

Ejercicios

Lenguajes de Reglas

Estos ejercicios trabajo tienen por objeto familiarizarse con en el desarrollo de programas en CLIPS tanto escribiendo código como depurando programas. Para ello, deberás escribir programas CLIPS que resuelvan los problemas planteados con los algoritmos que se piden. Utilizaremos el módulo de control para implementar los distintos algoritmos (A*, CU, voraz). 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.

1 Problema Fichas en CLIPS

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)))
```

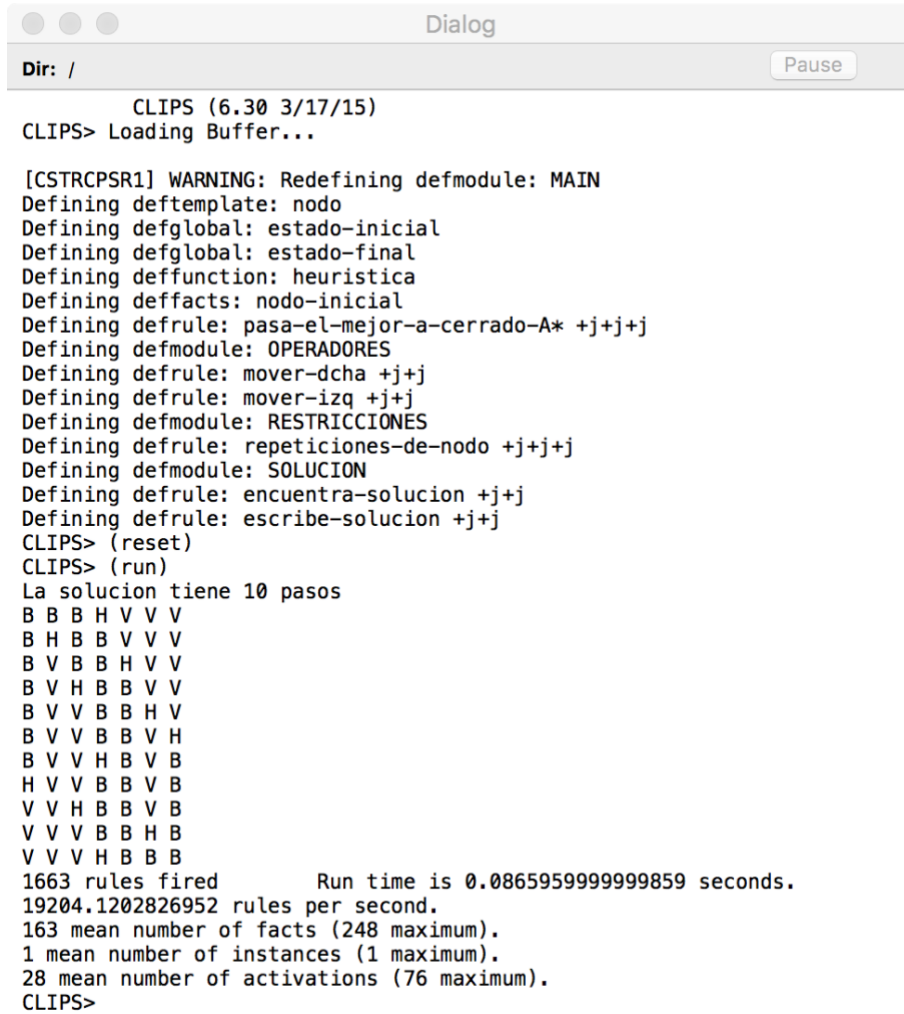
```
(defglobal MAIN
  ?*estado-inicial* = (create$ B B B H V V V))
```

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. Debes implementar un A* en grafo. 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...

[CSTRCPSR1] 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 0.0865959999999859 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>
```

Problema del Granjero en CLIPS

Resuelve el problema del granjero, el lobo, la cabra y la col. Puedes utilizar el código de las transparencias utilizado en clase. Pero 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 Coste Uniforme en grafo.

```
CLIPS> (reset)
CLIPS> (run)
```

Solución encontrada

```
El granjero se mueve con la cabra a la derecha.
El granjero se mueve solo a la izquierda.
El granjero se mueve con el lobo a la derecha.
El granjero se mueve con la cabra a la izquierda.
El granjero se mueve con la col a la derecha.
El granjero se mueve solo a la izquierda.
El granjero se mueve con la cabra a la derecha.
```

Solución encontrada

```
El granjero se mueve con la cabra a la derecha.
El granjero se mueve solo a la izquierda.
El granjero se mueve con la col a la derecha.
El granjero se mueve con la cabra a la izquierda.
El granjero se mueve con el lobo a la derecha.
El granjero se mueve solo a la izquierda.
El granjero se mueve con la cabra a la derecha.
69 rules fired          Run time is 0.0007781982421875 seconds.
88666.3529411765 rules per second.
7 mean number of facts (10 maximum).
0 mean number of instances (0 maximum).
6 mean number of activations (11 maximum).
```

Problema Ejercicio 30 colección problemas en CLIPS

La siguiente figura representa el estado inicial con cinco monedas colocadas en línea donde la O indica CARA y la C CRUZ:

O	C	O	C	O
---	---	---	---	---

Los **movimientos posibles de coste 1** son dar la vuelta a dos monedas contiguas. El **objetivo** del problema es llegar al estado siguiente:

C	C	C	O	C
---	---	---	---	---

Dada la **función heurística** $h(n) = \text{número de monedas mal colocadas}$

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$ O C O C O))
```

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.

Debes definir los 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. Debes implementar un A* en grafo. Puedes utilizar la **heurística** que cuenta el número de fichas descolocadas. Por ejemplo: La heurística del estado inicial es 4.

Debes entregar un único fichero Ej30.clp con los módulos mencionados y listo para ser ejecutado como en el ejemplo que se muestra a continuación:

