UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

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 -

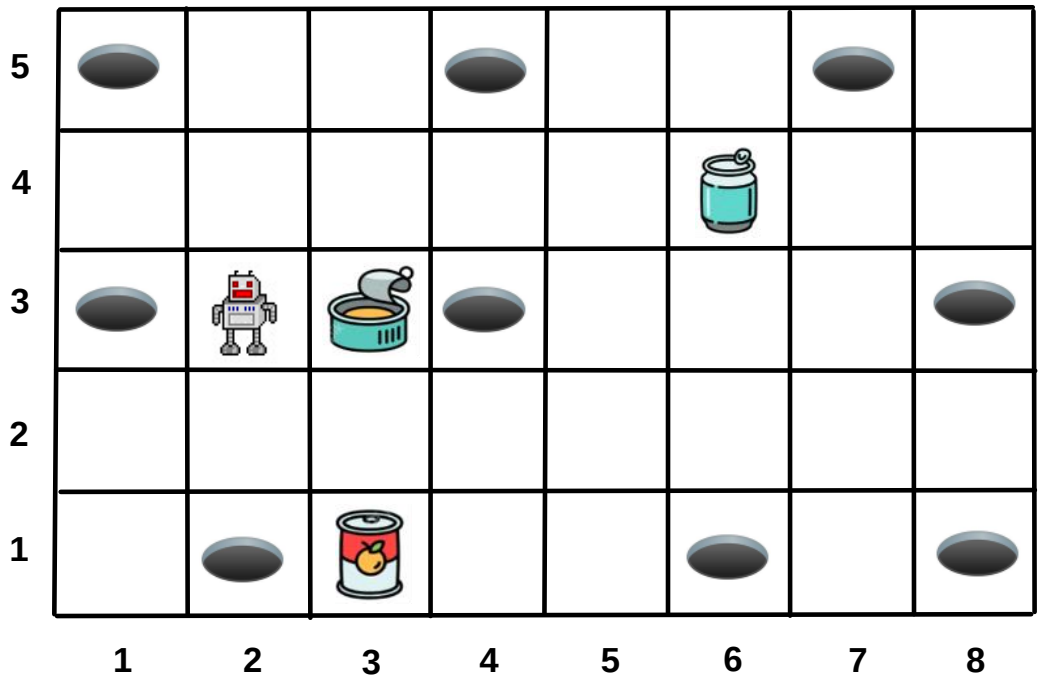
Contenedores trituradores de latas

Octubre 2021

Contenedores trituradores de latas

Sea la disposición de un grid como el que se muestra en la figura de abajo donde hay un robot, tres latas, así como varios contenedores trituradores de latas representados como un agujero. El objetivo es que el robot empuje todas las latas a los contenedores donde se triturarán para su posterior reciclaje.

Para explicar las reglas y condiciones del problema utilizaremos como ejemplo el grid de la figura aunque el programa a desarrollar ***debe ser válido para cualquier configuración de grid***, es decir, para cualquier dimensión del grid (coordenadas 'x' e 'y'), cualquier número de latas y número de contenedores trituradores. Asumiremos que siempre hay un único robot.



El robot puede moverse a una casilla adyacente en una de estas 4 direcciones: arriba, abajo, derecha e izquierda. El robot se desplaza a una celda adyacente en cualquiera de las 4 direcciones si la casilla destino no contiene un contenedor o una lata. Por ejemplo, el robot de la figura, que está en la posición (2,3), no puede moverse a la izquierda porque hay un contenedor en (1,3). Tampoco puede desplazarse a la derecha porque hay una lata en (3,3). Por otro lado, deben respetarse siempre los límites del grid en los movimientos del robot; en el ejemplo, $1 \leq x \leq 8$, $1 \leq y \leq 5$, lo que significa que no se puede desplazar a la derecha si está en una casilla de la última columna, a la izquierda si está en una casilla de la primera columna, hacia abajo si está en la fila inferior o hacia arriba si está en la última fila.

El robot puede asimismo empujar una lata en cualquiera de las cuatro direcciones: arriba, abajo, izquierda o derecha. Para empujar una lata, el robot debe situarse en una casilla adyacente a la lata y solo puede empujar la lata a una casilla que sea un contenedor triturador o que no contenga nada (no se puede empujar una lata a una

casilla que a su vez contiene una lata). En el ejemplo de la figura, las posibilidades para empujar la lata situada en la celda (3,3) son:

- El robot, que está situado en (2,3), puede empujar la lata a la derecha, en cuyo caso caería en el contenedor triturador (4,3); en este caso, tanto la lata como el robot se desplazan una posición a la derecha, terminando la lata en el contenedor y el robot en la posición (3,3).
- Si el robot se sitúa en la posición (3,4) podría empujar la lata hacia abajo; el efecto de esta operación es que tanto la lata como el robot se desplazan una fila hacia abajo en cuyo caso la lata se movería a la posición (3,2) y el robot a la posición (3,3).
- Si el robot se sitúa en la casilla (3,2) podría empujar la lata hacia arriba; el efecto de esta operación es que tanto la lata como el robot se desplazan una fila hacia arriba en cuyo caso la lata se movería a la posición (3,4) y el robot a la posición (3,3).
- El robot NO puede situarse en la casilla (4,3) para empujar la lata a la izquierda porque hay un contenedor triturador en (4,3).

Los contenedores trituradores tienen capacidad ilimitada para albergar cualquier número de latas. Una vez una lata se empuja a un contenedor, esta se tritura para su posterior reciclaje.

En resumen, los tipos de operaciones que se pueden realizar en este problema son:

- Mover un robot en cualquiera de las cuatro direcciones respetando las restricciones dadas.
- Empujar el robot una lata en cualquiera de las cuatro direcciones respetando las restricciones enunciadas. Esta operación se subdivide a su vez en dos tipos en función de los efectos de la operación, los cuales dependen del tipo de casilla a la que se desplaza la lata:
    - La lata se desplaza a una casilla donde no hay contenedor triturador.
    - La lata se desplaza a una casilla donde hay contenedor triturador; en este caso, la lata cae al contenedor y se tritura.

Se pide resolver este problema 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 PROFUNIDAD, 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 contenedores o latas sin necesidad de modificar las reglas. En todo caso, habrá solo un robot.
4. Toda la información inicial se puede representar directamente en el comando `deffacts` 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 y las posiciones de las latas en el grid.
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 contenedores se puede representar utilizando patrones independientes del patrón principal que representa la información de un estado del problema.

NOTA (en caso de utilizar patrones negados)

La sintaxis de CLIPS no permite utilizar un `test` para evaluar una variable instanciada en un ***patrón negado***. Por ejemplo, supongamos que se desea comprobar que no existe un hecho 'item' cuyo valor sea igual al doble de algún elemento contenido en un hecho 'list'. En este caso **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)))
     …
```

En la segunda regla se está usando la función 'multiplicación', indicada con el símbolo de función `*`, en un patrón de la LHS de la regla mediante el signo '='. Este signo indica que CLIPS evaluará la expresión que aparece a continuación del signo de multiplicación en lugar de realizar unificación sintáctica o *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).