Sistemas Inteligentes

Práctica 1

Diseño, Implementación y Evaluación de un SBR (CLIPS)

Grupo 3G2

2019 / 2020

Objetivo

Diseñar y evaluar temporal y espacialmente un Sistema Basado en Reglas para un problema propuesto.

Se divide en 5 sesiones:

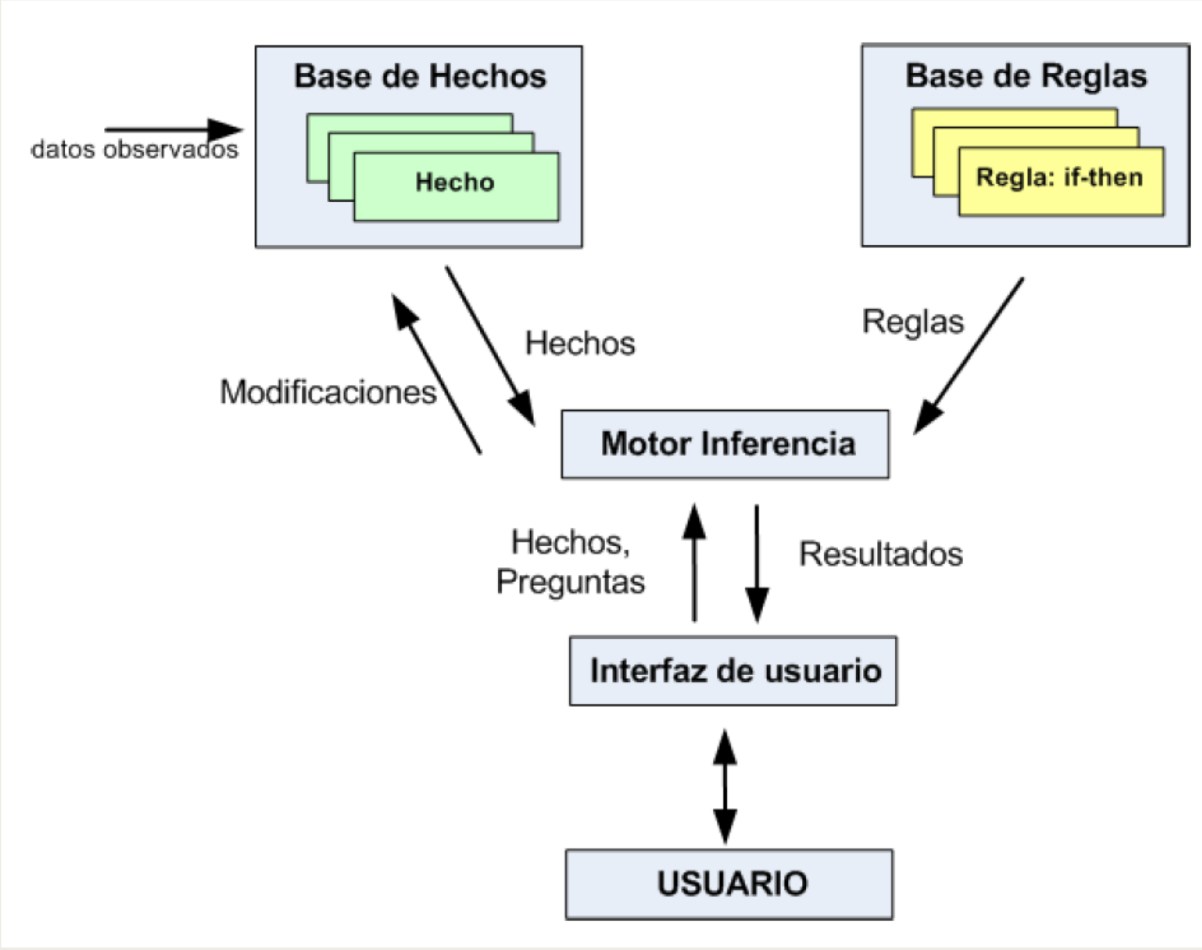
Semana 1	• Presentación Entorno CLIPS (ver Manual) • Probar Problemas de Ordenación y Mutaciones • Ver y probar el problema 8-Puzzle (Anchura y Profundidad)
Semana 2	• Planteamiento del problema a resolver.
Semanas 3 y 4	• Diseño e Implementación del Problema Propuesto como un SBR
Semana 5	• Evaluación INDIVIDUAL

