



CUADERNO DE PRACTICAS

Sistemas Inteligentes

Práctica 2

Representación de la información mediante hechos

David Sánchez Fernández

i22safed@uco.es

INDICE

1. Conocimiento y representación del conocimiento en CLIPS	2
1.1. Resumen de la practica	2
1.2. Ejemplos de clase	5
1.2.1. Ejemplo 1	5
1.2.2. Ejemplo 2	5
1.3. Ejemplos propuestos	6
1.3.1. Ejemplo 1	7
1.3.2. Ejemplo 2	8
2. Representación de la información mediante hechos	9
2.1. Introducción a las plantillas	9
2.2. Estructura de plantillas	9
2.2.1. Ejemplo de plantilla	10
2.3. Ordenes relacionadas con plantillas	10
2.4. Constructor de hechos (deffacts)	11
2.5. Estructura de constructor de hechos	11
2.5.1. Ejemplo del constructor de hechos	11
2.6. Ejercicios de plantillas	11
3. Bibliografía	18

1. Conocimiento y representación del conocimiento en CLIPS

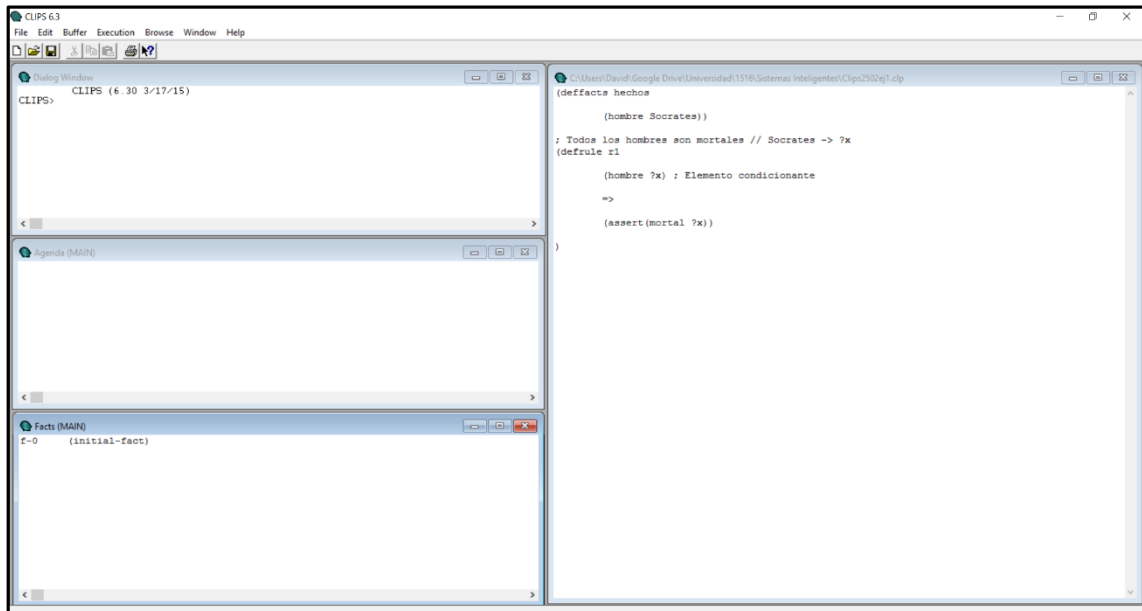
1.1. Resumen de la practica

En la primera práctica hemos realizado una toma de contacto con el lenguaje de programación CLIPS, un lenguaje basado en reglas y hechos destinado principalmente para sistemas expertos.

La arquitectura está compuesta principalmente por tres elementos:

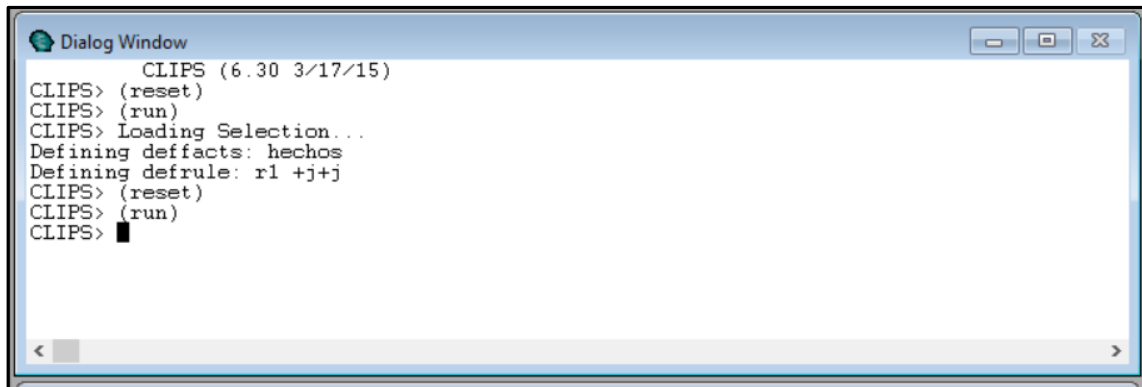
- Base de reglas: Almacena que representan el conocimiento para la resolución del problema.
- Interprete o motor de inferencia: Maneja la ejecución de las reglas del programa.
- Memoria del trabajo: Este elemento contiene los hechos que representan el conocimiento que el sistema ha adquirido del problema que se intenta recuperar.

El entorno con el que trabajaremos a lo largo de las practicas será CLIPS en su versión 6.3, que tiene el siguiente aspecto:

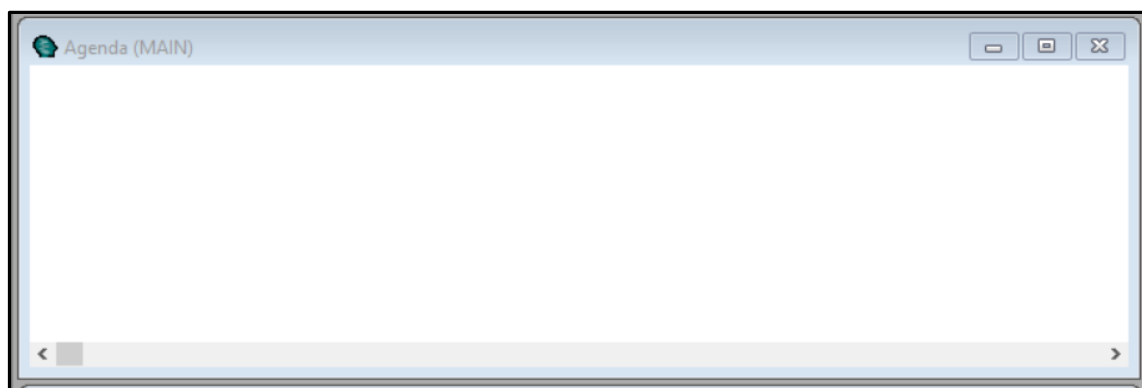


Las ventanas que más utilizaremos serán las siguientes

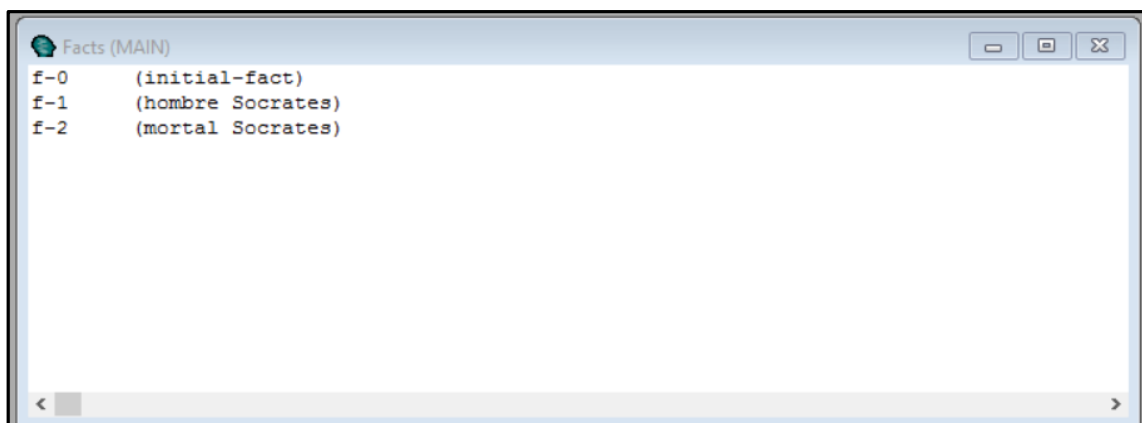
- Ventana de dialogo: Podríamos definir esta ventana como consola donde podremos ver los pasos que se siguen durante la ejecución e introducir órdenes



