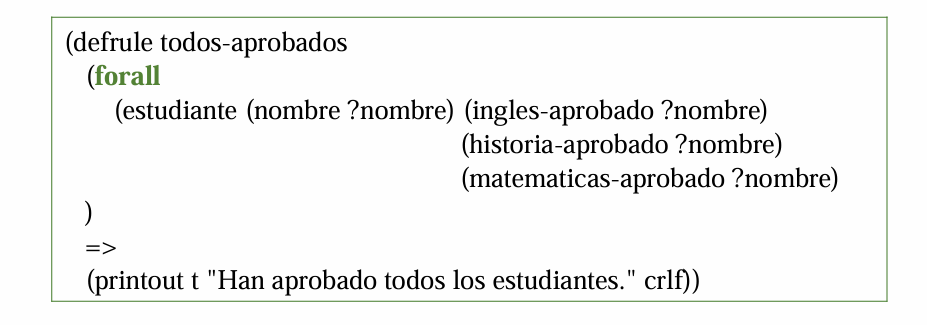
* **Or**  Este EC se satisface cuando al menos uno de los componentes que aparece se satisface
* **And** CLIPS supone que todas las reglas tienen un and implicito que rodea todos los elementos condicionales de la LHS

****

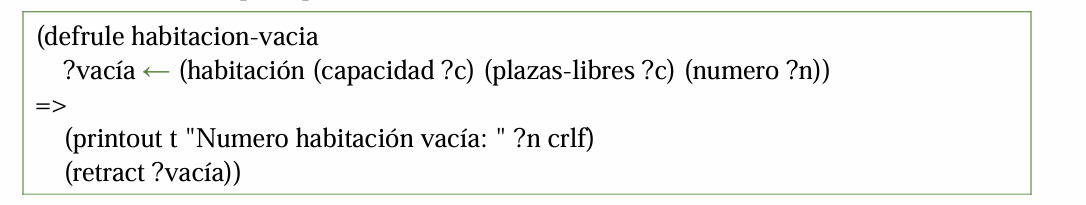
* **Not** Un elemento condicional negativo se satisface si no existe ninguna entidad que cumpla con las restricciones expresadas

****

* **Exists**  Este EC permite que se produzca el pattern matching cuando al menos exista un hecho que satisfaga la regla,sin tener en cuenta el numero total de hechos que pudiesen matchear.Esto permite una sola activación para una regla con la que matchean un conjunto de hechos
* **Forall** Permite el matching basado en un conjunto de EC’s que son satisfechos por cada ocurrencia de otro EC.Su funcionamiento es el contrario que el de exists.Para que el EC forall se satisfaga,todo lo quqe matchee con <primer EC> debe tener hechos que matcheen todos los demás EC que aparezcan a continuación de <primer EC>

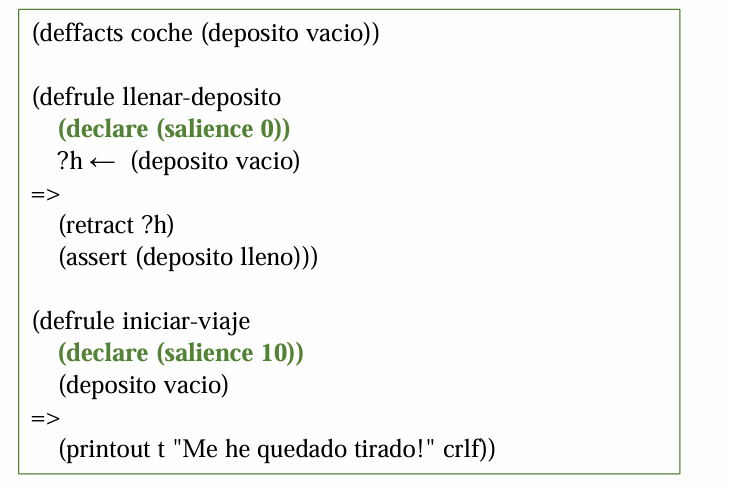
****

* **<-** Ligan la dirección de las entidades de la MT (hechos o instancias) que satisfacen el elemento condicional a una variable para poder realizar acciones sobre ellos

****

**Propiedades de las reglas**

* Las propiedades se declaran en la parte izquierda de la regla utilizando la palabra clave declare.Una regla solo puede tener una sentencia declare y debe aparecer antes del primer elmento condicional
* **Salience** Permite al usuario asignar una prioridad a una regla.La prioridad se asigna en ele rango de -10000 … 10000.Por defecto,la prioridad de una regla es 0
  + Si la agenda incluye varias reglas se desencadenara en primer lugar aquella que tenga mayor prioridad



Atrapar los hechos

* Los hechos se pueden atrapar desde la condición de la regla,o do-for-fact /do-for-all-facts

