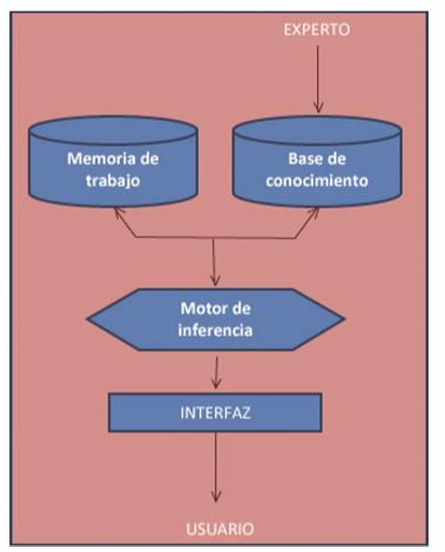
**Sistemas expertos**

* Alrededor de los años 70s se empezaron a crear sistemas expertos
* Estos “Sistemas Expertos” se llaman así por la pretensión de simular el comportamiento de un humano experto,en un área muy técnica y especifica.Tambien conocidos como Sistemas Basados en el Conocimiento (SBC)
* Todo problema es más sencillo de resolver si disponemos de conocimiento especifico sobre el
* Para operar con el conocimiento es necesario representarlo

**Representación y razonamiento**

* La mayor parte de los SBC están constituidos fundamentalmente por tres componentes
  + **Memoria del trabajo** Almacena información sobre el problema (Representación)
  + **Base del conocimiento** Almacena el conocimiento sobre el problema (Acciones)
  + **Motor de inferencia** Contiene información relativa al funcionamiento del sistema

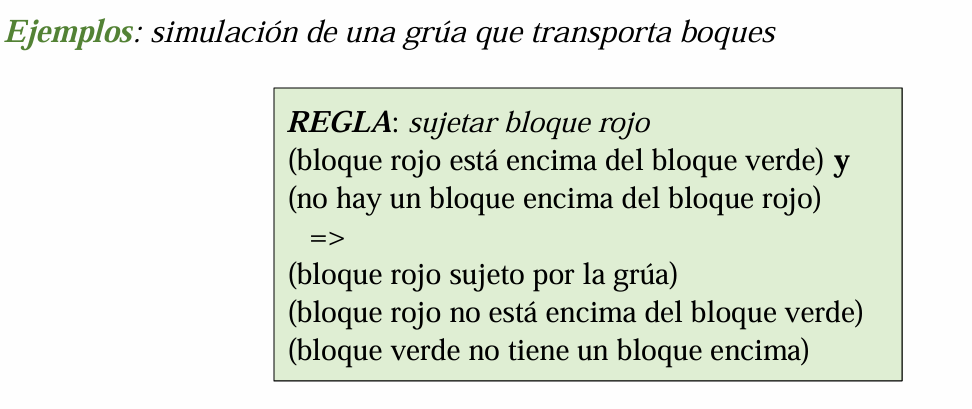
****

**Memoria de trabajo (representación)**

* También llamada base de hechos (BH)
* Contiene hechos sobre el mundo (representación del estado),que pueden observarse directamente o deducirse a través de las reglas
  + Se define mediante variables simples y variables multicampo
* Los hechos se pueden modificar mediante reglas, incluso añadir o eliminar

**Base de conocimiento (Acciones)**

* También llamada Base de Reglas (BR)
* Cada regla representa un paso de la resolución del problema. Se utilizan para representar las acciones del problema
* Una regla es un conjunto de condiciones junto con las operaciones a realizar si las condiciones se satisfacen
* Las reglas son conocimiento duradero, persistente, sobre el dominio. Una vez definidas, no se modifican, solo el experto del dominio puede modificarlas

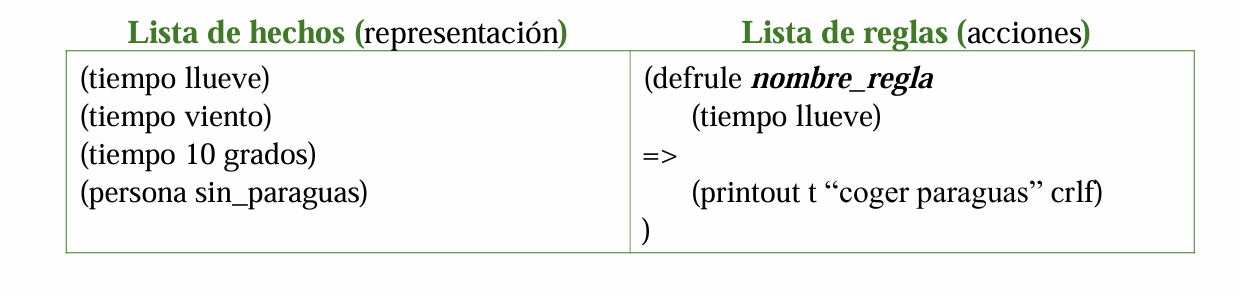


**Motor de inferencia**

* Procesa la información de la BH y BR
* Es el mecanismo de razonamiento
* Selecciona una regla de la BR y la aplica

