Práctica 1

Técnicas de Inteligencia Artificial

Curso 4

Andoni Martín Reboredo

David Ramirez Ambrosi

4-11-2014

1. Representación del problema del barquero

Un estado se define como una tupla (p, c, l, b) donde p, c, l, b toman los valores {i,d}. Los elementos representan al puma, la cabra, la lechuga y el barquero respectivamente.

El estado inicial se define como (i, i, i, i), mientras que el estado objetivo es (d, d, d, d).

A continuación se documentan las funciones utilizadasen el fichero "*estados-barquero.clp*":

igualBarquero? (?pos $?estado)	**Precondición:** **La posición enviada como parámetro es una posición válida para el sistema, es decir, está comprendida entre 1 y 3. El estado recibido debe de ser válido.** **Postcondición:** **Devuelve TRUE si el elemento de la posición indicada está en la misma posición que el barquero, FALSE en caso cotrario.**
exito ($?estado)	**Precondición:** **El estado obtenido debe ser válido.** **Postcondición:** **Devuelve TRUE si el estado recibido es el estado que se desea obtener dentro del problema.**
fracaso ($?estado)	**Precondición:** **Recibe un estado válido.** **Postcondición:** **Devuelve PROHIBIDO si el estado recibido como parámetro en un estado fallido según la definición del problema, el estado recibido en cualquier otro caso.**
contrario (?pos)	**Precondición:** **Recibe "i" o "d".** **Postcondición:** **Devuelve "i" si el parámetro recibido es "d" e "i" en caso contrario.**
MoverP ($?estado)	**Precondición:** **Recibe un estado válido.** **Postcondición:** **Si es posible mover al puma retorna el estado resultante de mover al puma incluyendo al barquero, si no es posible devuelve PROHIBIDO.**
MoverC ($?estado)	**Precondición:** **Recibe un estado válido.**

	Postcondición: **Si es posible mover a la cabra devuelve el estado que resulta de mover a la cabra de sitio incluyendo el movimiento del barquero, si no es posible, devuelve PROHIBIDO.**
MoverL ($?estado)	**Precondición:** **Recibe un estado válido.** **Postcondición:** **Si el estado recibido permite mover de sitio la lechuga la función la mueve moviendo además al barquero, si no es posible la función devuelve PROHIBIDO.**
MoverB ($?estado)	**Precondición:** **Recibe un estado válido.** **Postcondición:** **Mueve el barquero de orilla devolviendo el estado nuevo si es es posible mover al barquero sin violar las reglas del problema y PROHIBIDO si el estado objetivo no es posible.**

2. Resultados de los diferentes algoritmos

2.1. Búsqueda en profundidad

Ejecución:

CuÃ¡ntos pasos quiere ejecutar? si quiere todos escribir 0 y si el mismo -1
0
Â¿QuÃ© algoritmo quiere aplicar?

1.BÃºsqueda-en-profundidad-sin-visitados
2.BÃºsqueda-en-profundidad-con-visitados
3.BÃºsqueda-en-anchura-sin-visitados
4.BÃºsqueda-en-anchura-con-visitados

1
Paso 0
Padre (i i i i)
Paso 1
Padre (i d i d)
Paso 2
Padre (i i i i)
Paso 3
Padre (i d i d)
Paso 4
Padre (i i i i)
Paso 5
Padre (i d i d)
Paso 6
Padre (i i i i)
Paso 7
Padre (i d i d)
Paso 8
Padre (i i i i)
Paso 9
Padre (i d i d)
Paso 10
Padre (i i i i)
Paso 11
Padre (i d i d)
Paso 12
Padre (i i i i)
Paso 13
Padre (i d i d)
Paso 14

Padre (i i i i)
Paso 15
Padre (i d i d)
Paso 16
Padre (i i i i)
Paso 17
Padre (i d i d)
Paso 18
Padre (i i i i)
Paso 19
Padre (i d i d)
Paso 20
Padre (i i i i)
Paso 21
Padre (i d i d)
Paso 22
Padre (i i i i)

Esta ejecución no llega a obtener una solución. Cicla continuamente ejecutando la operación moverC. Lo que ocurre es que esta operación es la segunda de la lista de operaciones. La primera, moverP, no se puede ejecutar porque se quedaría sola la cabra con la lechuga. Entonces, y puesto que no se examina que se puedan repetir los estados, la única operación que llegamos a ejecutar mediante la búsqueda en profundidad es la de mover la cabra de una orilla a la contraria.

Hay que destacar que sí se puede ejecutar la operación mover barquero una vez que se ha movido la cabra a la orilla derecha. Sin embargo, debido al orden de las operaciones comentado y al tratamiento de los estados por recorrer mediante una pila, el estado desde el que avanzamos es siempre el resultante de mover la cabra de una orilla a otra.

2.2. Búsqueda en profundidad con visitados

Ejecución:

CuÃ¡ntos pasos quiere ejecutar? si quiere todos escribir 0 y si el mismo -1
0
Â¿QuÃ© algoritmo quiere aplicar?

