



CLIPS

Introducción a CLIPS (I)





Contenido

1. Introducción
2. Elementos básicos: hechos ordenados y reglas
3. Interacción con CLIPS.
4. Estrategias de resolución de conflictos.



1. Introducción



CLIPS (I)

- Historia de CLIPS (C Language Integrated Production System).
 - Inspirado en **OPS5** (Official Production System 5) y **ART** (Automated Reasoning Tool, Inference Corporation) (Herramientas de desarrollo de sistemas de producción o sistemas expertos).
 - Implementado en C por eficiencia y portabilidad.
 - Desarrollado por NASA (1986).
 - Licencia: public domain.



CLIPS (II)

- Herramienta para el desarrollo de Sistemas Expertos / SBC.
 - Intérprete de alto nivel.
 - Intérprete sistema de producción.
 - Programación orientada a objetos COOL (CLIPS Object-Oriented Language).
 - Lenguaje operacional sintaxis tipo LISP.
 - Permite integración con C, Ada o lenguajes procedurales.
 - Distingue entre mayúsculas y minúsculas (case-sensitive).



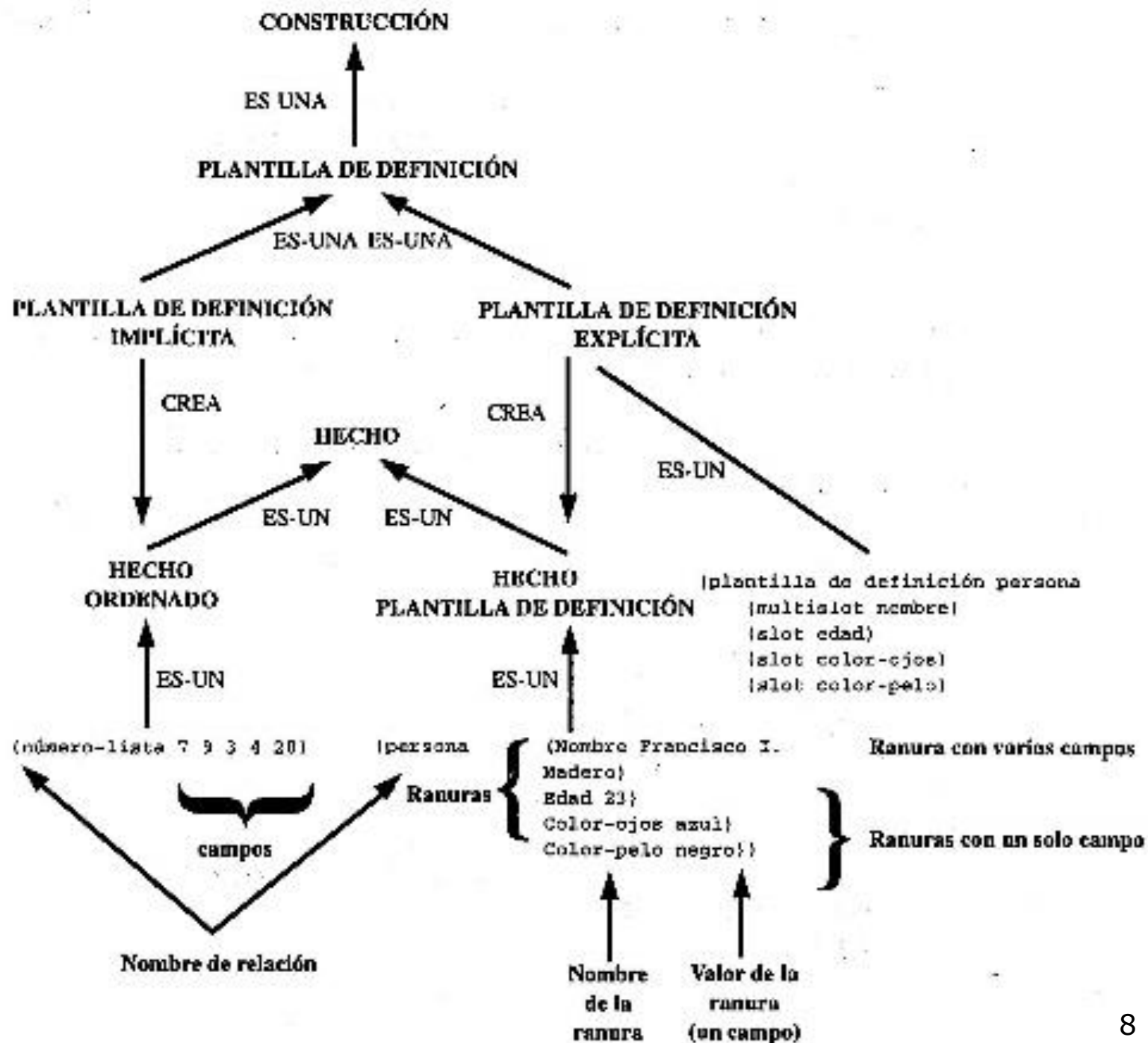
Sobre CLIPS: Manuales

- clipsrules.sourceforge.net
- Versión: 6.3
- Documentación en línea (<http://clipsrules.net/>)
 - [CLIPS user guide](#).
 - Introducción a la programación en CLIPS.
 - CLIPS reference manual.
 - Vol I: [programación básica](#).
 - Vol II [programación avanzada](#).
 - Vol III: [interfaces](#).
- J. Giarratano, G. Riley. EXPERT SYSTEMS, Principles and Programming, Third Edition. PWS Publishing Company, Boston 1998. ISBN 0-534-95053-1.



Intérprete sistema de producción

- Memoria de trabajo: hechos (instancias)
- Base de reglas
- Motor de inferencias: adelante, RETE
- Programa CLIPS
 - Hechos y reglas (constructores de) (tb. Objetos, funciones).
 - Los hechos que alcanzan la memoria de trabajo determinan qué reglas se pueden disparar.
 - El motor de inferencias determina qué reglas y cuándo se disparan.
 - Actividad guiada por datos.