**CLIPS hechos y reglas**

* Las reglas son como sentencias IF-THEN de lenguajes como ADA,C,JAVA
* CLIPS mantiene una lista de hechos y reglas, permitiendo estas operar con los hechos almacenados en la lista de hechos,dado que los hechos son necesarios para disparar p activar las reglas



**Hechos**

* Un hecho es una forma básica de representar la información. Puede tener un campo o varios, de tipo numérico, simbólico o Sting
* CLIPS diferencia entre mayúsculas y minúsculas
* Existen dos tipos de hechos **ordenados** y **no ordenados**
  + En un hecho al principio se pone el nombre\_del\_hecho
  + En los **hechos ordenados**, es importante el orden de los elementos o símbolos

**Hechos operaciones**

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

**Insertando hechos**

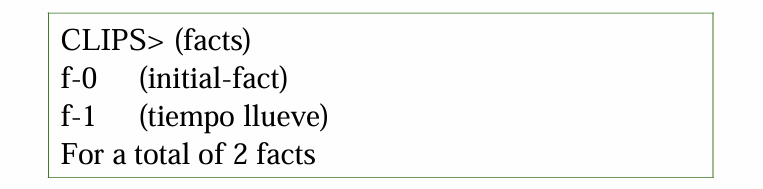
* El comando **(assert)** añade un hecho.Cada hecho se identificara a continuación mediante **un índice único** y será introducido en la memoria del trabajo (MT)
* El hecho **initial-fact** esta por defecto y puede ser usado para iniciar nuestro sistema

**Eliminando hechos**

* Para eliminar un hecho o varios se usa el comando **(retract)** con un identificador
  + **(retract 1)** elimina el hecho con identificador 1
  + **(retract \*)** elimina todos los hechos

**Comandos hechos**

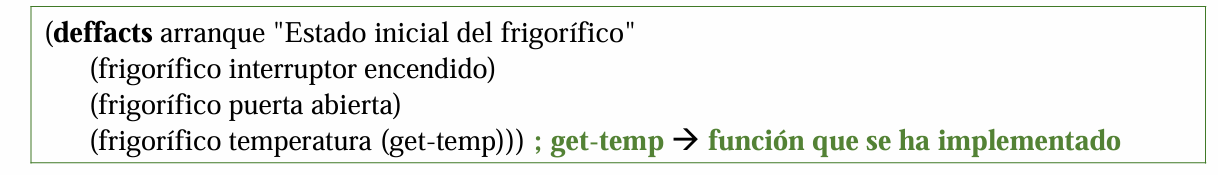
* Tecleando (facts) nos permite ver la lista completa de hechos actual,apareciendo cada hecho junto a su identificador

****

* Para listar los hechos a partir de un identificador usamos **(facts <iden>)**
* Para mostrar **un rango concreto de hechos (facts<idmin><idmax>)**

**Hechos definición de hechos iniciales**

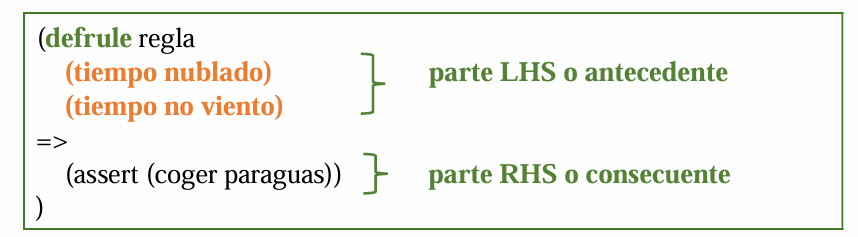
* En un problema siempre habrá una representación del mundo inicial:mundo pacman,etc
* El constructor deffacts permite especificar un conjunto de hechos como conocimiento inicial
  + (deffacts <nombre-colección-hechos>[<comentario>]<patron-RHS>)