1.BÃºsqueda-en-profundidad-sin-visitados
2.BÃºsqueda-en-profundidad-con-visitados
3.BÃºsqueda-en-anchura-sin-visitados
4.BÃºsqueda-en-anchura-con-visitados

2
Paso 0
Padre (i i i i)
Paso 1
Padre (i d i d)
Paso 2
Padre (i i i i)
Paso 2
Padre (i d i i)
Paso 3
Padre (d d i d)
Paso 4
Padre (i d i i)
Paso 4
Padre (d i i i)
Paso 5
Padre (d i d d)
Paso 6
Padre (i i d i)
Paso 7
Padre (d i d d)
Paso 7
Padre (i d d d)
Paso 8
Padre (i d i i)
Paso 8
Padre (i i d i)
Paso 8
Padre (d i i i)
Paso 8
Padre (d i d i)
Paso 9
Padre (d d d d)
La soluciÃ³n es (d d d d)

En este caso sí obtenemos solución. El algoritmo actúa de forma similar al anterior, pero en este caso antes de examinar los estados hijos de un estado padre, comprobamos que ese estado padre no haya sido examinado ya, de forma que no se da el problema del ciclo del apartado anterior y el algoritmo continúa explorando otros estados que han sido alcanzados por operaciones distintas a “moverC”.

Cuando se muestra paso 2 varias veces, indica que el estado correspondiente al primer paso 2 ya había sido examinado previamente y el algoritmo continúa en el paso 2 ya que no ha examinado los estados hijos del mismo. Esta exploración se lleva a cabo en los estados que al partir de un estado X llevan a un estado X+1.

2.3. Búsqueda en anchura

Ejecución:

> *CuÃ¡ntos pasos quiere ejecutar? si quiere todos escribir 0 y si el mismo -1*
> *0*
> *Â¿QuÃ© algoritmo quiere aplicar?*
>
> *1.BÃºsqueda-en-profundidad-sin-visitados*
> *2.BÃºsqueda-en-profundidad-con-visitados*
> *3.BÃºsqueda-en-anchura-sin-visitados*
> *4.BÃºsqueda-en-anchura-con-visitados*

3
Paso 0
Padre (i i i i)
Paso 1
Padre (i d i d)
Paso 2
Padre (i i i i)
Paso 3
Padre (i d i i)
Paso 4
Padre (i d i d)
Paso 5
Padre (d d i d)
Paso 6
Padre (i d d d)
Paso 7
Padre (i d i d)
Paso 8
Padre (i i i i)
Paso 9
Padre (i d i i)
Paso 10
Padre (i d i i)
Paso 11
Padre (d i i i)
Paso 12
Padre (i d i i)
Paso 13
Padre (i i d i)
Paso 14
Padre (i i i i)
Paso 15
Padre (i d i i)
Paso 16
Padre (i d i d)
Paso 17
Padre (d d i d)
Paso 18
Padre (i d d d)
Paso 19
Padre (i d i d)
Paso 20
Padre (d d i d)
Paso 21
Padre (i d d d)
Paso 22
Padre (i d i d)
Paso 23
Padre (d i d d)
Paso 24
Padre (d d i d)
Paso 25
Padre (d d i d)
Paso 26
Padre (i d d d)
Paso 27
Padre (i d i d)
Paso 28
Padre (d i d d)
Paso 29
Padre (i d d d)
Paso 30
Padre (i d i d)
Paso 31
Padre (d d i d)
Paso 32
Padre (i d d d)
Paso 33
Padre (i d i d)
Paso 34
Padre (i i i i)
Paso 35
Padre (i d i i)
Paso 36
Padre (i d i i)
Paso 37
Padre (d i i i)
Paso 38
Padre (i d i i)
Paso 39
Padre (i i d i)
Paso 40
Padre (i i i i)
Paso 41
Padre (i d i i)
Paso 42
Padre (i d i i)
Paso 43
Padre (d i i i)
Paso 44
Padre (i d i i)
Paso 45
Padre (i i d i)
Paso 46
Padre (i i i i)
Paso 47

