



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS
INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática



Dpto. Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

SISTEMAS INTELIGENTES

PRÁCTICA 1

DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA BASADO EN REGLAS

- Problema a resolver -

El juego Sokoban




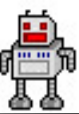



Octubre 2018

Sokoban

Sokoban es un rompecabezas inventado en Japón que significa “encargado de almacén” en japonés. El objetivo del juego es empujar una serie de cajas distribuidas en un grid hasta un almacén.

Para explicar las reglas y condiciones del juego utilizaremos como ejemplo el grid que se muestra en la figura aunque el programa a desarrollar **debe ser válido para cualquier configuración de grid**.

El grid de la figura contiene varios obstáculos (representados mediante celdas de color gris), tres cajas, tres almacenes y un robot (jugador). Las celdas del grid se referencian mediante dos números que indican la fila y la columna respectivamente. Así, por ejemplo, el robot está en la posición (4,1), hay un almacén en la posición (1,7), una caja en la (4,3), un obstáculo en la (3,5), etc.

	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								

El robot puede moverse a una casilla adyacente en una de estas 4 direcciones: arriba, abajo, derecha e izquierda. El robot puede asimismo empujar una caja a la casilla adyacente de arriba, de abajo, a la izquierda o la derecha.

El robot se puede desplazar a una celda adyacente en cualquiera de las 4 direcciones si:

- La casilla destino no contiene un obstáculo ni una caja
- La casilla destino no es un almacén

Para empujar una caja, el robot debe situarse en una casilla adyacente a la caja y solo puede empujar la caja a una casilla que no tenga nada o bien a una casilla que sea un

almacén. En el ejemplo de la figura, para empujar la caja situada en la celda (4,3), el jugador puede:

- situarse en la casilla (5,3) y empujar la caja hacia arriba; el efecto de esta operación es que tanto la caja como el robot se desplazan una fila hacia arriba; es decir, el robot terminará en la posición (4,3) y la caja en la posición (3,3)
- situarse en la casilla (3,3) y empujar la caja hacia abajo; el efecto de esta operación es que tanto la caja como el robot se desplazan una fila hacia abajo
- no es posible situarse en la casilla a la derecha de la caja porque hay un obstáculo
- situarse en la casilla a la izquierda pero en este caso el robot no puede empujar la caja hacia la derecha porque hay un obstáculo

Los almacenes tienen capacidad para una caja de modo que cuando el robot empuja una caja a un almacén, este se llena y ya no podrá almacenar más cajas.

Se pide resolver esta versión del juego de Sokoban utilizando un diseño basado en estados mediante un SBR implementado en CLIPS. Como estrategias de búsqueda para resolver el SBR se empleará ANCHURA y PROFUNDIDAD.

1. El programa deberá solicitar al usuario el nivel de profundidad máximo del árbol a desarrollar (ver ejemplo del programa del 8-puzzle).
2. Para cada ejecución del SBR con una de las dos estrategias de búsqueda, ANCHURA o PROFUNDIDAD, el programa debe devolver la profundidad donde la estrategia encuentra la solución y el número de nodos generados (no es necesario devolver el camino solución)
3. La representación debe ser generalista y permitir cambios fácilmente. Se debe poder trabajar con otras configuraciones de tablero de distintas dimensiones, incluir nuevos obstáculos, cajas o almacenes sin necesidad de modificar las reglas.
4. Toda la información inicial se puede representar directamente en el comando `def facts` que contiene los hechos iniciales.

INDICACIONES

1. El patrón para representar la información de un estado del problema debe contener únicamente la información dinámica susceptible de cambiar; esto es, la posición del robot, las posiciones de las cajas y el estado de los almacenes (vacío o lleno)
2. La información estática del problema (información que no cambia a lo largo de la ejecución del SBR) como la dimensión del grid o posición de los obstáculos se puede representar utilizando patrones independientes del patrón principal que representa la información de un estado del problema.
3. Para representar los obstáculos del grid se puede optar por una de las dos siguientes alternativas:
 - a. Representar explícitamente las casillas **donde hay un obstáculo**. En este caso:

- la información es estática y se puede representar con hechos independientes del hecho que representa la información del estado
 - las reglas de movimiento y empujar una caja deben comprobar QUE NO HAY UN OBSTÁCULO en la casilla destino por lo que hay que utilizar patrones negados (ver apartado NOTAS)
- b. Representar explícitamente las casillas **SIN obstáculo** (independientemente de que la casilla pueda contener una caja o un almacén). En este caso:
- la información es igualmente estática (si una casilla inicialmente no tiene obstáculo esta información no cambia a lo largo de la ejecución del problema) y se puede representar con hechos independientes del hecho que representa el estado
 - las reglas de movimiento y empujar una caja deben comprobar que la casilla destino es una casilla sin obstáculo por lo que dicha comprobación se puede realizar mediante patrones positivos

NOTAS

La sintaxis de CLIPS no permite utilizar un `test` para evaluar una variable instanciada en un ***patrón negado***.

Por ejemplo, deseamos comprobar que no existe un hecho 'item' cuyo valor sea igual al doble de algún elemento contenido en un hecho 'list'. **NO ES POSIBLE** escribir una regla como:

```
(defrule R1
  (list $? ?n1 $?)
  (not (item ?x))
  (test (= ?x (* ?n1 2)))
  ...)
```

La regla anterior debe escribirse del siguiente modo:

```
(defrule R1
  (list $? ?n1 $?)
  (not (item =(* ?n1 2)))
  ...)
```

Se puede usar una función en un patrón de la LHS de una regla si se utiliza el signo '='. Este signo indica que CLIPS evaluará la expresión que aparece a continuación en lugar de utilizarla literalmente para hacer *pattern matching*.

EVALUACIÓN DE LA PRÁCTICA

La evaluación de la práctica se realizará a través de un examen **INDIVIDUAL** que tendrá lugar el día señalado en el calendario de prácticas para cada grupo (ver transparencias

del *Bloque 0- Presentación y normativa de la asignatura*). En cada grupo se realizarán dos turnos de examen, ambos de 45 minutos.

El examen consistirá en una(s) pregunta(s) sobre el código realizado, evaluación del SBR o realización de alguna modificación del código para atender una nueva funcionalidad en el problema. Se generará una tarea en PoliformaT para la evaluación de la práctica.

Se deberá subir a la tarea de PoliformaT la versión del código realizada durante las sesiones de prácticas (versión antes del examen), así como la versión de código modificado acorde a las instrucciones del examen (**IMPORTANTE**: hay que indicar claramente el nombre de los dos componentes del grupo -si es el caso- en la versión del código de la práctica antes del examen).