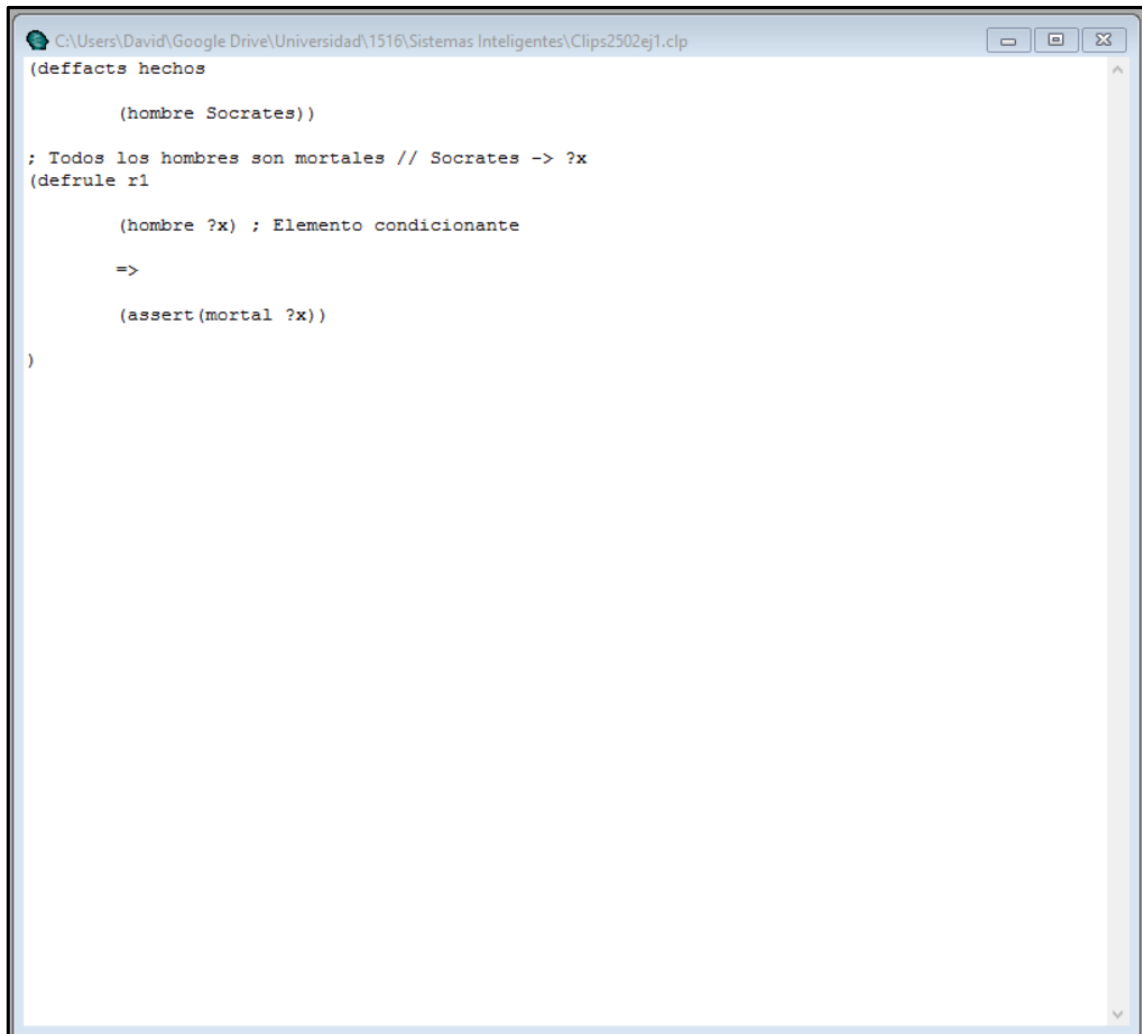
- Agenda: Aparece la colección de reglas activadas



- Base de hechos: En esta ventana aparecen los hechos definidos



- Editor: Utilizaremos el editor para visualizar el código, modificarlo y cargarlo (seleccionar texto y pulsar Ctrl. + K)

A screenshot of a text editor window. The title bar shows the file path: C:\Users\David\Google Drive\Universidad\1516\Sistemas Inteligentes\Clips2502ej1.clp. The editor contains the following Prolog code:

```
(def facts hechos
  (hombre Socrates))

; Todos los hombres son mortales // Socrates -> ?x
(defrule r1
  (hombre ?x) ; Elemento condicionante
  =>
  (assert(mortal ?x))
)
```

Para ejecutar un programa, primero debemos cargar el programa en el buffer pulsando Ctrl + K (Load selection) o pulsando en el menú de buffer las opciones de Load buffer.