****

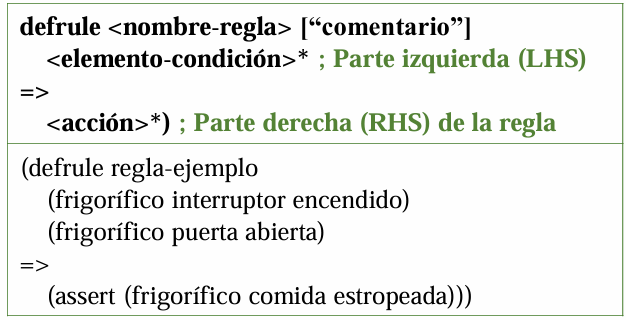
* + Se puede insertar la representación inicial mediante assert
  + Con el comando **reset** se eliminan todos los hechos que hubiera en la lista de hechos actual,y a continuación añade los hechos correspondientes a la sentencia deffacts.Siempre es recomendable y/o necesario al inicio ejecutar el comando **reset**
  + El comando (**clear)** elimina todos los hechos y las reglas

**Regla**

* 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(también llamados **elementos condicionales (EC)**) que deben satisfacerse para que la regla sea aplicable
* El consecuente de una regla es un conjunto de acciones a ser ejecutadas cuando la regla es aplicable

****

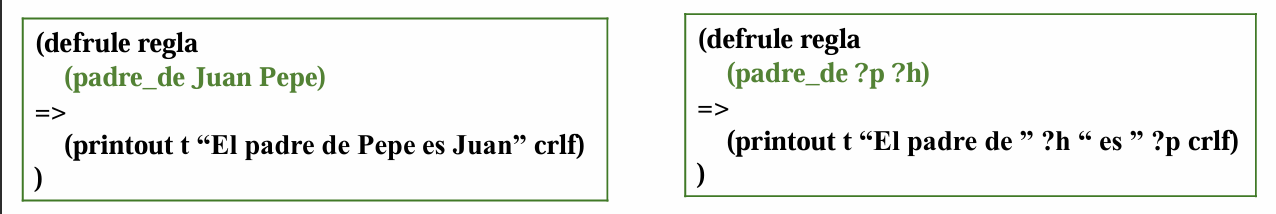
* **Una regla** CLIPS es una entidad **independiente:No es posible el paso de datos entre dos reglas**

****

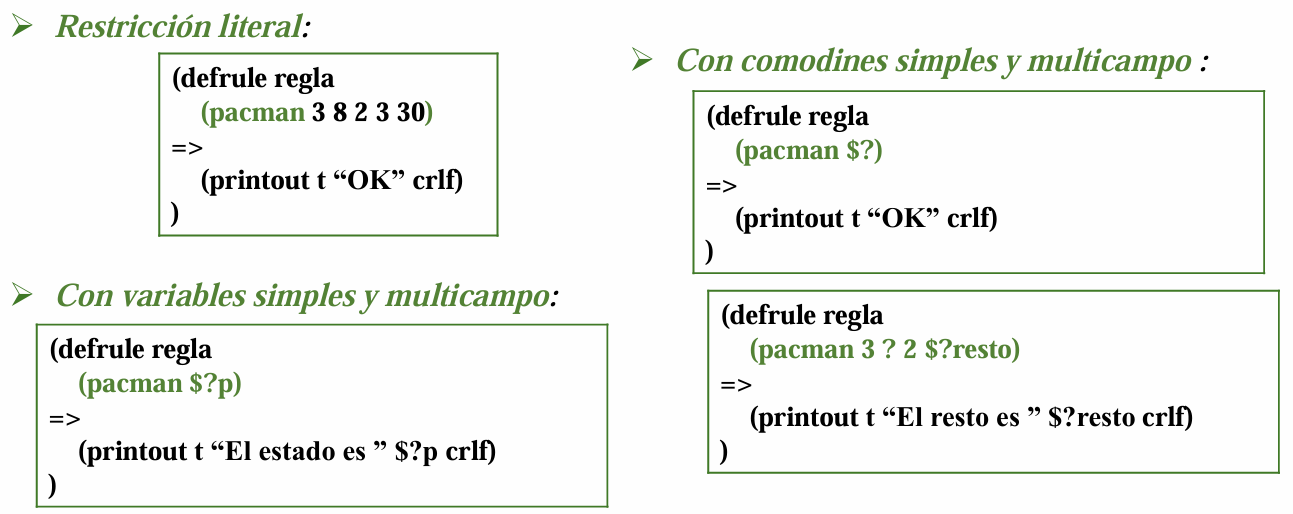
* **Si se introduce** en la base de reglas **una nueva regla con el mismo nombre** que el de una existente, la **nueva regla reemplazara a la antigua**
* El comentario, un string se usa normalmente para describir el propósito de la regla
* Si una regla **no tiene parte izquierda,** entonces el hecho inicial (initial fact) actuara como el elemento condicional, y la **regla se activará cada vez que se ejecute el comando reset**
* También puede no haber ninguna acción en el consecuente, con lo que la ejecución de la regla no tiene en ese caso ninguna consecuencia

