

## Práctica 3 :

# Resolución de problemas y búsqueda con sistemas de producción

## 1. Objetivo de la práctica

El objetivo de esta práctica es coger experiencia en el desarrollo de programas en CLIPS tanto escribiendo código como depurando programas. Para ello, deberás escribir un programa CLIPS que resuelva el problema de las fichas. Utilizaremos el modulo de control para implementar la búsqueda A\*. Para una adecuada separación del módulo de control (MAIN) del resto de módulos puedes consultar el ejemplo del 8-puzzle de la lección de control en sistemas de producción.

## 2. Tareas

### 2.1 Problema 1 (6/10)

La situación inicial es

```
+---+---+---+---+---+---+---+
| B | B | B |   | V | V | V |
+---+---+---+---+---+---+---+
```

La situación final es

```
+---+---+---+---+---+---+---+
| V | V | V |   | B | B | B |
+---+---+---+---+---+---+---+
```

Los movimientos permitidos consisten en desplazar una ficha al hueco, saltando como máximo, sobre otras dos. Puedes partir del programa Puzzle.clp presentado en las transparencias y modificar la representación del estado y los operadores. Puedes utilizar la siguiente representación:

```
(deftemplate nodo
  (multislot estado)
  (multislot camino)
  (slot heuristica)
  (slot coste)
  (slot clase (default abierto)))

(defglobal MAIN
  ?*estado-inicial* = (create$ B B B H V V V))
```

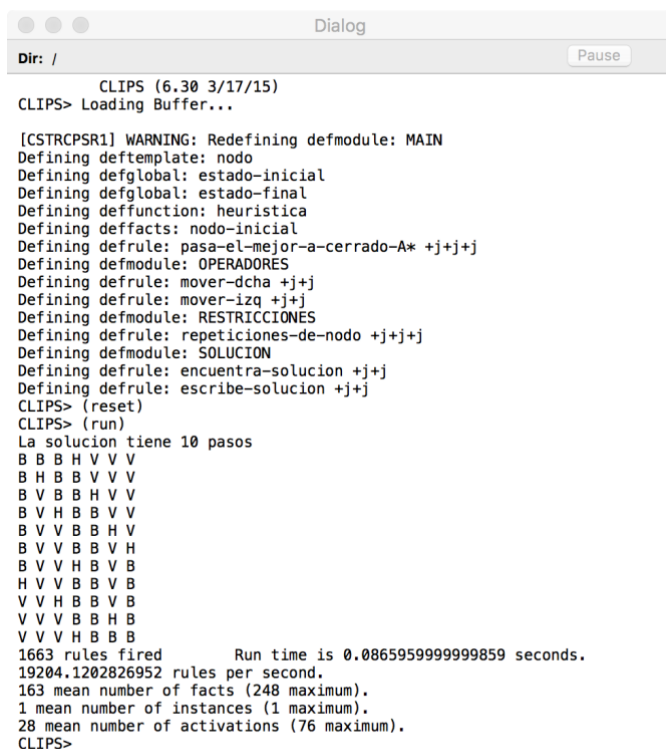
Funciones que pueden ser útiles: implode\$, explode\$, create\$, duplicate, loop-for-count ... En <http://clipsrules.sourceforge.net/OnlineDocs.html> encontrarás la documentación de CLIPS, aunque es suficiente con los ejemplos que encontrarás en las transparencias de clase.

En esta versión utilizaremos módulos: MAIN, OPERACIONES, RESTRICCIONES que detecta nodos repetidos y SOLUCION que reconoce la solución y escribe los pasos. El módulo MAIN implementa la búsqueda con heurística. Puedes utilizar la **heurística** que cuenta el número de fichas descolocadas.