2. Elementos básicos: hechos ordenados y reglas



Hechos ordenados

- Hechos ordenados:
 - (status walking)
 - (persona nombre "Luis Prieto" edad 53 altura 1.83)
 - (persona "Luis Prieto" 53 1.83)
- No hay que definir su estructura
 - Hechos iniciales mediante constructor **defacts**
 - Añadir dinámicamente mediante función **assert**



Operaciones con hechos (1)

- Añadir un hecho a la memoria de trabajo
CLIPS> (assert (persona "Luis Prieto" hombre 53))
<Fact-1>
- Examinar hechos en la memoria de trabajo
CLIPS> (facts)
f-0 (initial-fact)
f-1 (persona "Luis Prieto" hombre 53)
For a total of 2 facts.
- Eliminar un hecho de la memoria de trabajo
CLIPS> (retract 1)
CLIPS> (facts)



Operaciones con hechos (2)

- Definir hechos iniciales en base de conocimiento

```
(deffacts amigos "algunos amigos"  
  (persona "Luis Prieto" 53 1.83)  
  (persona "Ana Perez" 56 1.70)  
)
```



Reglas

- Definir reglas en base de conocimiento

```
(defrule nombre-amigos "Encontrar el nombre de los amigos"
  (declare (salience 0))
  (persona ?x ? ?)
=>
  (assert (encontrado ?x))
)
```



Comodines y Variables

- **Comodines:** indican que cualquier valor en esa posición de la entidad patrón es válido para emparejar con la regla.
Comodín **monocampo:** ? (empareja con un campo)
Comodín **multicampo:** \$? (empareja con 0 o más campos)
- **Variables monocampo:** ?x (un solo campo)
Variables multicampo: \$?y (entre 0 y más campos)



Prioridad de una Regla

- (saliency N)
 - Para especificar la prioridad de activación de una regla se declara esta propiedad. (declare (saliency 0))
 - N puede tomar valores entre -10000 y 10000.
 - Cuanto más alto, más prioridad.
 - Por defecto es 0.



Prioridad de una Regla (ejemplo)

CLIPS> (clear)

CLIPS> (defrule primera

 (declare (salience 10))

 =>

 (printout t "Me ejecuto la primera" crlf))

CLIPS> (defrule segunda

 =>

 (printout t "Me ejecuto la segunda" crlf))

CLIPS> (reset)

CLIPS> (run)

Me ejecuto la primera

Me ejecuto la segunda



3. Interacción con CLIPS



Base de conocimiento: extensión .clp

; Ejemplo inicial: AMIGOS



; Hechos iniciales

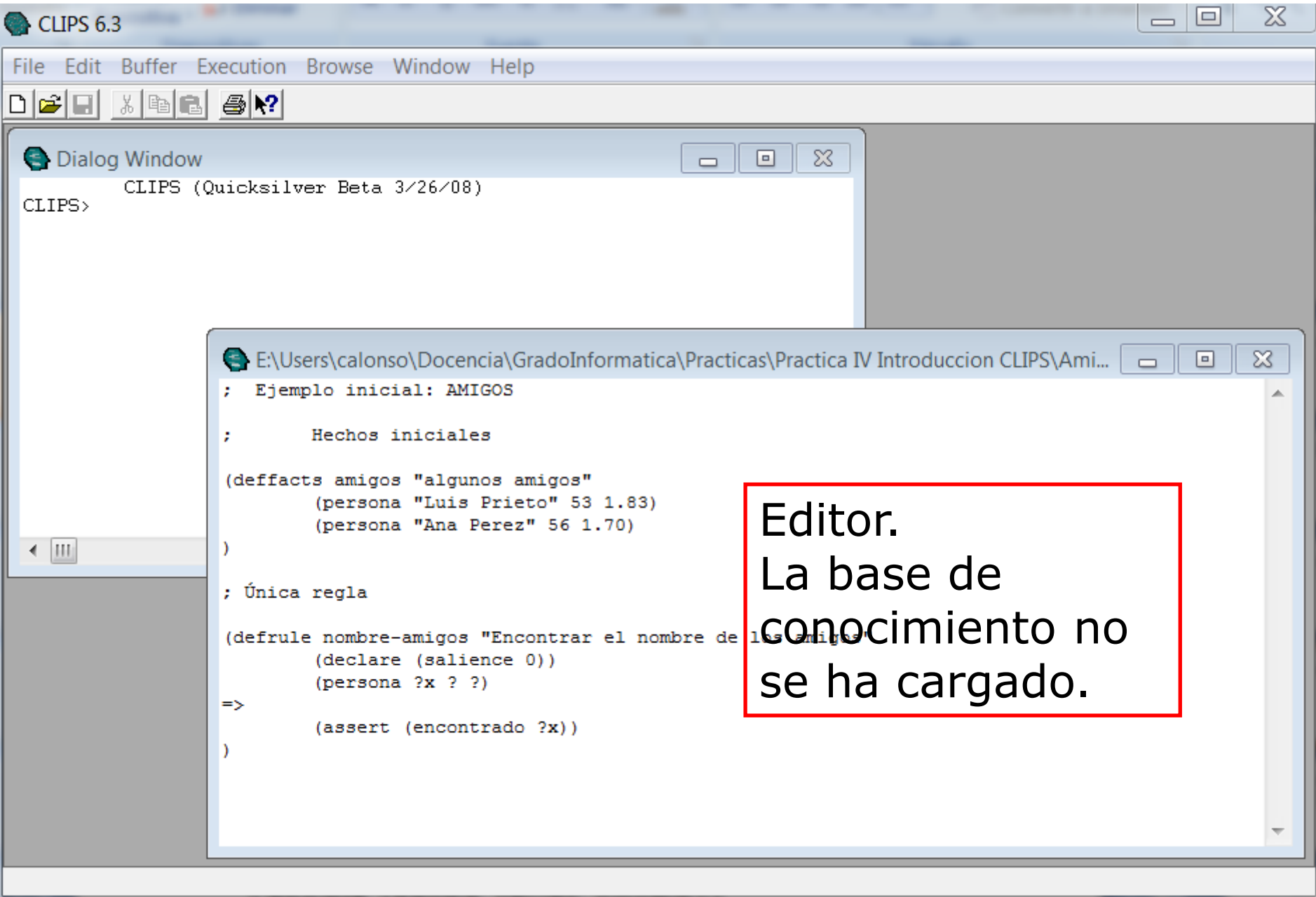
```
(deffacts amigos "algunos amigos"  
  (persona "Luis Prieto" 53 1.83)  
  (persona "Ana Perez" 56 1.70)  
)
```

; Única regla

```
(defrule nombre-amigos "Encontrar el nombre de los amigos"  
  (declare (salience 0))  
  (persona ?x ? ?)  
=>  
  (assert (encontrado ?x))  
)
```

