Trabajo Práctico 2
Programación Lógica

Paradigmas ~~de Lenguajes~~ de Programación
1er cuatrimestre 2024

Fecha de entrega: 28 de junio

1. Introducción

Se quiere trabajar con un robot que debe moverse en un espacio modelado como una grilla o tablero, el que está conformado por una cantidad finita de celdas. Cada celda puede estar libre, en cuyo caso el robot puede transitarla, o puede estar ocupada y nuestro robot deberá eludirla. El robot puede moverse en forma ortogonal y de a una celda a la vez.

Se espera que se exprese en Prolog de forma elemental, declarativa y clara algunas variantes de este problema según la descripción de los ejercicios del presente trabajo.

El primer paso será armar algunos tableros para luego buscar posibles caminos dentro del mismo dadas las coordenadas iniciales y finales.

Representación utilizada

Un tablero consta de $F \times C$ celdas con F filas y C columnas. El mismo queda representado como una matriz almacenada por filas, o sea, una lista de listas donde las internas son las filas de la matriz.

Cada elemento de la matriz contendrá un átomo `ocupada`, para denotar una celda ocupada, o será una variable sin unificar, para denotar una celda libre.

Además se usarán términos de la forma `pos(F,C)` para denotar una posición determinada, con los índices comenzando en 0. Así, una lista de estos elementos denotará un camino.

2. Ejercicios

A continuación se indican los predicados solicitados para el problema.

Tablero

1. `tablero(+Filas,+Columnas,-Tablero)` instancia una estructura de tablero en blanco de `Filas` × `Columnas`, con todas las celdas libres.

```
?- tablero(3, 2, T).
T = [[_, _], [_, _], [_, _]] ;
false.
```

2. `ocupar(+Pos,?Tablero)` será verdadero cuando la posición indicada esté ocupada.

```
?- tablero(3, 2, T), ocupar(pos(1, 0), T).
T = [[_, _], [ocupada, _], [_, _]] ;
false.
```

A fin de contar con tableros de ejemplo, se sugiere tener un predicado `tablero(+nombre,-Tablero)` que instancie una estructura de tablero determinada. Por ejemplo:

```
tablero(ej5x5, T) :-
  tablero(5, 5, T),
  ocupar(pos(1, 1), T),
  ocupar(pos(1, 2), T).
```

```
tablero(libre20, T) :-
  tablero(20, 20, T).

?- tablero(ej5x5,T).
T = [[_, _, _, _, _], [_, ocupada, ocupada, _, _], [_, _, _, _, _],
[_, _, _, _, _], [_, _, _, _|...]] ;
false.
```

El tablero `ej5x5` equivale a la Fig. 1.

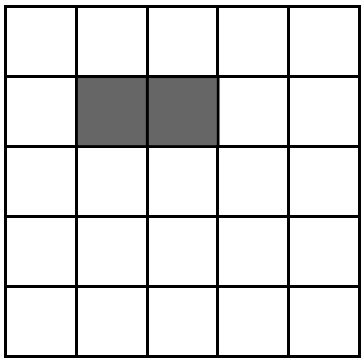

Figura 1: Tablero `ej5x5`

Vecinos

3. `vecino(+Pos, +Tablero, -PosVecino)` será verdadero cuando `PosVecino` sea una celda contigua a `Pos`. Las celdas contiguas pueden ser a lo sumo cuatro, dado que el robot se moverá en forma ortogonal.

De acá en adelante no pondremos la instanciación de la variable `T` con tablero de ejemplo `ej5x5`.

```
?- tablero(ej5x5, T), vecino(pos(0,0), T, V).
V = pos(1, 0) ;
V = pos(0, 1) ;
false.


?- tablero(ej5x5, T), vecino(pos(2,2), T, V).
V = pos(3, 2) ;
V = pos(2, 3) ;
V = pos(1, 2) ;
V = pos(2, 1) ;
false.
```

4. `vecinoLibre(+Pos, +Tablero, -PosVecino)` ídem `vecino/3`, pero además `PosVecino` debe ser una celda transitable (no ocupada) en `Tablero`.

```
?- tablero(ej5x5, T), vecinoLibre(pos(0,0), T, V).
V = pos(1, 0) ;
V = pos(0, 1) ;
false.

?- tablero(ej5x5, T), vecinoLibre(pos(2,2), T, V).
V = pos(3, 2) ;
V = pos(2, 1) ;
```

```
V = pos(2, 3) ;
false.
```