Entorno CLIPS

Herramienta para el desarrollo de SBR (ver boletín CLIPS)

Utiliza los siguientes conceptos:

- Base de Hechos
  - hechos
- Base de Reglas
  - reglas
- Motor de Inferencia
  - control



Entorno CLIPS

Herramienta para el desarrollo de SBR (ver boletín CLIPS)

- **Definición inicial de hechos:**

```
(deffacts  datos
      (jarra nombre jarraX capacidad 4 contenido 0)
      (jarra nombre jarraY capacidad 3 contenido 3)
      (jarra nombre jarraX colores rojo azul)  )
```

- **Definición de variables globales:**

```
(defglobal      ?*nod-gen*    =   0))
```

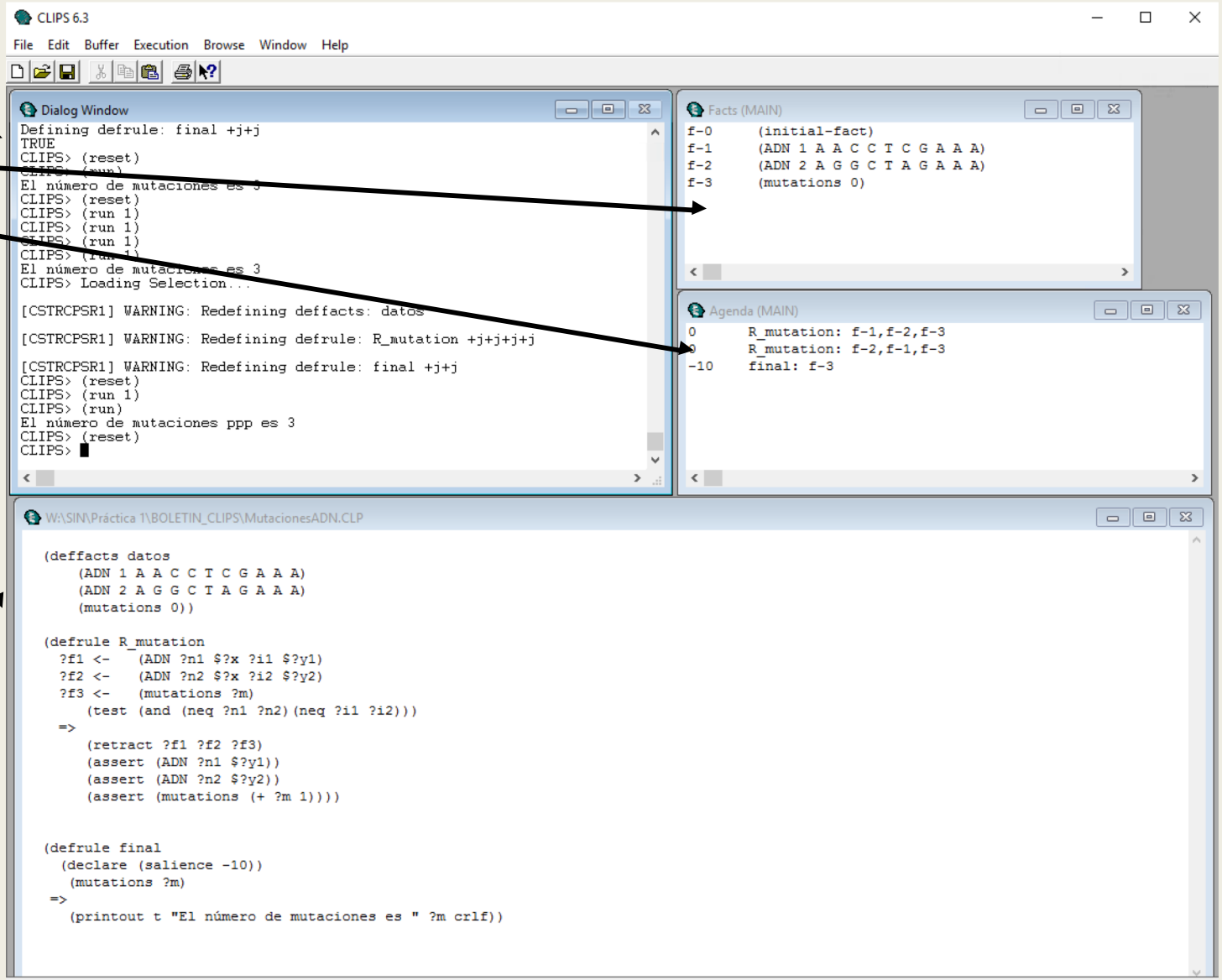
- **Definición de reglas:**

```
(defrule rellenar_jarra
            ?f <-  (jarra nombre ?name capacidad ?cap contenido ?cont)
            (test  (< ?cont ?cap))
=>
            (retract ?f)
            (assert (jarra nombre ?name capacidad ?cap contenido ?cap)  )
```