Padre (i d i i)
Paso 48
Padre (i i d i)
Paso 49
Padre (d i i i)
Paso 50
Padre (d i d i)
Paso 51
Padre (i d i i)
Paso 52
Padre (d i i i)
Paso 53
Padre (i d i i)
Paso 54
Padre (d i i i)
Paso 55
Padre (i d i i)
Paso 56
Padre (i i d i)
Paso 57
Padre (i i i i)
Paso 58
Padre (i d i i)
Paso 59
Padre (i i d i)
Paso 60
Padre (d i i i)
Paso 61
Padre (d i d i)
Paso 62
Padre (i d i i)
Paso 63
Padre (i i d i)
Paso 64
Padre (i i i i)
Paso 65
Padre (i d i i)
Paso 66
Padre (i d i i)
Paso 67
Padre (d i i i)
Paso 68
Padre (i d i i)
Paso 69
Padre (i i d i)
Paso 70
Padre (i i i i)
Paso 71
Padre (i d i i)
Paso 72
Padre (i d i d)
Paso 73
Padre (d d i d)
Paso 74
Padre (i d d d)
Paso 75
Padre (i d i d)
Paso 76
Padre (d d i d)
Paso 77
Padre (i d d d)
Paso 78
Padre (i d i d)
Paso 79
Padre (d i d d)
Paso 80
Padre (d d i d)
Paso 81
Padre (d d i d)
Paso 82
Padre (i d d d)
Paso 83
Padre (i d i d)
Paso 84
Padre (d i d d)
Paso 85
Padre (i d d d)
Paso 86
Padre (i d i d)
Paso 87
Padre (d d i d)
Paso 88
Padre (i d d d)
Paso 89
Padre (i d i d)
Paso 90
Padre (d d i d)
Paso 91
Padre (i d d d)
Paso 92
Padre (i d i d)
Paso 93
Padre (d i d d)
Paso 94
Padre (d d i d)
Paso 95
Padre (d d i d)
Paso 96
Padre (i d d d)
Paso 97
Padre (i d i d)
Paso 98
Padre (d i d d)
Paso 99
Padre (i d d d)
Paso 100
Padre (i d i d)
Paso 101
Padre (d d i d)
Paso 102
Padre (i d d d)
Paso 103
Padre (i d i d)
Paso 104
Padre (d i d d)
Paso 105
Padre (i d d d)
Paso 106
Padre (d i d d)
Paso 107
Padre (d d i d)
Paso 108
Padre (d d d d)
La soluciÃ³n es (d d d d)

De forma similar a lo sucedido en el caso de búsqueda en profundidad sin visitados, en la búsqueda en anchura sin visitados se dan ciclos que hacen que el algoritmo tenga que ejecutar muchas operaciones previamente ejecutadas. Sin embargo, y debido a la propia búsqueda en anchura, sí llegamos a alcanzar una solución, que además es la que menos pasos requiere para alcanzar el objetivo. Nótese que el algoritmo implementado utiliza el concepto “pasos” como los estados que son necesarios examinar hasta alcanzar la solución óptima frente al número de operaciones

requeridas para alcanzar la solución óptima, que es el valor del cual la búsqueda en anchura garantiza encontrar el mínimo.

2.4. Búsqueda en anchura con visitados

Ejecución:

> *CuÃ¡ntos pasos quiere ejecutar? si quiere todos escribir 0 y si el mismo -1*
> *0*
> *Â¿QuÃ© algoritmo quiere aplicar?*
>
> *1.BÃºsqueda-en-profundidad-sin-visitados*
> *2.BÃºsqueda-en-profundidad-con-visitados*
> *3.BÃºsqueda-en-anchura-sin-visitados*
> *4.BÃºsqueda-en-anchura-con-visitados*
>
> *4*
> *Paso 0*
> *Padre (i i i i)*
> *Paso 1*
> *Padre (i d i d)*
> *Paso 2*
> *Padre (i i i i)*
> *Paso 2*
> *Padre (i d i i)*
> *Paso 3*
> *Padre (d d i d)*
> *Paso 4*
> *Padre (i d d d)*
> *Paso 5*
> *Padre (i d i d)*
> *Paso 5*
> *Padre (i d i i)*
> *Paso 5*
> *Padre (d i i i)*
> *Paso 6*
> *Padre (i d i i)*
> *Paso 6*
> *Padre (i i d i)*
> *Paso 7*
> *Padre (d i d d)*
> *Paso 8*
> *Padre (d d i d)*
> *Paso 8*
> *Padre (d i d d)*
> *Paso 8*
> *Padre (i d d d)*
> *Paso 8*
> *Padre (i d d d)*
> *Paso 8*
> *Padre (i i d i)*
> *Paso 8*
> *Padre (d i i i)*
> *Paso 8*
> *Padre (d i d i)*
> *Paso 9*
> *Padre (d d d d)*
> *La soluciÃ³n es (d d d d)*

Al igual que lo sucedido al registrar los estados visitados en el caso del recorrido en profundidad, introducir el registro de los estados que ya han sido examinados en la búsqueda en anchura nos ahorra examinar varias veces los mismos estados.

Debido a esto el número de estados que llega a examinar el algoritomo se reduce de 108 a 9, obteniendo una solución equivalente a la del algoritmo anterior en cuanto a número de operaciones eecutadas necesarias para alcanzar la solución óptima se refiere.

3. Función minimo-de-lista

El reto en este apartado era conseguir ejecutar la función “min” de CLIPS teniendo como parámetro un campo multivalor no aceptado por la función.

La solución:

```
(eval (format nil "(%s %s)" "min" (implode$ ?lista))))
```

Construimos el comando clips a ejecutar en la línea de comandos mediante la función “format” en la que indicamos la cadena a construir (concatenación de dos strings “%s %s”) y los strings que la componen que son “min”, la función que queremos ejecutar, y el campo multivalor transformado a

string mediante la función “implode$”.

Una vez construido el comando, lo ejecutamos mediante la función “eval” para obtener finalmente el elemento mínimo mediante la función “min” de un campo multivalor.