Invocando clips

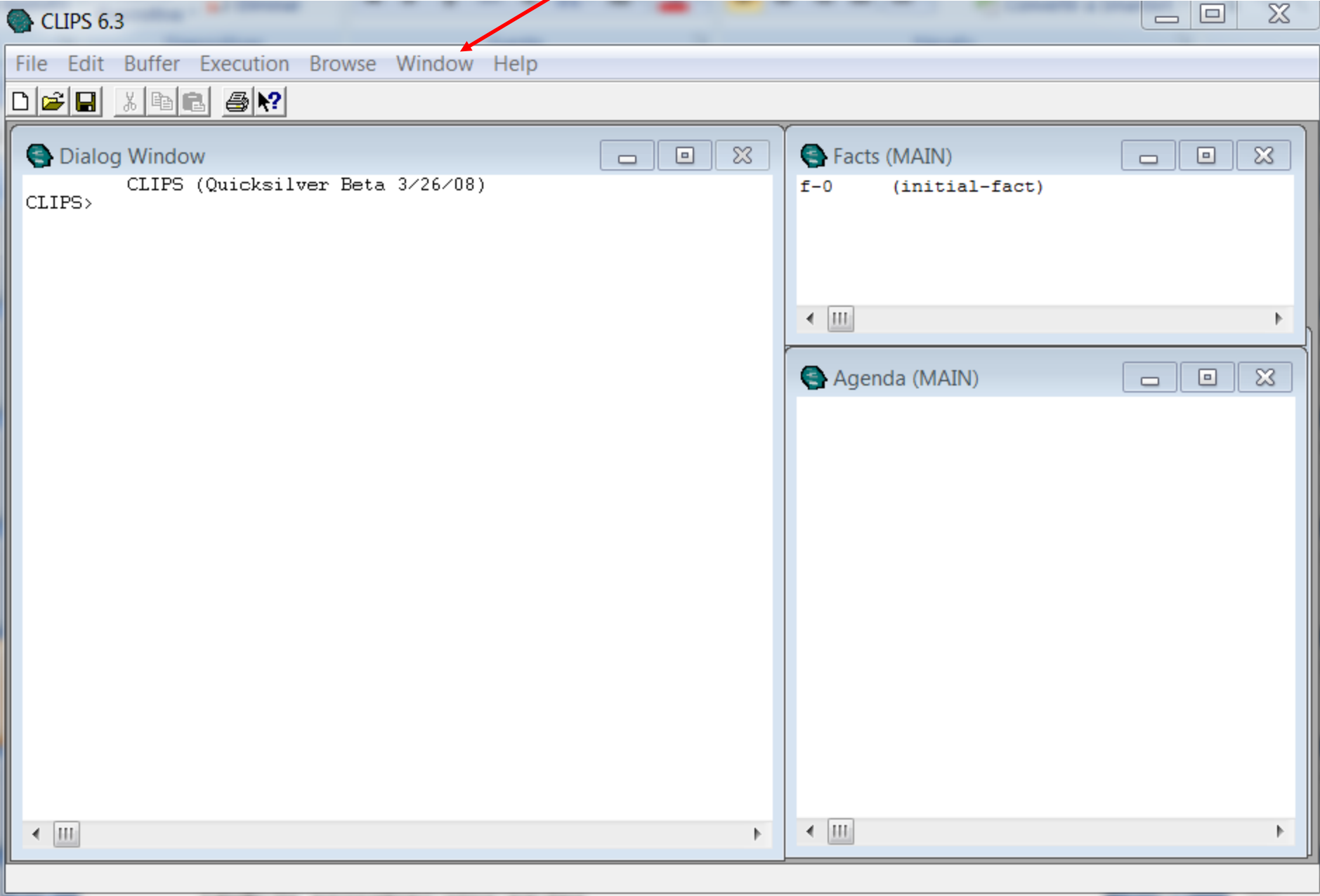
Nombre	Fecha de modifica...	Tipo	Tamaño
 Amigos	18/01/2011 12:32	Archivo CLP	1 KB
 CLIPS Introduccion	18/01/2011 12:32	Presentación de M...	209 KB





Examinando el entorno

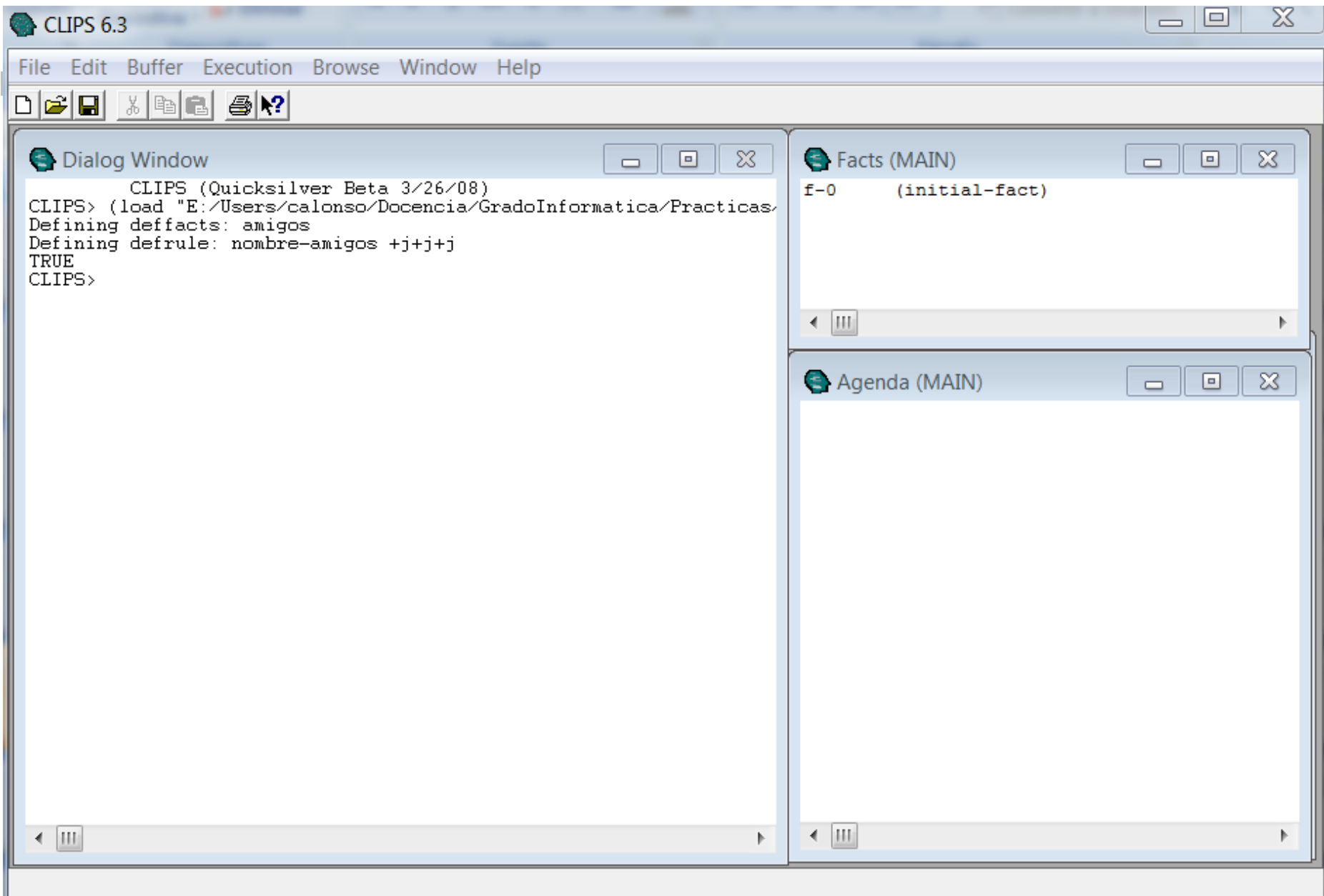
- Ventanas:
 - Dialog: interfaz comandos, entrada/salida por defecto.
 - Facts: Hechos en la memoria de trabajo.
 - Agenda: reglas activadas, ordenadas según estrategia resolución de conflictos, pendientes de disparo.
 - En menú *Window*.