Una vez cargado, ahora en el menú Execution, debemos pulsar Reset (Ctrl + E), y acto seguido run (Ctrl + R)

1.2. Ejemplos de clase

1.2.1. Ejemplo 1

```
(deffacts hechos
  (hombre Socrates))

; Por comparación, hombre empareja hombre, y por ende Sócrates
; se asigna a ?x
(defrule r1
  (hombre ?x) ; Elemento condicionante
=>
  (assert(mortal ?x))
)
```

En este ejemplo comprobamos si **Socrates es mortal**, por lo tanto, declaramos que **Socrates es hombre**. Después definimos la regla **todos los hombres son mortales** además declaramos **hombre** como elemento condicionante. Cuando la regla valide el elemento condicionante, el aserto se activará confirmando que **Socrates es mortal**.

1.2.2. Ejemplo 2

```
(deffacts h1; constructor de hechos
(hombre Socrates); hecho ordenador con dos campos ocupados por
símbolos
(hombre Platon)
)
(defrule r1; declaración de la primera regla
  (hombre ?x); Elemento condicional del patrón
=>
  (assert(mortal ?x)); Acción de afirmación del hecho
                        ; ordenado (mortal)
)
```

```
(defrule r2; Declaración de la segunda regla
  (mortal ?x)
  =>
  (printout t ?x " Es mortal" crlf )
)
```

Este ejemplo es similar al anterior con la única diferencia de que tenemos una regla, es decir, afirmamos que Socrates es mortal y Platon es mortal.

Comprobamos primeramente el antecedente de la primera regla, hombre coincide con hombre y a la variable del consecuente (?x) se le asigna el valor Socrates después ejecutamos la segunda regla y donde mortal coincide y por lo tanto imprime Socrates (?x) es mortal.

Pasa lo mismo para el caso de Platón

1.3. Ejemplos propuestos

1.3.1. Ejemplo 1

```
; Ejemplo para ver como funciona el motor de inferencia
; - Como el motor de inferencia realiza el mecanismo de inferencia
; de comparación de patrones
; - Como se liga el valor de un campo a una variable
; - Ver elemento condicional del test
; - Ver operadores aritmeticos
; - Ver como se liga la direccion de un hecho a una variable
; - Ver como retractar un hecho

; =====
; Presentar por pantalla los diez primeros números naturales

( deffacts h1; constructor de hechos
```

```

        (n 0); Hecho ordenado
    )

(defrule r1
    ?f<-(n ?x) ; Elemento condicional de patron(ECP)
                ; A la variable ?x se le ligará valores de los
                ; hechos que emparejen
                ; A la variable ?f se le liga la direccion de hecho
                ; con el que empareje el ECP

    (test ( z ?x 10)); Elemento condicional del test

    =>

    (printout t "n= " ?x crlf) ; Accion de imprimir

    (assert (n (+ ?x 1)))      ; Afirmación de un hecho nuevo ( n
                                ; resultado_de_la_suma)

    (retract ?f)
                                ; Elimina el hecho de cuya dirección está en
                                ; la variable ?f

)

```

1.3.2. Ejemplo 2

; Ejemplo de como sumar los 7 numeros naturales


```

( deffacts h1 ; Constructor de hechos

    (n 0) ; Hecho ordenado
    (suma 0)

)

(defrule r1
?f1<-(n ?x) ; Elemento condicional patron (ECP)
    ; A la variable ?x se le ligará los valores de
    ; los hechos emparejados
    ; A la variable ?f se le liga la direccion de
    ; hecho con el que empareje
    ; el ECP

?f2<-(suma ?s)
(test (< ?x 7)) ; Elemento condicionante del test
=>
(printout t "n= " ?x crlf) ; Accion de imprimir

    (assert (n (+ ?x 1))) ; Afirmacion de un hecho
    ; nuevo ( n resultado-
    ; de-la-suma)

    (retract ?f1) ; Elimina el hecho cuya
    ; direccion esta en la

```

```

; variable ?f1
(retract ?f2) ; Elimina el hecho cuya
; direccion esta en la
; variable ?f2

(assert (suma (+ ?s ?x)))
)

```

2. Representación de la información mediante hechos

2.1. Introducción a las plantillas

Utilizaremos el constructor de plantillas (deftemplate)

```

(deftemplate persona
  (slot nombre)
  (slot edad)
  (multislot direccion)
)

```

2.2. Estructura de plantillas

· Valores por defecto

```

<atributo_por_defecto> ::=
(default ?DERIVE | ?NONE | <expresion>*) | ...

