David Sánchez Fernández

i22safed@uco.es

Práctica 1

Conocimiento y representación del Conocimiento en CLIPS

Cuaderno de practicas

Sistemas Inteligentes

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. Bibliografía 11

**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.

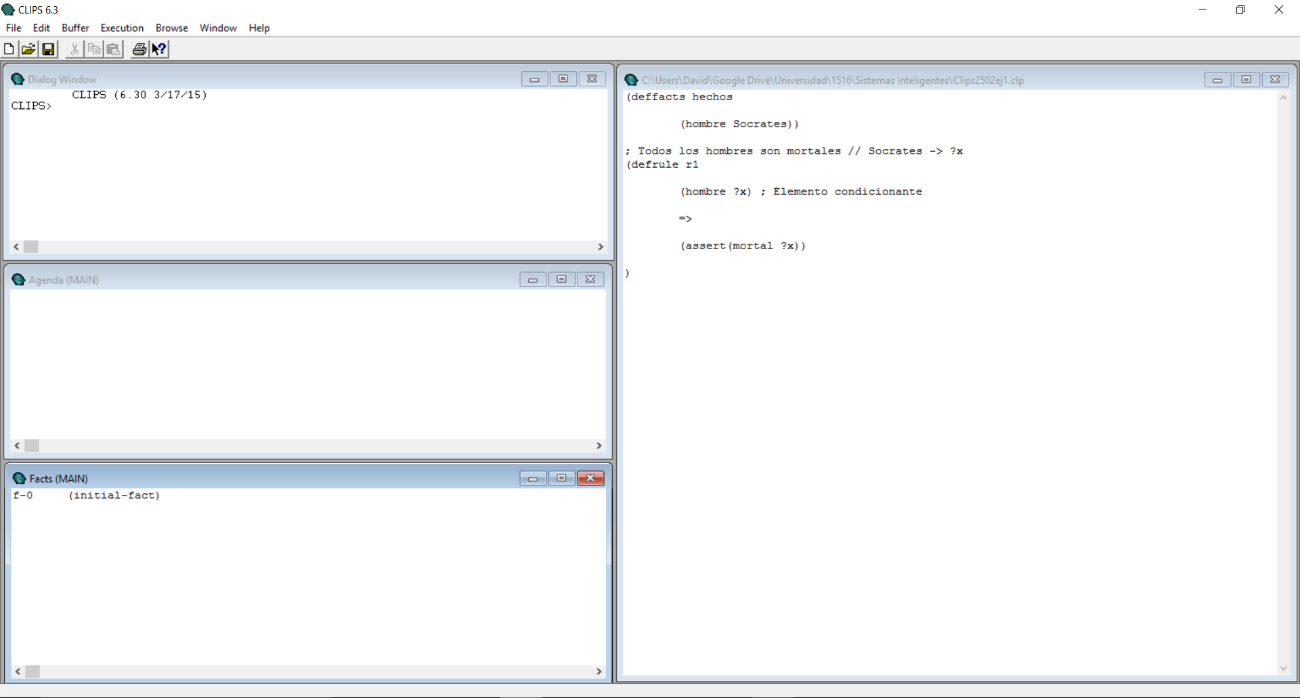
La arquitectura está compuesta principalmente por tres elementos:

· 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.

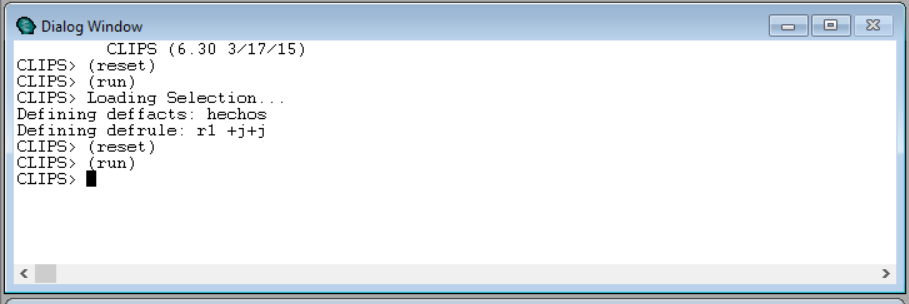
· Memoria del trabajo: Este elemento contiene los hechos que representan el conocimiento que el sistema ha adquirido del problema que se intenta recuperar.