Preparar el entorno (1)

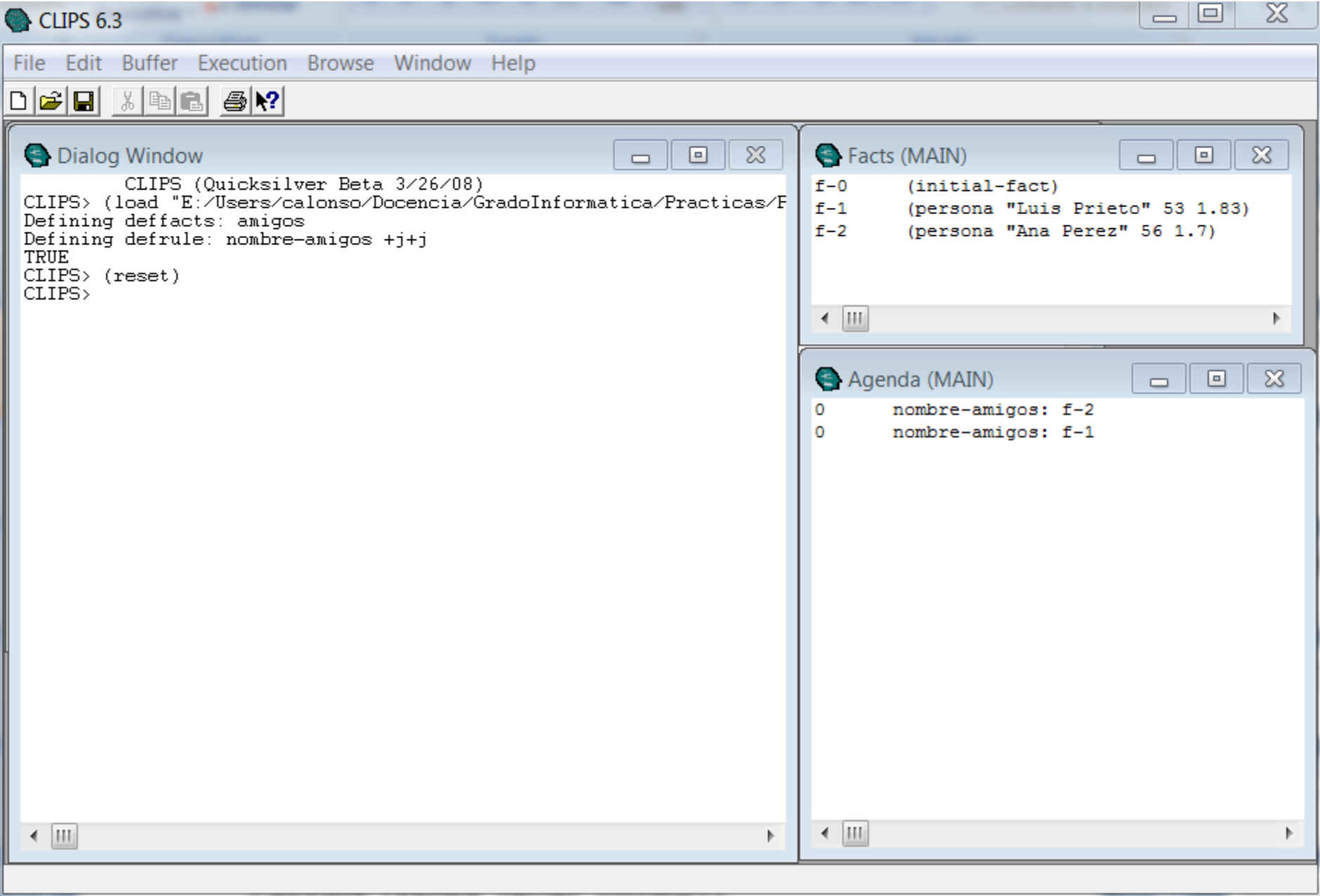
- (clear)
 - Eliminar base de conocimiento y limpiar memoria de trabajo.
 - En menú *Execution*.
- (load)
 - Leer base de conocimiento.
 - Evaluar constructores.
 - Crear red Rete (compilación incremental).
 - En menú *File*.





Preparar el entorno (2)

- (reset)
 - Limpia la memoria de trabajo.
 - Procesa los hechos de los constructores de facts con la red Rete.
 - Determina las reglas activadas: su antecedente se satisface
 - Ordena las reglas en la agenda: aplica la estrategia de resolución de conflictos.
 - En menú *Execution*.