Entorno CLIPS

Se divide en varias ventanas:

- Dialog
- Facts
- Agenda
- Buffer



Entorno CLIPS

Carga y ejecución de ejemplos:

→ Utilizar la opción de menú “File / Load” o “File / Open” para cargar un fichero .clp

→ Ejecutar la opción de menú “Buffer / Load Buffer”

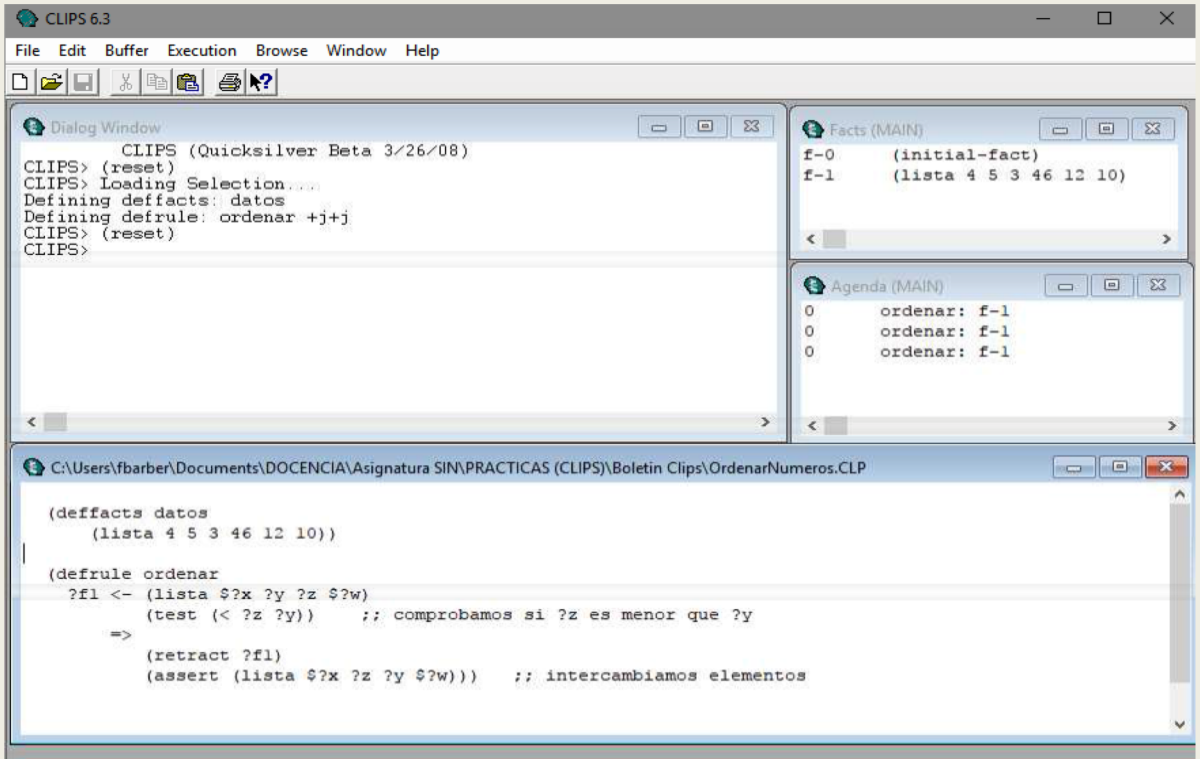
→ En la ventana de diálogo se puede ejecutar:

- → (reset) *inicializa el sistema (ej. carga hechos iniciales)*
- → (run) *ejecuta el sistema hasta que no hayan más reglas*
- → (run n) *ejecuta el sistema n pasos*
- → *(clear) limpia el buffer (utilizar previo a la carga de otro ejemplo)*