**Como se activa/dispara una regla?**

* ¿Cómo se satisfacen los EC de una regla? La satisfactibilidad de un EC se basa en la existencia o no existencia en la MT (representacion) de los hechos especificados
* Para **matchear** con una regla sus campos deben aparecer en el mismo orden que indique la regla. Es decir, los hechos ordenados **codifican** la información según la posición

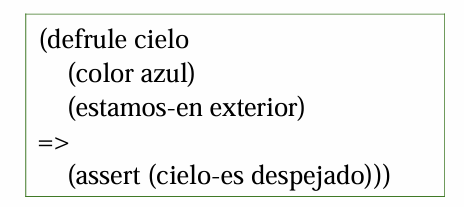
****

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

****

**Elementos Condicionales (EC)**

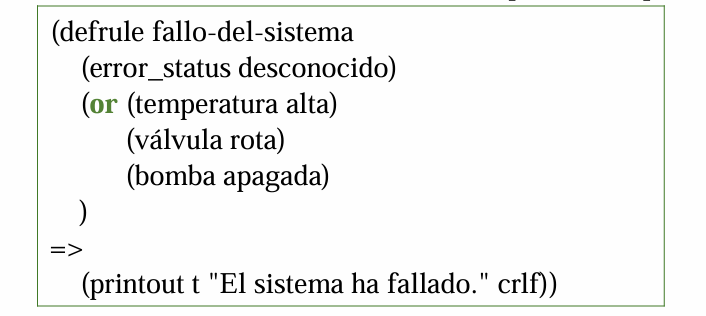
* **EC** Especifica restricciones sobre elementos de la lista de hechos:solo se satisface si existe una entidad que cumple las restricciones expresadas
* **La condición** o antecedente de una regla puede ser multiple

****

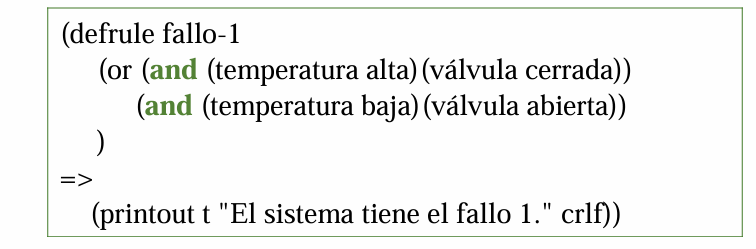
* Una regla se ejecuta cuando:
  + Todo sus elemento condicionales son satisfechos por la lista de hechos y/o la lista de instancias
  + El motor de inferencia la selecciona

