Universidad catolica “Nuestra señora de la Asunción”
Facultad de ciencias y tecnología
Departamento de electrónica e informática

Implementación de un programa en SWI-Prolog que encuentre soluciones a niveles simples del juego Sokoban

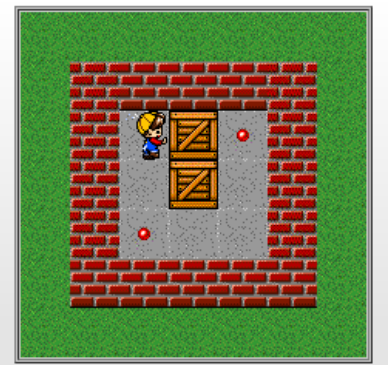
Trabajo Práctico Final - Informatica 2

Carrera: Ingeniería Informática.
Materia: Informática 2.
Profesores: Francisco Benza Bareiro y Alethia Hume.
Alumnos: Alain Vega y Mathias Martínez.

Fecha: 14/07/2023

Índice

Introducción



Sokoban es un juego de lógica en el que el jugador debe empujar cajas para colocarlas en destinos específicos en un almacén, utilizando un número limitado de movimientos.

Inventado en Japón por Hiroyuki Imabayashi, Sokoban significa "encargado de almacén" en japonés.

Parece fácil, pero los niveles van desde muy fáciles a extremadamente difíciles, y algunos lleva horas e incluso días resolverlos. La simplicidad y la elegancia de las reglas han hecho de Sokoban uno de los juegos de ingenio más populares.

La persona representa el jugador, la caja de madera representa la caja objetivo que debe ser trasladada al destino, el cual está representado con el puntito rojo, los ladrillos son obstáculos y el piso libre representa lo que significa, una casilla libre “transitable”.

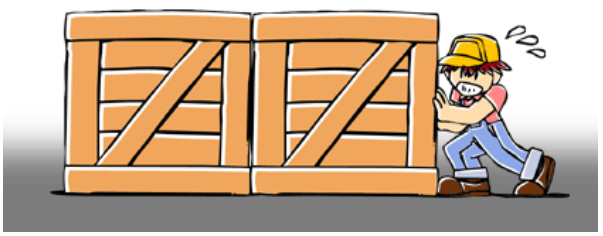
Prolog es un lenguaje de programación lógico e interpretado, se suele utilizar en el campo de la inteligencia artificial. PROLOG viene del francés: PROgrammation en LOGique, es decir, programación lógica.

SWI-Prolog es una implementación en código abierto del lenguaje de programación Prolog. El nombre SWI deriva de Sociaal-Wetenschappelijke Informatica ("Informática de Ciencias Sociales") el antiguo nombre de un grupo de investigación en la Universidad de Ámsterdam en el que Jan Wielemaker (el autor principal) está integrado.

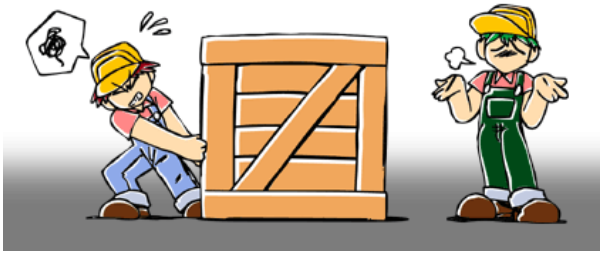
Reglas del Sokoban

1. El jugador al empujar una caja (que no esté bloqueada por obstáculos) se mueve con ella (la caja es muy pesada como para empujarla y no moverse con ella)
   Es decir, si el jugador empuja una caja, este pasa a la posición antigua de la caja y la caja se mueve exactamente una posición frente a ella (no se puede empujar la caja a más de una casilla de distancia de su posición original).

2. No se puede empujar 2 o más cajas a la vez.



3. No se puede “tirar” una caja, solo empujarla.



Implementación en Prolog

Más información se puede encontrar en la carpeta dedicada a la documentación: /docs.