Por ejemplo: La heurística del siguiente estado para h es 4.

```
+---+---+---+---+---+---+---+
| B | V | B |   | V | V | B |
+---+---+---+---+---+---+---+
```

Debes entregar un único fichero `fichas.clp` con los módulos mencionados y listo para ser ejecutado como en el ejemplo que se muestra a continuación:



```
Dialog
Dir: /
CLIPS (6.30 3/17/15)
CLIPS> Loading Buffer...

[CLSTRCPSR1] WARNING: Redefining defmodule: MAIN
Defining deftemplate: nodo
Defining defglobal: estado-inicial
Defining defglobal: estado-final
Defining deffunction: heuristica
Defining deffacts: nodo-inicial
Defining defrule: pasa-el-mejor-a-cerrado-A* +j+j+j
Defining defmodule: OPERADORES
Defining defrule: mover-dcha +j+j
Defining defrule: mover-izq +j+j
Defining defmodule: RESTRICCIONES
Defining defrule: repeticiones-de-nodo +j+j+j
Defining defmodule: SOLUCION
Defining defrule: encuentra-solucion +j+j
Defining defrule: escribe-solucion +j+j
CLIPS> (reset)
CLIPS> (run)
La solucion tiene 10 pasos
B B B H V V V
B H B B V V V
B V B B H V V
B V H B B V V
B V V B B H V
B V V B B V H
B V V H B V B
H V V B B V B
V V H B B V B
V V V B B H B
V V V H B B B
1663 rules fired          Run time is 0.086595999999859 seconds.
19204.1202826952 rules per second.
163 mean number of facts (248 maximum).
1 mean number of instances (1 maximum).
28 mean number of activations (76 maximum).
CLIPS>
```

## 2.2 Problema 2 (4/10)

Dada la siguiente representación del estado en CLIPS del problema de los misioneros y los caníbales, completa el código escribiendo las reglas que falten con el prefijo de módulo correspondiente para que realice una búsqueda A\*.

La representación del estado es:

- (listaM<sub>i</sub> s listaC<sub>i</sub> barca listaM<sub>f</sub> s listaC<sub>f</sub>), donde
- listaM<sub>i</sub> representa los misioneros en la orilla inicial, con tantas m<sub>s</sub> como misioneros en la orilla inicial.
- listaC<sub>i</sub> representa los caníbales en la orilla inicial, con tantas c<sub>s</sub> como caníbales en la orilla inicial.
- listaM<sub>f</sub> representa los misioneros en la orilla final, con tantas m<sub>s</sub> como misioneros en la orilla final.
- listaC<sub>f</sub> representa los caníbales en la orilla final, con tantas c<sub>s</sub> como caníbales en la orilla final.
- s se utiliza para separar los elementos de listaM<sub>i</sub> y listaC<sub>i</sub> o de listaM<sub>f</sub> y listaC<sub>f</sub>
- barca toma el valor i (orilla inicial) o f (orilla final).

El **estado inicial** es (m m m s c c c i s) y el **estado final** (s f m m m s c c c).

```
;-----  
; MODULO MAIN  
;-----  
(defmodule MAIN (export deftemplate nodo)  
                  (export deffunction heuristica)  
                  (export defglobal capacidad)  
                  (export deffunction problema))  
  
;;; Definición del nodo para la exploración  
(deftemplate MAIN::nodo  
  (multislot estado)  
  (multislot camino)  
  (slot heuristica)  
  (slot coste (default 0))  
  (slot clase (default abierto)))  
  
;;; Definición de variables globales y valores iniciales  
(defglobal MAIN  
  ?*estado-inicial* = (create$ m m m s c c c i)  
  ?*estado-final* = (create$ f m m m s c c c )  
  ?*ci* = (create$ )  
  ?*mi* = (create$ )  
  ?*misioneros* = 3  
  ?*canibales* = 3  
  ?*capacidad* = 2  
)  
  
;;; Definición de la heurística  
;;; numero de misioneros mas numero de caníbales    mi+ci  
(deffunction MAIN::heuristica (?mi ?ci)  
  (bind ?res (+ (length$ ?mi) (length$ ?ci)))  
  ?res)
```

```
;;; A* REGLA de CONTROL/Estrategia de búsqueda (COMPLETA)
```

```

;-----
; MODULO OPERADORES
;-----
; Acciones del Problema: 1M, 2C, 2M, 1C1M de la orilla inicial
; a la final 0 de la orilla final a la inicial
;-----
(defmodule OPERADORES
  (import MAIN deftemplate nodo)
  (import MAIN deffunction heuristica))

;;; 1C, 1M, 2C, 2M, 1C1M de la orilla inicial a la final
(defrule OPERADORES::MC-F
  (nodo (estado $?rmi $?mi s $?rci
        $?ci&:(and (>= (+ (length$ ?mi) (length$ ?ci)) 1)
                  (<= (+ (length$ ?mi) (length$ ?ci)) ?*capacidad*))
        i $?mf s $?cf)
   (camino $?movimientos) (coste ?coste) (clase cerrado))
=>
  (bind $?nuevoestado (create$ ?rmi s ?rci f ?mf ?mi s ?cf ?ci))
  (assert (nodo (estado ?nuevoestado)
                (camino ?movimientos (implode$ ?nuevoestado))
                (coste (+ ?coste 1))
                (heuristica (heuristica ?mi ?ci)))))