Ejemplo ordenación números

Ejemplo sencillo para ordenar una lista de números que se guarda como un hecho

```
(deffacts datos
    (lista 4 5 3 46 12 10))
(defrule ordenar
  ?f1 <- (lista $?x ?y ?z $?w)
   (test (< ?z ?y)) ;; comprobamos si ?z es menor que ?y
=>
   (retract ?f1)
   (assert (lista $?x ?z ?y $?w))) ;; intercambiamos elementos
```



Ejemplo Mutaciones ADN

El ejemplo compara dos secuencias de ADN y cuenta el número de mutaciones entre esas dos secuencias

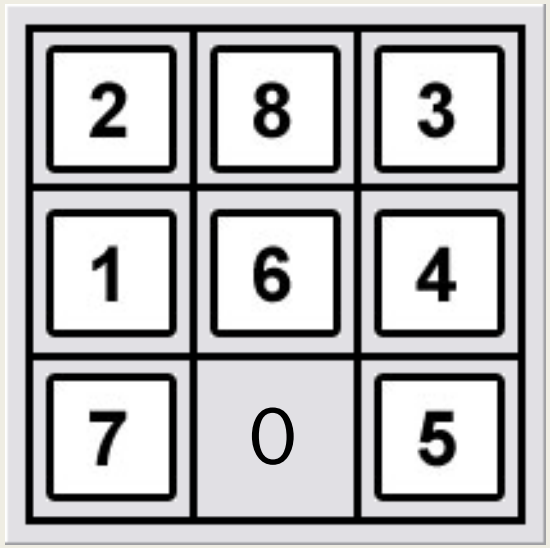
```
(deffacts datos
    (ADN 1 A A C C T C G A A A)
    (ADN 2 A G G C T A G A A A)
    (mutations 0))

(defrule R_mutation
 ?f1 <-  (ADN ?n1 $?x ?i1 $?y1)
 ?f2 <-  (ADN ?n2 $?x ?i2 $?y2)
 ?f3 <-  (mutations ?m)
         (test (and (neq ?n1 ?n2)(neq ?i1 ?i2)))
  =>
    (retract ?f1 ?f2 ?f3)
    (assert (ADN ?n1 $?y1))
    (assert (ADN ?n2 $?y2))
    (assert (mutations (+ ?m 1))))


(defrule final
  (declare (salience -10))
   (mutations ?m)
 =>  (printout  t "El número de mutaciones es " ?m crlf))
```

El problema del 8-puzzle

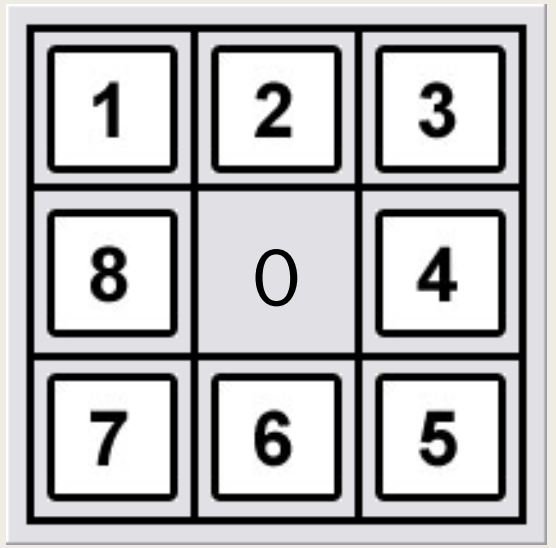
- Un n-puzzle es una matriz de números desde 0 (vacío) hasta n
- Un estado o configuración es una permutación de estos números
- El espacio libre se representa con el 0
- Las acciones consisten en mover (intercambiar) el 0 por otro número adyacente