CLIPS 6.3

File Edit Buffer Execution Browse Window Help

Dialog Window

CLIPS (Quicksilver Beta 3/26/08)
CLIPS> (load "E:/Users/calonso/Docencia/GradoInformatica/Practicas/F
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
TRUE
CLIPS> (reset)
CLIPS>

Facts (MAIN)

f-0 (initial-fact)
f-1 (persona "Luis Prieto" 53 1.83)
f-2 (persona "Ana Perez" 56 1.7)

Agenda (MAIN)

0 nombre-amigos: f-2
0 nombre-amigos: f-1

Índice a los hechos

Último: f-2

CLIPS 6.3

File Edit Buffer Execution Browse Window Help

Dialog Window

CLIPS (Quicksilver Beta 3/26/08)
CLIPS> (load "E:/Users/calonso/Docencia/GradoInformatica/Practicas/F
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
TRUE
CLIPS> (reset)
CLIPS>

Índice a los hechos
Último: f-2

Facto (MAIN)

f-0 (initial-fact)
f-1 (persona "Luis Prieto" 53 1.83)
f-2 (persona "Ana Perez" 56 1.7)

Agenda (MAIN)

0 nombre-amigos: f-2
0 nombre-amigos: f-1

Activaciones
(particularizaciones de reglas)

Primera: con f-2
El siguiente ciclo dispara la primera regla activada



Poniendo en marcha el motor de inferencias: run

- (run)
- Pone en marcha el ciclo básico de funcionamiento
 - Dispara la primera regla de la agenda
 - Determina nuevas reglas activadas
 - Reordena la agenda (si nuevas reglas activadas)
- El ciclo termina cuando no hay más reglas activadas
- En menú *Execution*.

CLIPS 6.3

File
Edit
Buffer
Execution
Browse
Window
Help

Dialog Window

CLIPS (6.30 3/17/15)

CLIPS> (load "D:/Docencia/GradoInformatica/IC/IGTerceroActual/03Practicas/Practica-C1-IntroduccionCLIPS-I/
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
Defining defrule: registrar-amigo +j+j+j
Defining defrule: registrar-amiga =j+j+j
TRUE
CLIPS> (reset)
CLIPS> (clear)
CLIPS> (load "D:/Docencia/GradoInformatica/IC/IGTerceroActual/03Practicas/Practica-C1-IntroduccionCLIPS-I/
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
TRUE
CLIPS> (reset)
CLIPS> (ppdefrule nombre-amigos)
(defrule MAIN::nombre-amigos "Encontrar el nombre de los amigos"
 (declare (salience 0))
 (persona ?x ? ?)
 =>
 (assert (encontrado ?x)))
CLIPS> (matches nombre-amigos)
Matches for Pattern 1
f-1
f-2
Activations
f-2
f-1
CLIPS> (run)
CLIPS>

Facts (MAIN)

f-0 (initial-fact)
f-1 (persona "Luis Prieto" 53 1.83)
f-2 (persona "Ana Perez" 56 1.7)
f-3 (encontrado "Ana Perez")
f-4 (encontrado "Luis Prieto")

Agenda (MAIN)