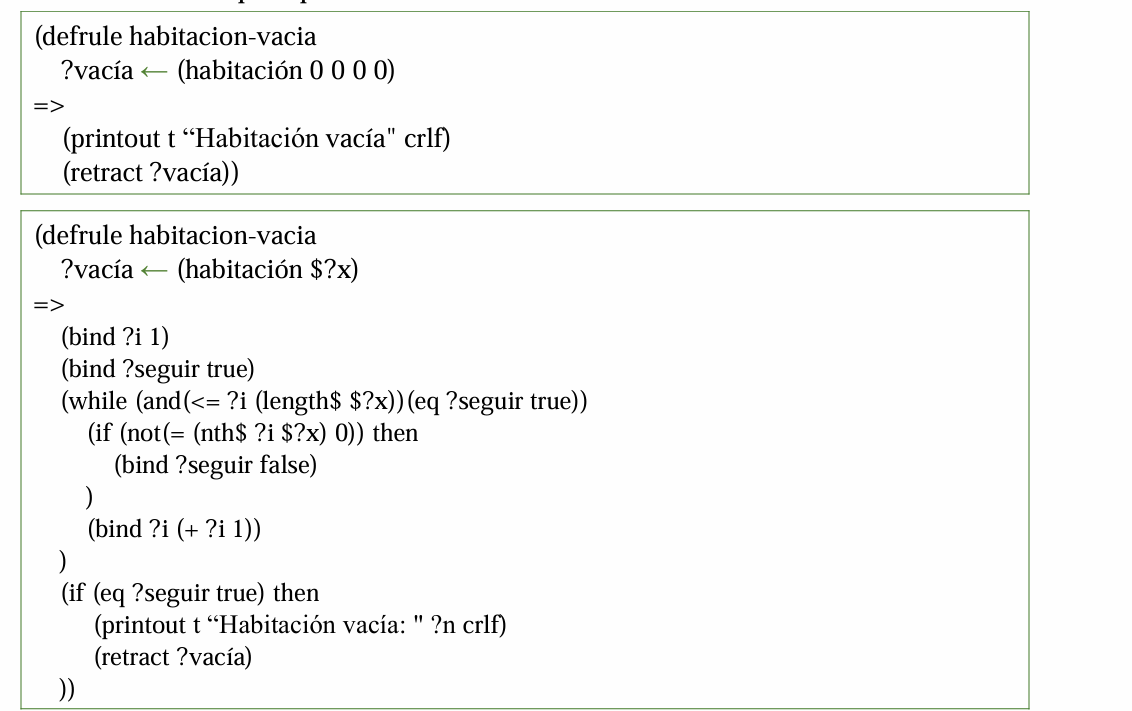
**Tipos de elementos condicionales (LHS)**

* **Or** este EC se satisface cuando al menos uno de los componentes que aparece se satisface

****

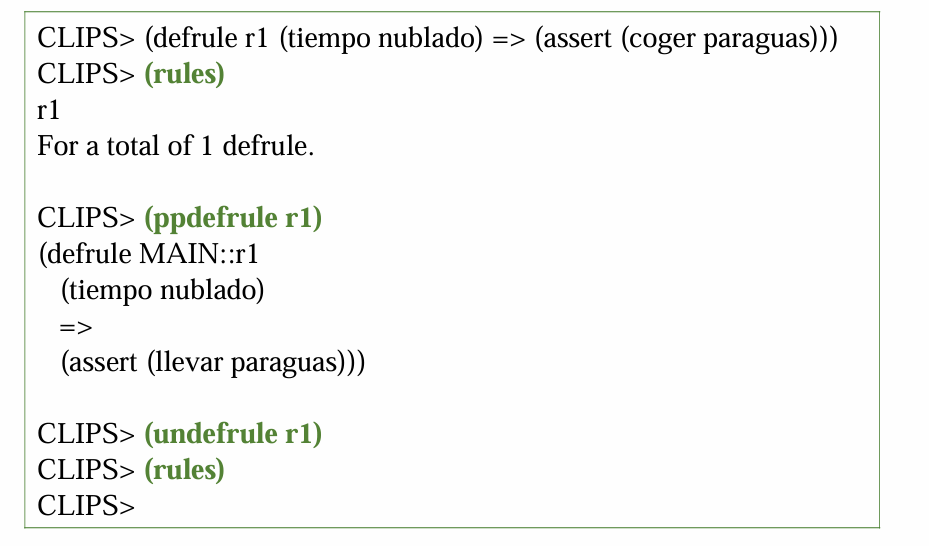
* **And** CLIPS supone que todas las reglas tienen un and implicito que rodea todos los elementos condicionales de la LHS

****

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

**Comandos reglas**

* Una vez introducida una o varias reglas,tecleando (rules) aparecen los nombres de las presentes en la base de conocimiento.El contenido de una regla con sus comentarios se muestra con (pprdeule <nombre regla>) y (undefrule <nombre regla>) la elimina (\* las elimina todas)

****

**Motor de inferencia**

* El motor de inferencia **intenta emparejar o encajar los hechos de la lista de hechos con los patrones de las reglas**
* Si el encaje **ocurre para todos los patrones de una regla, este** se **activa** o se **dispara**
  + ¿Cuántas veces se ejecuta una regla?
    - Esta comprobación para una regla **solo se realiza tras eliminar un hecho y volver a añadir o al modificar (que tengan que ver con la regla)**
* **La agenda** (Base de Conocimiento BC) almacena la lista de reglas activadas siguiendo un orden de prioridad
  + Existen **distintas estrategias** de resolución de conflictos que permiten establecer criterios a la hora de insertar una activación de regla en la agenda