Posible estado inicial

Proceso de búsqueda

Estado final



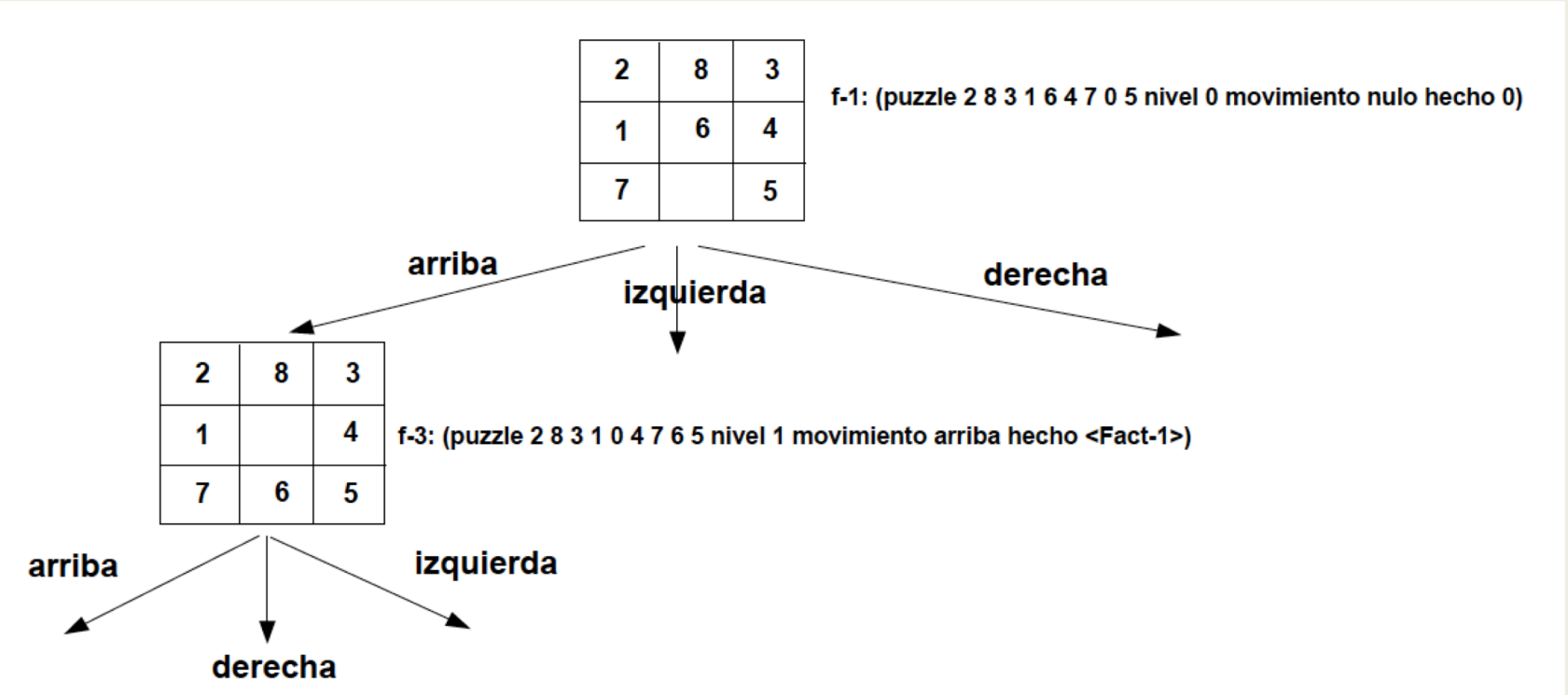
El problema del 8-puzzle

Representación del estado (**un único hecho**): *Estado, Nivel, Movimiento-Previo, Nodo-padre*

(puzzle *x1 x2 x3 x4 x5 x6 x7 x8 x9* nivel *y* movimiento *z* hecho *w*)

Ej: (puzzle 2 8 3 1 6 4 7 0 5 nivel 0 movimiento nulo hecho 0)

En realidad es un proceso de **búsqueda**



Estrategias posibles: **anchura o profundidad**

El problema del 8-puzzle

Reglas: 4 movimientos (arriba, abajo, izquierda, derecha)

```
(defrule arriba
  ?f<-(puzzle $?x  ?a  ?b  ?c  0  $?y   nivel  ?nivel   movimiento ?mov   hecho ?)
      (profundidad-maxima ?prof)              ;profundidad máxima
      (test (neq ?mov abajo))                 ;movimiento inverso
      (test (< ?nivel ?prof))
  =>
  (assert (puzzle $?x 0 ?b ?c ?a $?y    nivel (+ ?nivel 1)    movimiento arriba    hecho ?f))
  (bind ?*nod-gen* (+ ?*nod-gen* 1)))       ;nodos generados. Var. Global: (defglobal ?*nod-gen* = 0))
```

Regla meta: (puzzle 1 2 3 8 0 4 7 6 5 nivel ?nivel movimiento ?mov hecho ?h)