Definición de caminos

5. `camino(+Inicio, +Fin, +Tablero, -Camino)` será verdadero cuando `Camino` sea una lista `[pos(F1,C1), pos(F2,C2),..., pos(Fn,Cn)]` que denote un camino desde `Inicio` hasta `Fin` pasando sólo por celdas transitables. Además, se espera que `Camino` no contenga ciclos.
Notar que la cantidad de caminos es finita y por ende se tiene que poder recorrer todas las alternativas, eventualmente. No es relevante el orden en el que se instancian las soluciones.
Consejo: Utilizar una lista auxiliar con las posiciones visitadas.

La Fig. 2 ilustra la primera solución que podría ofrecer
`?- tablero(ej5x5, T), camino(pos(0,0), pos(2,3), T, C).`

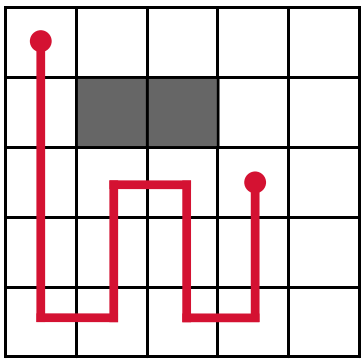

Figura 2: Ejemplo camino/4

5.1. Analizar la reversibilidad de los parámetros `Fin` y `Camino` justificando adecuadamente en cada caso por qué el predicado se comporta como lo hace.

6. `camino2(+Inicio, +Fin, +Tablero, -Camino)` ídem `camino/4`, pero se espera que las soluciones se vayan instanciando en orden creciente de longitud.

La Fig. 3 muestra dos posibles soluciones a la consulta
`?- tablero(ej5x5, T), camino2(pos(0,0), pos(2,3), T, C).` El camino rojo, al ser más corto deberá ser generado antes que el camino celeste, que también se ofrecerá como solución, eventualmente.

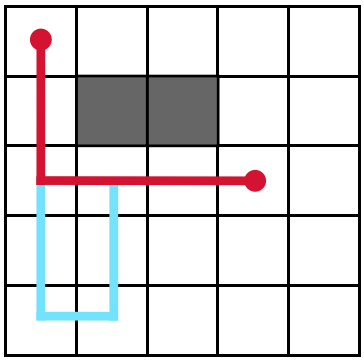

Figura 3: Ejemplo camino2/4

Como dato adicional, el total de caminos posibles es 287 según `camino2/4`. No se filtran soluciones con respecto a `camino/4`.

6.1. Analizar la reversibilidad de los parámetros `Inicio` y `Camino` justificando adecuadamente en cada caso por qué el predicado se comporta como lo hace.

7. `caminoOptimo(+Inicio, +Fin, +Tablero, -Camino)` que también instancia caminos pero sólo aquellos de distancia mínima (es decir, que no realizan pasos de más).
La Fig. 4 ilustra la primera solución que podría ofrecer
`?- tablero(ej5x5, T), camino2(pos(0,0), pos(2,3), T, C).`
A diferencia de `camino/2`, en la Fig. 3, el camino celeste no es una solución factible ya se conoció uno mejor a él (que llegó a la celda `(2,1)` en 3 pasos).

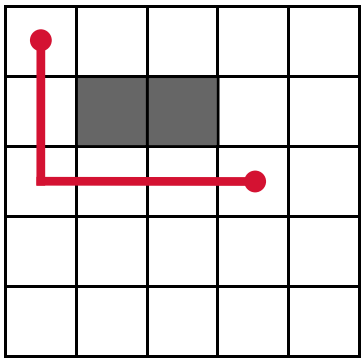

Figura 4: Ejemplo camino3/4

Notar que podría haber más de un camino óptimo. En el tablero de ejemplo existen 2 distintos.

Tableros simultáneos

8. `caminoDual(+Inicio, +Fin, +Tablero1, +Tablero2, -Camino)` será verdadero cuando `Camino` sea un camino desde `Inicio` hasta `Fin` pasando al mismo tiempo sólo por celdas transitables de ambos tableros.