```

2.2.1. Ejemplo de plantilla

Ejemplo del cuestionario el de los slots w, x, y, z

```
(slot edad ( type LEXEME)) ; tipo lexema == string
(slot edad ( type INTEGER SYMBOL)) ; tipo entero o
                                ; símbolo
(slot x(allowed-values uno dos)) ; valores permitidos
                                ; uno o dos,
                                ; es decir que no puede
                                ; introducir 1 ó 2
```

2.3. Ordenes relacionadas con plantillas

```
(ppdeftemplate <nombre-plantilla>) ; Muestra plantilla ya
definida
(list-deftemplates [<nombre-modulo> | *]) ; Lista las
                                           ; plantillas que
                                           ; están siendo
                                           ; utilizadas
                                           ; por el sistema
(undeftemplate <nombre-plantilla> | *) ; Elimina una
                                           ; plantilla
                                           ; existente
```

2.4. Constructor de hechos (deffacts)

Corchetes: Opcional (`list-deftemplates [<nombre-modulo> | *]`)

Asterisco: 0 ó más hechos (`list-deftemplates [<nombre-modulo> | *]`)

Suma: Que se repite (`eq <expression> <expression>+`)

2.5. Estructura de constructor de hechos

```
(deffacts <nombre-definición> [<comentario>] <hecho> *)
```

2.5.1. Ejemplo del constructor de hechos

```
(deffacts personas

  (persona (nombre Juan)
            (apellidos Fernandez)
            (dni 1))

)
```

2.6. Ejercicios de plantillas

; Ejercicio 1

```
(deftemplate persona
```

```
  (slot nombre (type STRING))
```

```
  (slot apellido (type STRING))
```

```
(slot color_ojos (type SYMBOL)
                (default marrones))
(slot altura(type FLOAT)
            (default 1.85))
(slot edad (type INTEGER)
            (default 22))
)
```

; Ejercicio 2

```
(deftemplate pacientes
  (slot nombre)
  (slot apellido)
  (slot dni)
  (slot seguroMedico)
)
```

```
(deftemplate visitas
  (slot fecha)
  (multislot sintomas)
  (slot pruebas)
  (slot medicacion)
  (slot dni)
)
```

; Ejercicio 3

```
(deftemplate trayectos_aereos
  (slot origen)
  (slot destino)
)
;v1
```

```

(assert(trayectos_aereos(origen Lisboa)(destino Paris)))
(assert(trayectos_aereos(origen Estocolmo)(destino Paris)))
(assert(trayectos_aereos(origen Lisboa)(destino Madrid)))
(assert(trayectos_aereos(origen Roma)(destino Madrid)))
(assert(trayectos_aereos(origen Roma)(destino Lisboa)))
(assert(trayectos_aereos(origen Paris)(destino Roma)))
(assert(trayectos_aereos(origen Frankfurt)(destino Roma)))
(assert(trayectos_aereos(origen Roma)(destino Frankfurt)))
(assert(trayectos_aereos(origen Frankfurt)(destino Estocolmo)))

;v2
(deffacts datos
  (trayectos_aereos(origen Lisboa)(destino Paris))
  (trayectos_aereos(origen Estocolmo)(destino Paris))
  (trayectos_aereos(origen Lisboa)(destino Madrid))
  (trayectos_aereos(origen Roma)(destino Madrid))
  (trayectos_aereos(origen Roma)(destino Lisboa))
  (trayectos_aereos(origen Paris)(destino Roma))
  (trayectos_aereos(origen Frankfurt)(destino Roma))
  (trayectos_aereos(origen Roma)(destino Frankfurt))
  (trayectos_aereos(origen Frankfurt)(destino Estocolmo))
)

```

; Ejercicio 4

```

(deftemplate familia
  (slot familiar1(type STRING))
  (slot familiar2(type STRING))

```

```

        (slot relacion(type STRING))
    )

    (deffacts hechos
    (familia(familiar1 "Toni") (familiar2 "Elena") (relacion
    "marido_mujer"))
    (familia(familiar1 "Toni") (familiar2 "Rafa") (relacion
    "padre_hijo"))
    (familia(familiar1 "Toni") (familiar2 "Saul") (relacion
    "padre_hijo"))
    (familia(familiar1 "Lourdes") (familiar2 "Rafa") (relacion
    "madre_hijo"))
    (familia(familiar1 "Lourdes") (familiar2 "Saul") (relacion
    "madre_hijo"))

        (familia(familiar1 "Rafa") (familiar2 "Toni") (relacion
        "hermanos"))
    )