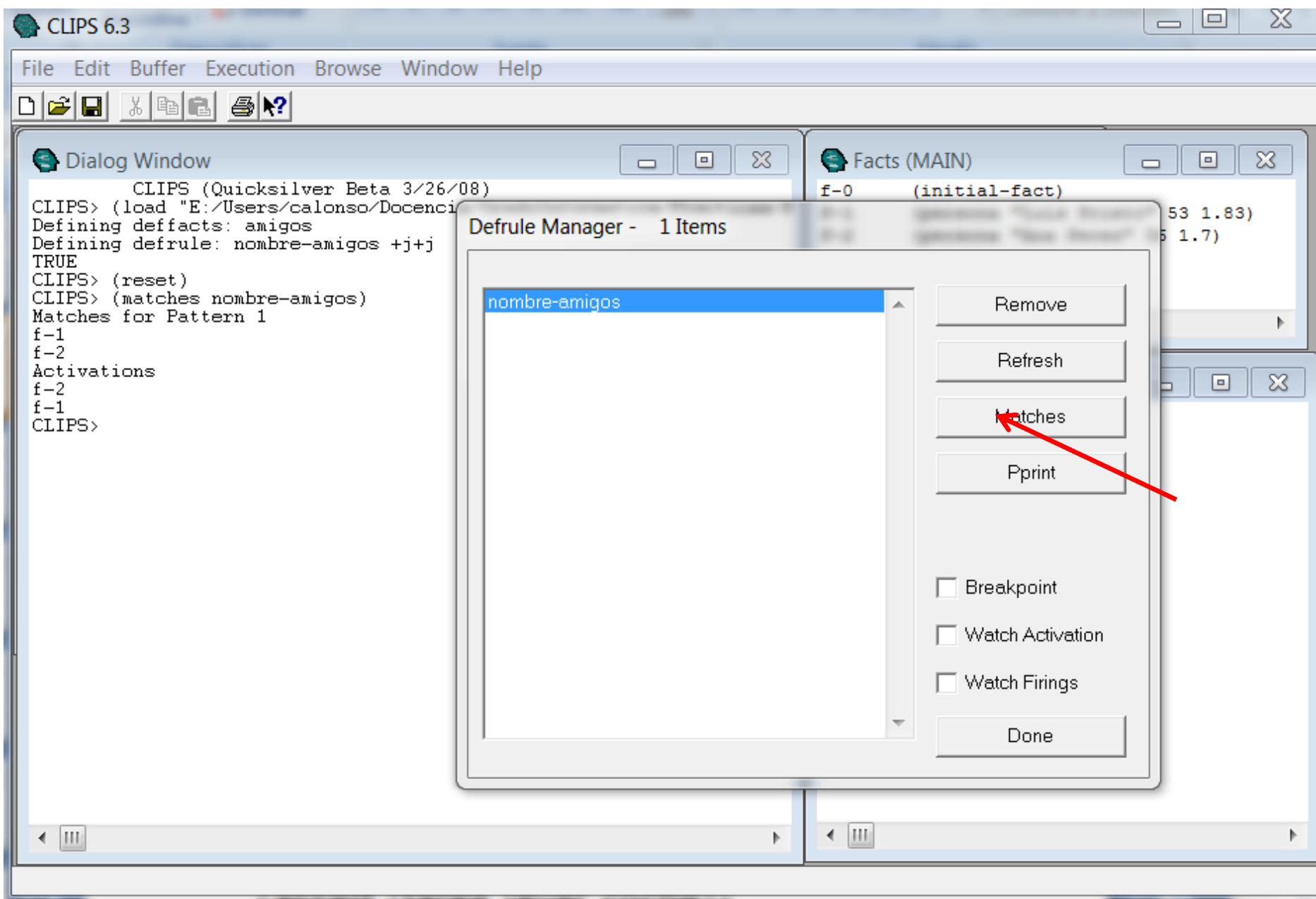
Volviendo al estado inicial: reset

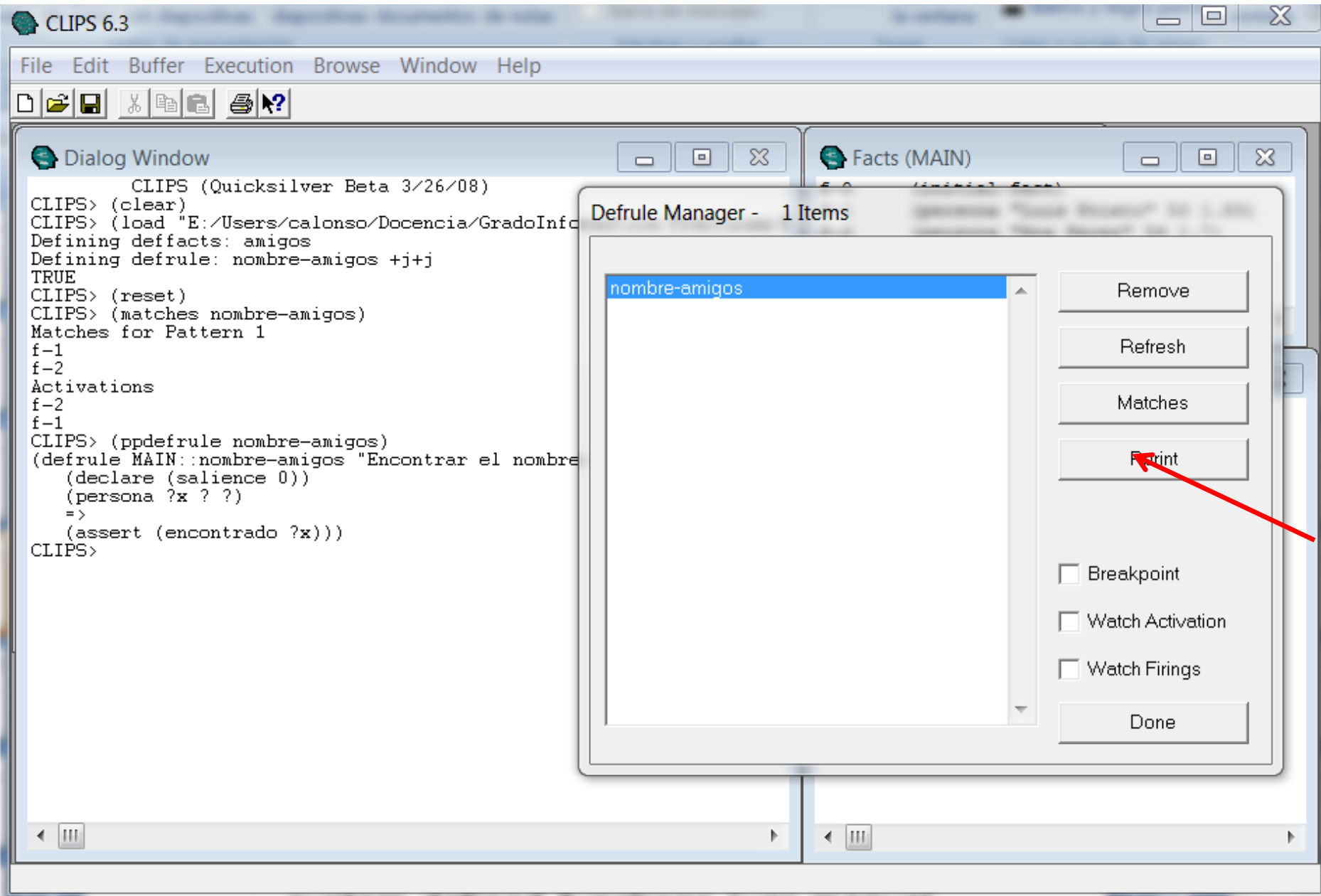
- (reset)
- Recordar:
 - Limpia memoria de trabajo
 - Añade hechos iniciales (constructor *deffacts*)
 - Determina reglas activadas
 - Ordena la agenda (resolución de conflictos)