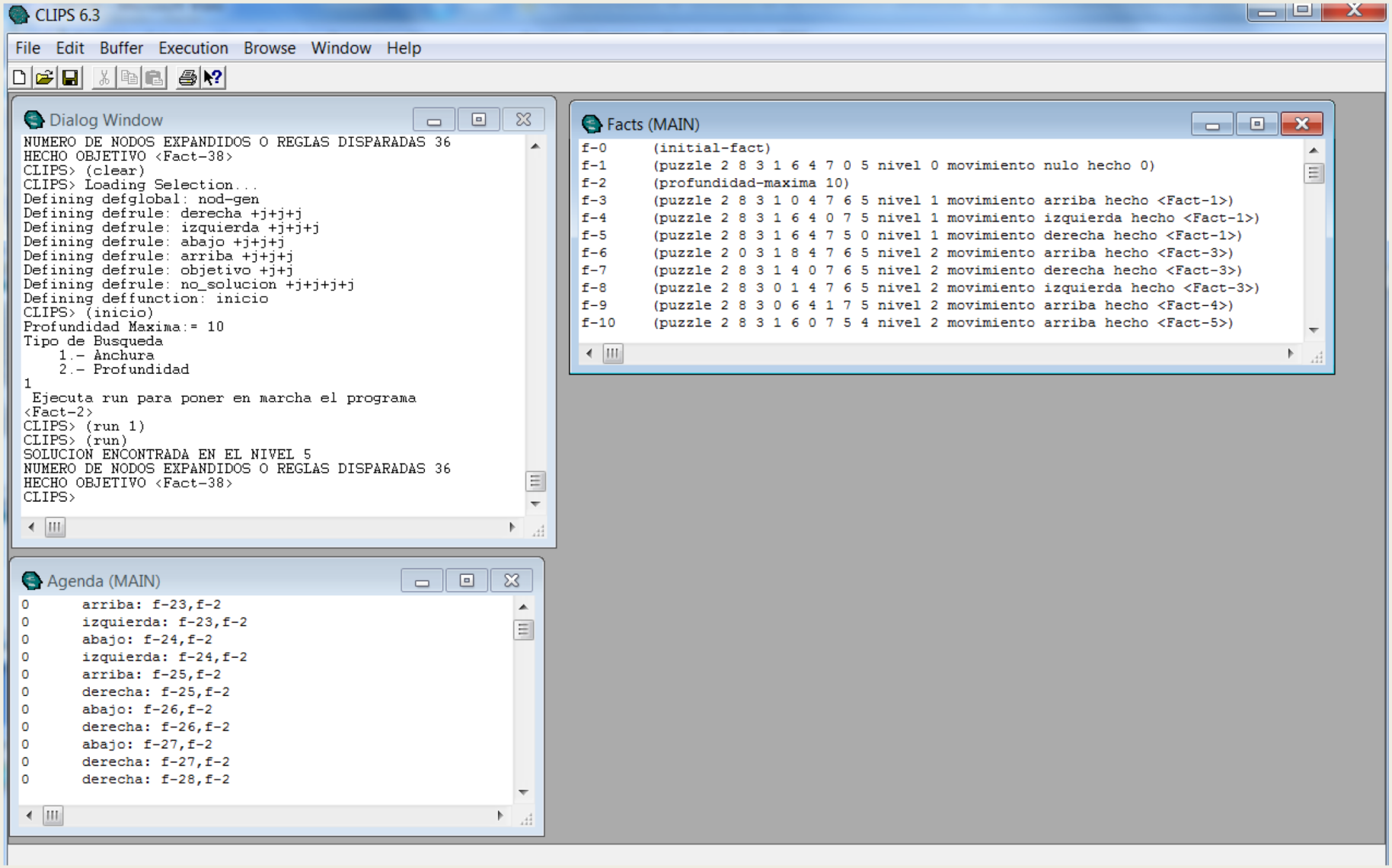
Lanzamiento del ejemplo: (inicio) ;determina bh-inicial (deffacts….), estrategia y prof máxima

Comprobar : (comprobar_conf (create$ 2 8 3 1 6 4 7 0 5)) ; Comprueba que sea resoluble

Ver el camino de la solucion: (camino *nº*) ; donde nº es el número del hecho final

El problema del 8-puzzle

Vamos a probarlo … cargar el fichero “puzando.clp”



Tareas

Probar entorno CLIPS

- Ejecutar ejemplo ordenación: modificar hecho inicial
- Ejecutar ejemplo mutación: modificar hechos iniciales

- Ejecutar ejemplo 8-puzzle:
  - Modificar estado inicial
  - Probar métodos de búsqueda: anchura y profundidad