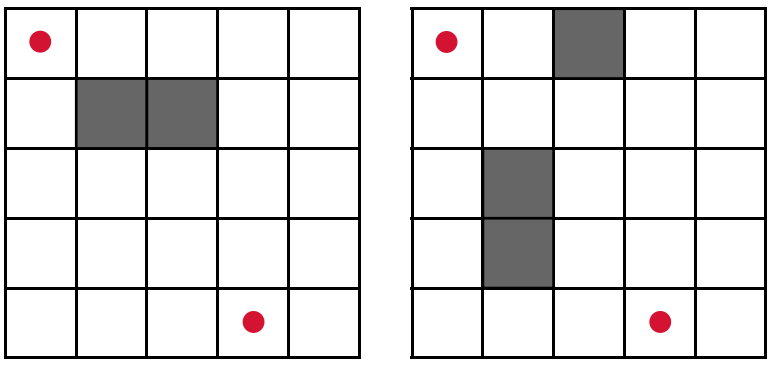

Figura 5: Dos tableros con los mismos puntos de inicio y fin.

En otras palabras, el predicado debe instanciar en Camino un camino que sea válido tanto en el primer tablero como en el segundo. Por ejemplo, de acuerdo a los tableros de la Fig. 5 y las celdas de inicio y fin marcadas, se puede ver un camino dual posible en la Fig. 6.

Observación: puede que no haya una solución posible, en tanto no hay ningún camino que conecta Inicio y fin y que sea idéntico para ambos tableros.

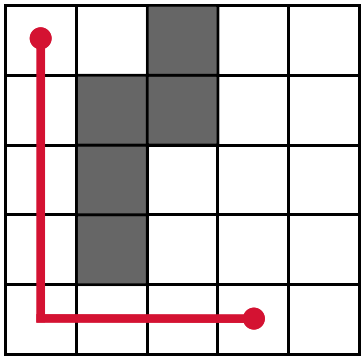

Figura 6: Una representación del camino dual para los dos tableros de la Fig. 5

3. Condiciones de aprobación

El principal objetivo de este trabajo es evaluar el correcto uso del lenguaje PROLOG de forma declarativa para resolver el problema planteado. El código debe estar *adecuadamente* comentado (es decir, los comentarios que escriban deben facilitar la lectura de los predicados, y no ser una traducción al castellano del código). También se debe explicitar cuáles de los argumentos de los predicados auxiliares deben estar instanciados usando `+`, `-` y `?`.

La entrega debe incluir casos de prueba en los que se ejecute al menos una vez cada predicado definido.

En el caso de los predicados que utilicen la técnica de *generate and test*, deberán indicarlo en los comentarios.

4. Pautas de Entrega

Se debe entregar a través del campus un único archivo llamado "tp2.pl" conteniendo el código con la implementación de los predicados pedidos. Para eso, ya se encuentra disponible la entrega "TP2 - Programación Lógica" en la solapa "TPs" (configurada de forma grupal para que sólo una persona haga la entrega en nombre del grupo). Cada predicado asociado a los ejercicios debe contar con ejemplos que muestren que exhibe la funcionalidad solicitada. El código debe poder ser ejecutado en SWI-Prolog. No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté adecuadamente comentado. Los objetivos a evaluar en la implementación de los predicados son los siguientes:

- Corrección.
- Declaratividad.
- Reutilización de predicados previamente definidos.
- Uso adecuado de unificación, backtracking, generate and test y reversibilidad.
- Salvo donde se indique lo contrario, los predicados no deben instanciar soluciones repetidas. Vale aclarar que no es necesario filtrar las soluciones repetidas si la repetición proviene de las características de la entrada.

5. Referencias y sugerencias

Como referencia se recomienda la bibliografía incluída en el sitio de la materia (ver sección *Enlaces y Bibliografía* → *Programación Lógica*).

Se recomienda que, siempre que sea posible, utilicen los predicados ISO y los de SWI-Prolog ya disponibles. Recomendamos especialmente examinar los predicados y metapredicados que figuran en la sección *útil* de la página de la materia. Pueden hallar la descripción de los mismos en la ayuda de `SWI-Prolog` (a la que acceden con el predicado `help`). También se puede acceder a la documentación online de SWI-Prolog.