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

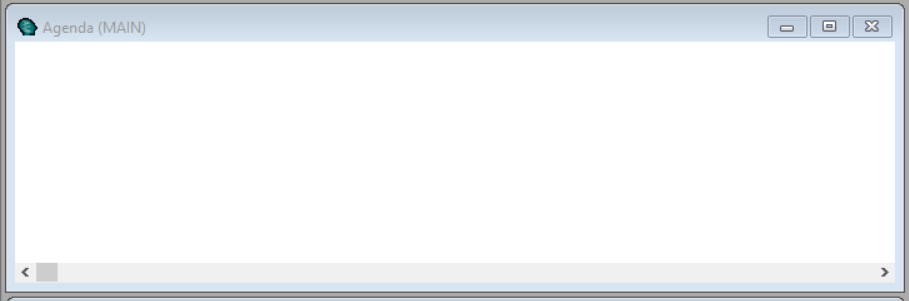


Las ventanas que más utilizaremos serán las siguientes

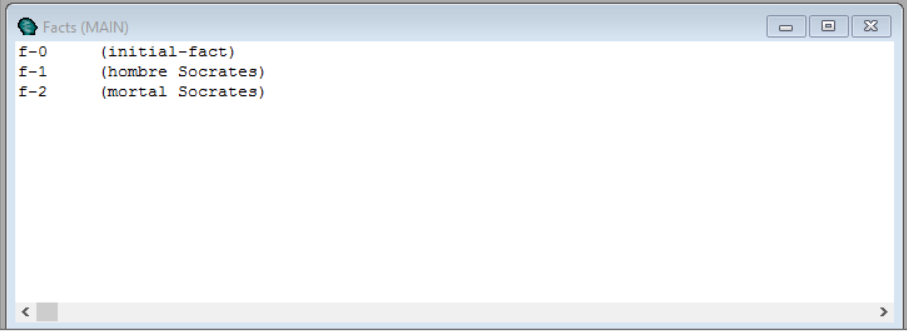
· Ventana de dialogo: Podríamos definir está ventana como consola donde podremos ver los pasos que se siguen durante la ejecución e introducir órdenes



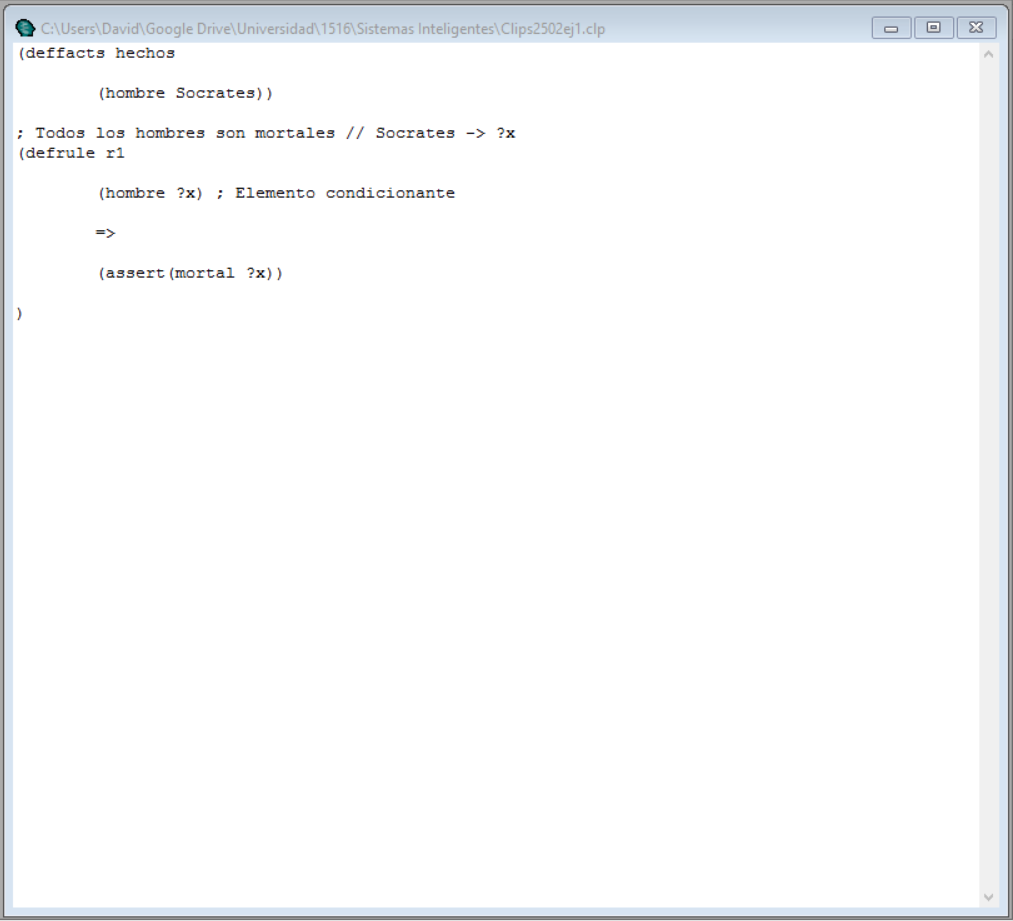
· Agenda: Aparece la colección de reglas activadas



· 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)



Para ejecutar un programa, primero debemos cargar el programa 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. Bibliografía**

· Diapositivas Tema 1 de la asignatura:http://moodle.uco.es/m1516/pluginfile.php/183484/mod\_resource/content/0/Practic as/tema1.pdf