```

; Ejercicio 5

```

(deftemplate libro
    (multislot autor (type STRING)
        (default ?NONE))
    (multislot titulo (type STRING)
        (default ?NONE))
    (multislot editorial (type STRING)
        (default ?DERIVE))
    (multislot edicion(type NUMBER)
        (default 0))

```

```

        (multislot anyo (type INTEGER)
          (default ?DERIVE))
      )

      (deffacts hechos
        (libro (autor "Mira,J.Delgado A.E, Boticario J.G") (titulo "Aspectos
        basicos de la inteligencia artificail") (editorial "Sanz y
        Torres")(edicion) (anyo 1995))

        (libro (autor "Galan, S. F. ;Boticario, J. G. ;Mira, J") (titulo
        "Problemas resueltos de inteligencia artificial") (editorial "Adisson-
        weley IberoAmericana")(edicion)(anyo 1998))

        (libro (autor "Rich E. Knight, K") (titulo "Inteligencia Artificial")
        (editorial "McGraw-Hill") (edicion 2)(anyo 1994))
      )

```

; Ejercicio 6

```

(deftemplate coche
  (slot num_coches(type INTEGER)
    (default ?DERIVE))
  (slot modelo(type STRING))
  (slot cilindrada(type INTEGER)
    (default ?DERIVE))
  (slot combustible(type STRING)
    (allowed-strings "gasoil" "gasolina"))
  (multislot num_puertas(type INTEGER))
  (multislot color(type STRING))
  (multislot vendedor(type STRING)
    (default ?DERIVE))
  (multislot fecha(type SYMBOL))

```



```

        (default ?DERIVE))

    (multislot cliente(type STRING))

)

(deffacts hechos

    (coche(num_coches 1)(modelo "clio")(cilindrada 1600)(combustible
"gasolina")(num_puertas 3)(color "azul"))

    (coche(num_coches 1)(modelo "clio")(cilindrada 1800)(combustible
"gasoil")(num_puertas 5)(color "blanco"))

    (coche(num_coches 1)(modelo "megane")(cilindrada
1800)(combustible "gasoil")(num_puertas 5)(color "dorado"))

    (coche(num_coches 2)(modelo "megane")(cilindrada
1600)(combustible "gasolina")(num_puertas 5)(color "gris"))

    (coche(num_coches 1)(modelo "laguna")(cilindrada
2000)(combustible "gasolina")(num_puertas 5)(color "negro"))

    (coche(num_coches 1)(modelo "megane")(fecha 10/10/2003)(vendedor
"Juan Perez")(cliente "Esteban Losada"))

    (coche(num_coches 1)(modelo "laguna")(fecha 13/10/2003)(vendedor
"Ana Ballester")(cliente "Juan Cano"))

    (Juan Perez vendio un mengane el 10/10/2003 al cliente Esteban
Losada)

    (Ana Ballester vendio un laguna el 13/10/2003 al cliente Juan
Cano)

)

```

; Ejercicio 8

```

(deftemplate ingredientes

    (multislot pisto)

    (multislot tortilla)

    (multislot despensa)

```

```

        (multislot comprar)
    )

    (def facts hechos
        (ingredientes(pisto pimientosVerdes pimientosRojos berenjenas
            calabacines cebollas tomateTriturado sal aceite))

        (ingredientes(tortilla huevos patatas cebollas sal aceite))

        (ingredientes(despensa pimientosVerdes pimientosRojos cebollas
            aceite)(comprar calabacines berenjenas tomateTriturado sal huevos
            patatas ))

        (compras)

        (ingredientes(despensa pimientosVerdes pimientosRojos cebollas
            aceite calabacines berenjenas tomateTriturado sal huevos
            patatas)(comprar todo comprado)))

    )

    (def facts hechos2
        (cocinar)

        (ingredientes(despensa pimientosVerdes pimientosRojos aceite
            calabacines berenjenas sal huevos patatas)(comprar tomateTriturado
            cebollas)))

    )

```

3. Bibliografía

- Diapositivas Tema 1 de la asignatura:

http://moodle.uco.es/m1516/pluginfile.php/183484/mod_resource/content/0/Practicas/tema1.pdf

- Elementos básicos de CLIPS:

<http://www.uco.es/users/sventura/misc/TutorialCLIPS/TutorCLIPS02.htm>