```

```

;;; 1C, 1M, 2C, 2M, 1C1M de la orilla final a la inicial (COMPLETA)

```

```

;-----
; MODULO RESTRICCIONES
;-----

(defmodule RESTRICCIONES
  (import MAIN deftemplate nodo))

```

```

; eliminamos nodos repetidos (COMPLETAR)

```

```

;; Eliminamos nodos con más caníbales que misioneros en una orilla (COMPLETAR)

```

```

;-----
; MODULO SOLUCION
;-----
;;; Función para definir problemas de distinto número
;;; de misioneros, caníbales, y capacidad de la barca.
;;; Por ejemplo:
;;; (problema 3 3 2)
;;; (problema 5 5 3)

(defmodule SOLUCION
  (import MAIN deftemplate nodo))

;miramos si hemos encontrado la solución
(defrule SOLUCION::encuentra-solucion
  (declare (auto-focus TRUE))
  ?nodo1 <- (nodo (estado s f $?m s $?c) (camino $?camino1))
=>
  (retract ?nodo1)
  (assert (solucion ?camino1)))

```

```

;escribimos la solución por pantalla
(defrule SOLUCION::escribe-solucion
  (solucion $?movimientos)
  =>
    (printout t "La solución tiene " (length$ ?movimientos) " pasos" crlf)
    (loop-for-count (?i 1 (length$ ?movimientos))
      (printout t (nth$ ?i $?movimientos) crlf))
    (halt))

;-----
; MODULO MAIN (última mención modulo MAIN, para
; poder invocar funciones de este módulo
;-----
;;; Para ejecutar el programa
;;; (load "misioneros.clp")
;;; (problema 3 3 2)

(deffunction MAIN::problema (?m ?c ?b)
  (reset)
  (bind ?*misioneros* ?m)
  (bind ?*canibales* ?c)
  (bind ?*capacidad* ?b)
  (loop-for-count (?i 1 ?m)
    (bind ?*mi* (create$ ?*mi* m)))
  (loop-for-count (?i 1 ?c)
    (bind ?*ci* (create$ ?*ci* c)))
  (bind ?*estado-inicial* (create$ ?*mi* s ?*ci* i s))
  (bind ?*estado-final* (create$ s f ?*mi* s ?*ci*))
  (assert (nodo (estado ?*estado-inicial*)
    (camino (implode$ ?*estado-inicial*))
    (heuristica (heuristica ?*ci* ?*mi*))
    ))
  (run)
)

```

Debes entregar un único fichero `misioneros.clp` con los módulos mencionados y listo para ser ejecutado como en el ejemplo que se muestra a continuación:

```

CLIPS> Loading Buffer...
+%:::!!*+*+*+*+*+*
CLIPS> (problema 5 5 3)
<Fact-1>
CLIPS> (problema 3 3 2)
La solución tiene 12 pasos
m m m s c c c i s
m m m s c f s c c
m m m s c c i s c
m m m s f s c c c
m m m s c i s c c
m s c f m m s c c
m m s c c i m s c
s c c f m m m s c
s c c c i m m m s
s c f m m m s c c
s c c i m m m s c
s f m m m s c c c

```