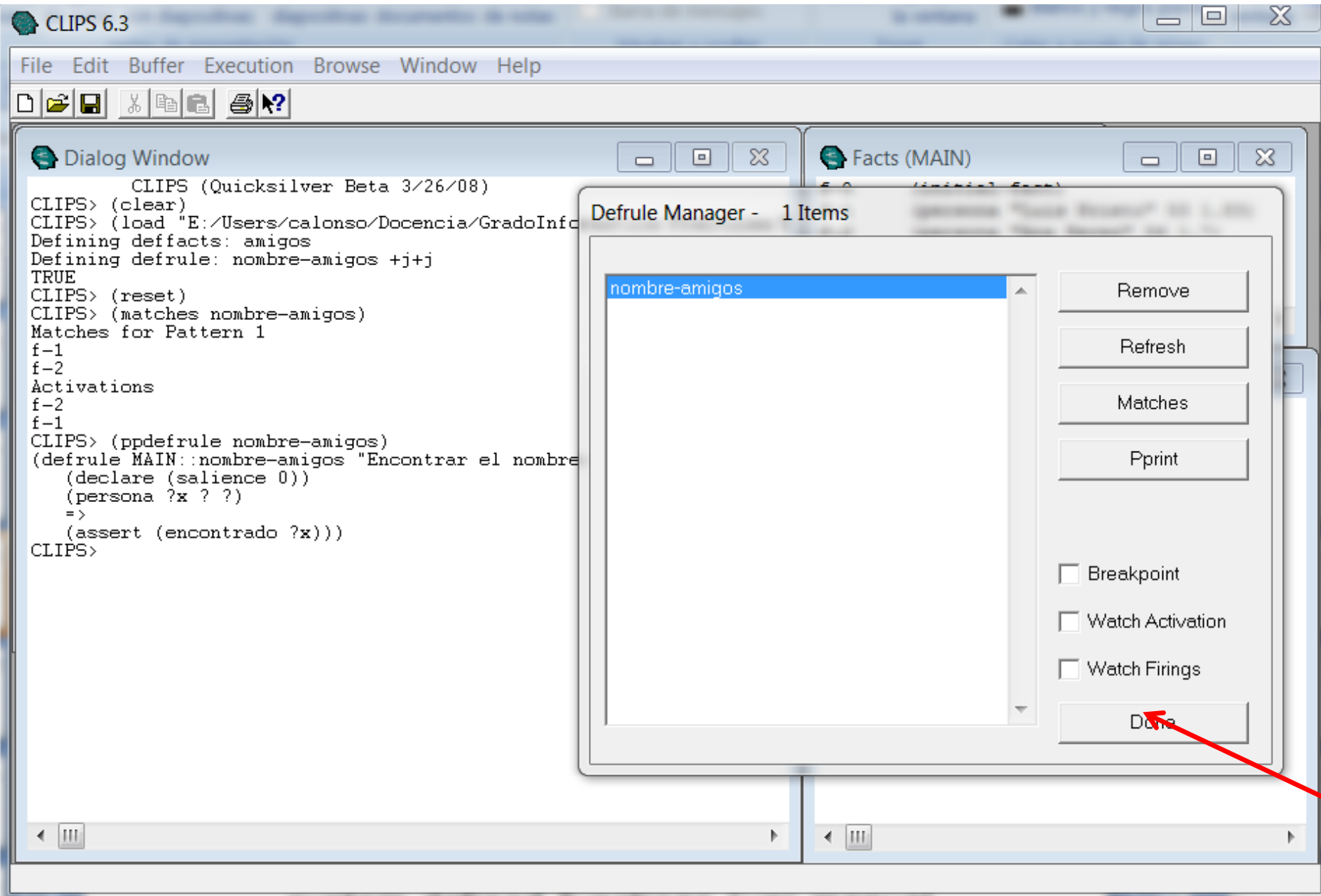


Examinar reglas y equiparación

- Defrule Manager
 - Acceso a la lista de reglas, por nombre
 - Observación, manipulación, trazado de reglas, equiparación de patrones y activaciones
 - Menú *Browse*







CLIPS 6.3

File Edit Buffer Execution Browse Window Help

Dialog Window

```
CLIPS (Quicksilver Beta 3/26/08)
CLIPS> (clear)
CLIPS> (load "E:/Users/calonso/Docencia/GradoInformatica/Practicas/F
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
TRUE
CLIPS> (reset)
CLIPS> (matches nombre-amigos)
Matches for Pattern 1
f-1
f-2
Activations
f-2
f-1
CLIPS> (ppdefrule nombre-amigos)
(defrule MAIN::nombre-amigos "Encontrar el nombre de los amigos"
  (declare (salience 0))
  (persona ?x ? ?)
  =>
  (assert (encontrado ?x)))
CLIPS>
```

Facts (MAIN)

```
f-0      (initial-fact)
f-1      (persona "Luis Prieto" 53 1.83)
f-2      (persona "Ana Perez" 56 1.7)
```

Agenda (MAIN)

```
0      nombre-amigos: f-2
0      nombre-amigos: f-1
```



Estrategia de resolución de conflictos

- Por defecto: depth
 - Las nuevas particularizaciones activadas se sitúan sobre todas las reglas con igual prioridad (saliency)
 - Se dispara la primera particularización entre las de mayor prioridad
- En el ejemplo
 - Primero activación f-2
 - Segundo activación f-1



Puesta en marcha del motor de inferencias

- (run)
 - Repetición del ciclo básico, hasta que no haya reglas activadas
- (run 1)
 - Dispara la primera regla de la agenda y realiza la fase de reconocimiento del siguiente ciclo (*step* en el menú).
- (reset)
 - Limpia la memoria de trabajo y evalúa deffacts
- Todos en menú Execution

CLIPS 6.3

File Edit Buffer Execution Browse Window Help

Dialog Window

```
CLIPS (Quicksilver Beta 3/26/08)
CLIPS> (clear)
CLIPS> (load "E:/Users/calonso/Docencia/GradoInformatica/Practicas/F
Defining deffacts: amigos
Defining defrule: nombre-amigos +j+j
TRUE
CLIPS> (reset)
CLIPS> (matches nombre-amigos)
Matches for Pattern 1
f-1
f-2
Activations
f-2
f-1
CLIPS> (ppdefrule nombre-amigos)
(defrule MAIN::nombre-amigos "Encontrar el nombre de los amigos"
  (declare (salience 0))
  (persona ?x ? ?)
  =>
  (assert (encontrado ?x)))
CLIPS> (run)
CLIPS>
```

Facts (MAIN)

```
f-0      (initial-fact)
f-1      (persona "Luis Prieto" 53 1.83)
f-2      (persona "Ana Perez" 56 1.7)
f-3      (encontrado "Ana Perez")
f-4      (encontrado "Luis Prieto")
```

Agenda (MAIN)

; Segundo ejemplo: AMIGOS y AMIGAS

; Hechos iniciales

(deffacts amigos "algunos amigos"

 (persona nombre "Luis Prieto" sexo varon) (persona nombre "Ana Perez" sexo mujer)
)

; Tres reglas

(defrule nombre-amigos "Encontrar el nombre de los amigos"

 (persona nombre ?x ? ?) => (assert (encontrado ?x))
)

(defrule registrar-amigo "Solo para los varones"

 (encontrado ?x)
 (persona nombre ?x sexo varon)
=>
 (assert (amigo ?x))
)

(defrule registrar-amiga "solo para las mujeres"

 (encontrado ?x)
 (persona nombre ?x sexo mujer)
=>
 (assert (amiga ?x))
)



Ejercicio 1

- Determinar el orden de disparo de las reglas, el contenido de la memoria de trabajo y el índice de cada hecho cuando el motor de inferencias se para, SIN utilizar CLIPS.
- Comprobar el resultado con CLIPS



4. Estrategias de resolución de conflictos

- Agenda: refleja el orden de disparo de las reglas según estrategia de resolución de conflictos.
- Por defecto:
 - Dispara una regla
 - La primera de la agenda
 - **SIEMPRE** REFRACCION (por ahora: no se repiten activaciones)
- Estrategia por defecto DEPTH
 - Las nuevas activaciones se sitúan sobre las reglas de igual prioridad, de forma arbitraria.



Otras estrategias de resolución de conflictos (1)

- Breath
 - Las nuevas activaciones se sitúan bajo las reglas de igual prioridad, de forma arbitraria.
- Simplicity, Complexity
 - Variantes de especificidad
 - No el número de patrones, sino en función del número de comparaciones en LHS (Left Hand Side).
- Estrategia LEX
 - Reciencia + especificidad
 - Se comparan los patrones uno a uno, empezando por los más recientes, hasta que algún patrón es más reciente.
 - Si todos patrones comunes iguales: más patrones.
 - Si todos patrones comunes iguales, igual número patrones, más específica.
 - Empate: arbitrario.