Especificaciones del juego

El estado inicial del almacén estará representado por una cuadrícula bidimensional, donde cada celda puede contener un jugador, una caja, una pared, un destino o estar vacía. El jugador puede moverse en cuatro direcciones (arriba, abajo, izquierda y derecha) y solo puede empujar una caja si no hay obstáculos en el camino.

Representación del tablero

El tablero de juego se representa como una lista [] donde cada miembro de la lista representa un elemento del tablero.
Donde un elemento puede ser:

- j = jugador
- c = caja
- x = destino
- o = obstáculo.

Representación de las pistas

Una pista es la referencia de un elemento del tablero en una casilla específica. Cuya sintaxis es:

- pista(fila, columna, elemento)

Un elemento dentro del tablero puede ser:

- El jugador (j)
- Una caja cualquiera (c)
- Un destino cualquiera (x)
- Un obstáculo cualquiera (o)

Definición de estados

Estado

Un estado cualquiera *S* es una lista de donde está el jugador y todas las cajas, es decir:

$$S = [\,[j,\ [f_j, c_j],\ [c, [f_c, c_c]^+]\,]$$

El miembro $j,\ [f_j, c_j]$ representa que el jugador j está en la casilla $T_{f_j c_j}$ del tablero T

El miembro $c,\ [f_c, c_c]$ representa que el jugador j está en la casilla $T_{f_c c_c}$ del tablero T

Estado inicial

El estado inicial del juego I es una lista de donde está el jugador y todas las cajas al principio, las cuales se definen en los archivos (nivel0.pro, nivel1.pro, …, nivel5.pro) por ejemplo, el estado inicial del nivel 1 es:

$$I = [\,[j,\ [1, 2]],\ [c, [2, 2]]\,]$$

Meta

La meta M es una lista de las posiciones que deberían tener las cajas, en un estado cualquiera S por ejemplo la meta del nivel 1 es:

$$M = [\ [c, [2, 1]]\,]$$

Estado final

Los estados finales F_i (porque pueden ser más de uno) son una lista en donde está el jugador (realmente no importa donde este, por como se definió la meta) y donde todas las cajas del nivel (deberían estar en las posiciones destino, por como se definió la meta)
Es decir, un estado final F_i del nivel 1 tiene la siguiente forma.

$$F_i = [\,[j,\ [f_i, c_i]],\ [c, [2, 1]]\,]$$

Acciones posibles

Existen dos acciones principales (y se ramifican dependiendo de la dirección) las cuales son:

- $move(j, P_j, P_j')$: mover el jugador j de una posición P_j a otra P_j'.
- $push(c, P_c, P_c', P_j)$: empujar la caja c de una posición P_c a otra P_c' donde la posición del jugador es P_j

Primero definamos un breve vocabulario:

- $at(E, [F, C])$ el elemento E está en la posición $[F, C]$?
- $onTable([F, C\ ])$ la casilla $[F, C]$ pertenece al tablero?
- $free([F, C])$ la casilla $[F, C]$ está libre (no tiene un jugador o caja u obstáculo encima)?
- $up([F_1, C_1],\ [F_2, C_2])$ la casilla $[F_1, F_2]$ está arriba de la casilla $[F_2, C_2]$?
- $down([F_1, C_1],\ [F_2, C_2])$ la casilla $[F_1, F_2]$ está abajo de la casilla $[F_2, C_2]$?
- $right([F_1, C_1],\ [F_2, C_2])$ la casilla $[F_1, F_2]$ está a la derecha de la casilla $[F_2, C_2]$?
- $left([F_1, C_1],\ [F_2, C_2])$ la casilla $[F_1, F_2]$ está a la izquierda de la casilla $[F_2, C_2]$?

moveUp(j, [F_1, C_1], [F_2, C_2])

Precondición:

$at(j,\ [F_1,\ C_1]) \wedge down([F_1,\ C_1],\ [F_2,\ C_2]) \wedge\ onTable([F_2,\ C_2]) \wedge\ free([F_2,\ C_2])$

Efecto:

$at(j,\ [F_2,\ C_2]) \wedge \neg at(j,\ [F_1,\ C_1]) \wedge \neg free([F_2,\ C_2])$

moveDown(j, [F_1, C_1], [F_2, C_2])

Precondición:

$at(j,\ [F_1,\ C_1]) \wedge up([F_1,\ C_1],\ [F_2,\ C_2]) \wedge\ onTable([F_2,\ C_2]) \wedge\ free([F_2,\ C_2])$

Efecto:

$at(j,\ [F_2,\ C_2]) \wedge \neg at(j,\ [F_1,\ C_1]) \wedge \neg free([F_2,\ C_2])$

moveRight(j, [F_1, C_1], [F_2, C_2])

Precondición:

$at(j,\ [F_1,\ C_1]) \wedge left([F_1,\ C_1],\ [F_2,\ C_2]) \wedge\ onTable([F_2,\ C_2]) \wedge\ free([F_2,\ C_2])$

Efecto:

$at(j,\ [F_2,\ C_2]) \wedge \neg at(j,\ [F_1,\ C_1]) \wedge \neg free([F_2,\ C_2])$

moveLeft(j, [F_1, C_1], [F_2, C_2])

Precondición:

$at(j,\ [F_1,\ C_1]) \wedge right([F_1,\ C_1],\ [F_2,\ C_2]) \wedge\ onTable([F_2,\ C_2]) \wedge\ free([F_2,\ C_2])$

Efecto:

$at(j,\ [F_2,\ C_2]) \wedge \neg at(j,\ [F_1,\ C_1]) \wedge \neg free([F_2,\ C_2])$

pushUp(c, [F_1, C_1], [F_2, C_2], [F_j, C_j])

Precondición:

$at(c,\ [F_1, C_1]) \wedge at(j,\ [F_j, C_j]) \wedge down([F_1,\ C_1],\ [F_2, C_2]) \wedge down([F_j, C_j], [F_1, C_1]) \wedge free([F_2, C_2])$

Efecto:

$at(c,\ [F_2,\ C_2]) \wedge at(j,\ [F_1,\ C_1]) \wedge \neg free([F_2,\ C_2]) \wedge \neg at(c,\ [F_1,\ C_1]) \wedge \neg at(j,\ [F_j,\ C_j])$

pushDown(c, [F_1, C_1], [F_2, C_2], [F_j, C_j])

Precondición:

$at(c, [F_1, C_1]) \land at(j, [F_j, C_j]) \land up([F_1, C_1], [F_2, C_2]) \land up([F_j, C_j], [F_1, C_1]) \land free([F_2, C_2])$

Efecto:

$at(c, [F_2, C_2]) \land at(j, [F_1, C_1]) \land \neg free([F_2, C_2]) \land \neg at(c, [F_1, C_1]) \land \neg at(j, [F_j, C_j])$

pushRight(c, [F_1, C_1], [F_2, C_2], [F_j, C_j])

Precondición:

$at(c, [F_1, C_1]) \land at(j, [F_j, C_j]) \land left([F_1, C_1], [F_2, C_2]) \land left([F_j, C_j], [F_1, C_1]) \land free([F_2, C_2])$

Efecto:

$at(c, [F_2, C_2]) \land at(j, [F_1, C_1]) \land \neg free([F_2, C_2]) \land \neg at(c, [F_1, C_1]) \land \neg at(j, [F_j, C_j])$

pushLeft(c, [F_1, C_1], [F_2, C_2], [F_j, C_j])

Precondición:

$at(c, [F_1, C_1]) \land at(j, [F_j, C_j]) \land right([F_1, C_1], [F_2, C_2]) \land right([F_j, C_j], [F_1, C_1]) \land free([F_2, C_2])$

Efecto:

$at(c, [F_2, C_2]) \land at(j, [F_1, C_1]) \land \neg free([F_2, C_2]) \land \neg at(c, [F_1, C_1]) \land \neg at(j, [F_j, C_j])$

Representación de un nivel

Un nivel del juego es un archivo escrito en lenguaje Prolog, el cual tiene 3 partes.