Otras estrategias de resolución de conflictos (2)

- Estrategia MEA
 - Reciencia del primer patrón.
 - Si empate, LEX



Ejercicio 2

- En un entorno de CLIPS vacío evaluamos los siguientes constructores en el orden proporcionado:

```
(defacts hechos-iniciales (hecho 1 ) (hecho 2 ) (hecho 3 ) (hecho 4 )  
  (hecho 5 ) (hecho 6 ))
```

```
(defrule regla-1 "hechos 1-2-3" (hecho 1) (hecho 2) (hecho 3)  
=> (assert (regla disparada 1)))
```

```
(defrule regla-2 "hechos 3-2" (hecho 3) (hecho 2)  
=> (assert (regla disparada 2)))
```

```
(defrule regla-3 "hecho 2-6-4" (hecho 2) (hecho 6) (hecho 4)  
=>(assert (regla disparada 3)))
```

```
(defrule regla-4 "hecho 6-4-3-2" (hecho 6) (hecho 4) (hecho 3) (hecho 2)  
=> (assert (regla disparada 4)))
```

- Determinar el orden de disparo de las reglas con cada una de las siguientes estrategias: Depth, Breath, LEX, MEA, Simplicity y Complexity.

Ejercicio 3

La figura 1 muestra un fragmento de una red causal que modela conocimiento del dominio para la tarea de diagnóstico en el dominio de los automóviles. La red asocia posibles causas de fallo –fusible fundido, batería baja o depósito de combustible vacío– con estados intermedios –potencia, combustible en motor– y síntomas –comportamiento motor, inspección fusible, indicador batería... –. Se puede observar que la red refleja la dirección causal: la causa “Depósito de combustible vacío” tiene como efecto “Combustible en motor falso” que a su vez es causa de “Comportamiento motor se para”.

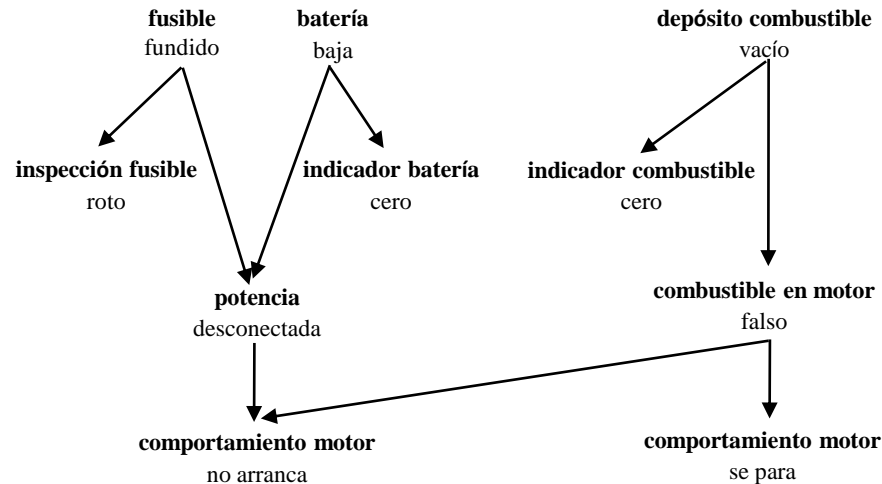


Figura 1



Ejemplo O-A-V (1)

- La conceptualización requiere identificar el conjunto de objetos y los atributos de cada objeto.
- El vocabulario está formado por los símbolos de Objetos, Atributos y Valores
- También debemos decir si el atributo es univaluado (solo puede tener un valor) o multivaluado (podría tener varios valores a la vez)
- Deberíamos conceptualizar el ejercicio anterior siguiendo ese formalismo.



Ejemplo O-A-V (2)

- Por tanto, lo primero sería establecer nuestros objetos
 $O = \{\text{fusible}, \text{batería}, \dots\}$
- Atributos, si son univaluados o multivaluados y sus posibles valores. Se expresan como objeto.atributo para saber cada atributo a qué objeto corresponde.
 $\text{bateria.nivel}^s = \{\text{baja}, \text{normal}\}$
 $\text{fusible.estado}^s = \{\text{correcto}, \text{fundido}\}$
- Si un atributo fuera multivaluado se expresaría de la siguiente forma:
 $\text{objeto.atributo}^m = 2^{\{\text{valor1}, \text{valor2}, \dots\}}$
- En el ejemplo del ejercicio anterior, cuando programemos en CLIPS mediante hechos ordenados, podemos tomar el objeto batería, con el atributo nivel y el valor bajo y escribir:
(bateria nivel bajo)
- Después, modelaríamos o conceptualizaríamos las reglas.
- Una vez tengamos la conceptualización completa, es más fácil obtener el código del programa.



Ejercicio 3 (Cont.)

a) A partir de la Red de la figura 1, elaborar una base de conocimiento basada en reglas utilizando el formalismo objeto-atributo-valor y un lenguaje proposicional (sin variables).

Sugerencia:

- Modelar en la dirección anti causal, -razonamiento abductivo o explicativo- de forma que a partir de los síntomas observables (los valores de los indicadores y el comportamiento del motor) las reglas permitan encontrar las causas que explican las observaciones.
- Proceder por etapas: i) de las quejas (comportamiento del motor) a las causas intermedias y ii) de las causas intermedias y las lecturas de los indicadores a las causas originales.

b) Codificar la base de reglas elaborada en a) en CLIPS, utilizando patrones ordenados.

c) Probar los siguientes escenarios:

1. Se observa que el motor no arranca y que el indicador de batería marca cero.
2. Se observa que el motor se para y que el indicador de combustible marca cero



Imprimir o leer

Imprimir (ver en el manual en línea)

- (printout <logical-name> <expression>*)
 - Para sacarlo por la salida por defecto (pantalla), usar **t** o **T** para logical-name.
 - **crLf** en cualquier lugar de expression fuerza retorno de carro/nueva_línea.

Leer (ver en el manual en línea)

- (read [<logical-name>])
 - Si logical-name es **t** o **T**, lee de la entrada por defecto (teclado)
 - Ejemplo: (assert (cadenadecaracteres = (read T)))



Ejercicio 4

- Considerar el asistente al diagnostico propuesto por Pool y Mackworth.
 - a) Elaborar una base de conocimiento basada en reglas utilizando el formalismo Objeto-Atributo-Valor con variables en los objetos y en los valores.

Atención: Clips no admite que el primer elemento de un patrón sea una variable.
Codificar los patrones como (atributo objeto valor)
 - b) Implementar la base anterior en CLIPS, utilizando patrones ordenados y reglas con variables.

Dominio: asistente al diagnóstico