1. Definición del tamaño del tablero.

   ```
   filas(cantidad_filas).
   columnas(cantidad_columnas).
   ```

   donde cantidad_filas y cantida_columnas deben ser números.

2. Definición de las pistas del nivel.

   ```
   pista(fila, columna, elemento).
   ```

   dónde fila y columna deben ser números.

3. "Inclusión" del archivo sokoban.pro el cual contiene las funciones principales para la resolución de los niveles.

   ```
   :- consult('sokoban.pro').
   ```

Predicado sokoban(L)

Es el predicado principal para la resolución del juego sokoban. Implementa la búsqueda *iterative deepening* para llegar a la meta.

Limitaciones

En niveles medianos, como el 3, 4 y 5, el algoritmo no es capaz de encontrar una solución ya que entra en un bucle infinito de acciones, por más que en el algoritmo se disponga de una lista de estados visitados.

¿Cómo probar el programa?

Clonar el repositorio, e ir a la carpeta base del proyecto:
`git clone https://github.com/AlainVega/Sokoban-Prolog.git`
`cd Sokoban-Prolog`

Instalar SWI-Prolog (en distros basadas en Ubuntu):
`sudo apt-get update`
`sudo apt-get install swi-prolog`
más información: https://www.swi-prolog.org/download/stable

Ejecutar el intérprete swi-prolog con un nivel cualquiera. Por ejemplo:
`swipl nivel1.pro`

Luego escribir lo siguiente:
`sokoban(L).`
En L estará la solución al problema del nivel1. (importante incluir el punto final.)

Niveles creados

Fueron creados un total de seis niveles, los cuales son:

- Nivel 0:
  - Tablero de 2 filas y 3 columnas.
  - El jugador, una caja y su único destino (Resuelto ya de inicio).

j		
		xc

- Nivel 1 :
    - Tablero de 2 filas y 3 columnas.
    - El jugador, una caja y su único destino.

	j	
x	c	

- Nivel 2:
    - Tablero de 3 filas y 3 columnas.
    - El jugador, dos cajas, sus dos destinos y un obstáculo.

j		o
	c	x
	c	x

- Nivel 3:
    - Tablero de 4 filas y 6 columnas.
    - El jugador, tres cajas, tres destinos y tres obstáculos.

x	j			c	o
x		o		c	
	c	o			
					x

- Nivel 4:
    - Tablero de 7 filas y 3 columnas.
    - El jugador, cuatro cajas y cuatro destinos.

xc		x
		c
	j	c
	c	
x		x

- Nivel 5:
    - Tablero de 7 filas y 10 columnas.
    - El jugador, cinco cajas, sus cinco destinos y ocho obstáculos.

o						x			
x		o					c		
				c			o		
			c	j	c		o		
				c			o		
	o			x		o	o		
x									x

Conclusión

Como consecuencias del trabajo realizado por el grupo para la realización del presente proyecto, se destacan:

- La aplicación de conceptos fundamentales de la programación lógica, como la lógica de predicados, la recursividad y el planeamiento.
- Afianzar conceptos estudiados en clase al momento de representar el problema.
- La experiencia obtenida en el paradigma de programación lógica.

Referencias

- https://www.sokoban.jp/
- https://www.swi-prolog.org/pldoc/doc_for?object=manual
- Artificial Intelligence. A Modern Approach (2da edición) Stuart Russell and Peter Norvig
- https://swish.swi-prolog.org/p/STRIPS%20Block%20World.swinb