Lógica para Cs. de la Computación

Fabrega, Juan Ignacio
Schwerdt, Matías David
Comision 16

Proyecto 2: El Trizzle

Primer Cuatrimestre 2020

Requerimientos de Implementación

Implementar en Prolog un predicado desplazar(+Dir, +Num, +Cant, +Tablero, -EvolTablero)

donde:

*Dir, parámetro de entrada, especifica la dirección del desplazamiento, y puede ser izq, der (desplazamiento de una fila), arriba o abajo (desplazamiento de una columna).

*Num, parámetro de entrada, especifica el número de fila (si Dir es izq o der) o de columna (si Dir es arriba o abajo) a desplazar, y es un numero del 1 al 5.

*Cant, parámetro de entrada, especifica la cantidad de lugares a desplazar, y es un numero del 1 al 4 (ya que desplazar 5 lugares nos dejar´ıa en el mismo tablero).

*Tablero, parámetro de entrada, especifica el tablero sobre el cual se aplicará el desplazamiento.

*EvolTablero, parámetro de salida, es una lista de tableros que representa la evolución del tablero original como consecuencia de la movida realizada.

desplazar/4

La representación del tablero en formato de lista de listas lo convertimos en 25 hechos celda/4. Luego se realiza el movimiento, se buscan todos los colapsos y se marca cada celda/4 en su cuarto parámetro. Se acomodan las mamushkas en base a la marca realizada. Finalmente se aplica la gravedad y se genera las mamushkas al azar en donde hubo colapsos. En caso de que no haya habido colapsos se retorna solamente 1 tablero con el movimiento realizado. En caso contrario, se retornan los 4 tableros: con el movimiento, con los colapsos, con la gravedad aplicada y con las mamushkas al azar.

celda/4

Se generan 25 hechos basándose en las 25 celdas del Tablero. A lo largo del código se trabaja siempre con los hechos, dado que es más fácil que operar directamente con la lista de listas del Tablero.

En éste hecho las filas y columnas van desde 0 a 4 inclusive. El tercer parámetro es la mamushka, y el cuarto es para saber si la mamushka debe recibir un cambio o no dependiendo de si se produjo un colapso

guardarTablero/1

Recibe una lista de listas, las cuales representan las filas del tablero. Para cada una de ellas llamamos al predicado guardarFila/2.

guardarFila/2

Recibe el número de fila y una lista de celdas. Para cada una de ellas creamos el predicado celda/3 utilizando assert/1. De ésta manera logramos que cada celda del tablero esté escrito como un hecho.

mover/3

Recibe tres parámetros. El primero representa hacia donde quiere realizarse el movimiento(izquierda,derecha,arriba y abajo), el segundo nos indica el número de la columna o fila en la que de se desea realizar el movimiento, y el tercer parámetro nos indica la cantidad de espacios se desea mover esa fila/columna.

moverFila/2, moverColumna/2

Para cada celda de la fila/columna, calcula su nueva ubicación, borra la antigua celda y crea la nueva celda con la ubicación nueva.

hechosATablero/1

A partir de los hechos de celda/4, retorna el tablero en forma de lista de filas.

filaALista/2

Este predicado tiene dos parámetros, el primero recibe un número que representa la fila de la cual se quiere obtener las mamushkas. El segundo parámetro es una lista con las muñecas de la esa fila.

insertar_ultimo/3

Este predicado se encargue de recibir un elemento y una lista, e insertar este elemento como el último elemento de la lista pasada por parámetro. Luego de hacer lo anterior se devuelve la lista en el tercer parámetro.

buscarColpasosDeFilas/2

Este predicado tiene dos parámetros, el primero es una lista y el segundo es la misma lista con todos los colapsos de todos las filas adentro. En otras palabras este predicado se encarga de retornar todos los colapsos de todas las filas en una lista.

buscarColapsoFila/3

Este predicado tiene tres parámetros, el primero indica el número de fila que se desea analizar, el segundo representa una lista, y el último esa misma lista con los colapso que se encuentren. Es decir, dado un cierto número de fila, este predicado devuelve, si que existen colapsos de: primero de 5 mamushkas, si no existe tal colapso intenta buscar un colapso de 4 muñecas, si no es posible intenta con 3 mamushkas, y en caso de tampoco encontrar colapso devuelve la misma lista pasada por parámetros ya que no es posible agregarle algún tipo colapso.

buscarColapsoFila/4

Este predicado tiene 4 parámetros, el primero indica el número de la fila en la que se desea buscar el colapso, el segundo la cantidad de colapsos que se quieren encontrar(puede valer 5, 4 o 3), el tercero es una lista y el cuarto es esa misma lista más los colapsos encontrados. Es decir dado un cierto número de fila busca y si encuentra el colapso de 5/4/3 en la misma fila lo agrega a la lista y la devuelve.

buscarColapsosDeColumnas/2

Este predicado tiene dos parámetros, el primero es una lista y el segundo es la misma lista con todos los colapsos de todos las columnas adentro. En otras palabras este predicado se encarga de retornar todos los colapsos de todas las columnas en una lista.

buscarColapsoColumna/3

Este predicado tiene tres parámetros, el primero indica el número de columna que se desea analizar, el segundo representa una lista, y el último esa misma lista con los colapso que se

encuentren. Es decir, dado un cierto número de columna, este predicado devuelve, si que existen colapsos de: primero de 5 mamushkas, si no existe tal colapso intenta buscar un colapso de 4 muñecas, si no es posible intenta con 3 mamushkas, y en caso de tampoco encontrar colapso devuelve la misma lista pasada por parámetros ya que no es posible agregarle algún tipo colapso.

buscarColapsoColumna/4

Este predicado tiene 4 parámetros, el primero indica el número de la columna en la que se desea buscar el colapso, el segundo la cantidad de colapsos que se quieren encontrar(puede valer 5, 4 o 3), el tercero es una lista y el cuarto es esa misma lista más los colapsos encontrados. Es decir dado un cierto número de columna busca y si encuentra el colapso de 5/4/3 dentro de esa misma columna lo agrega a la lista y la devuelve.

buscarTodosLosColapsos/1

Este predicado devuelve en forma de lista todos los colapsos que hay en el tablero. Devuelve una lista, donde cada elemento de esa lista es un colapso. Cada colapso se representa como una lista de pares y un colapso se representa de la siguiente manera [[0,0], [0,1], [0,2]].

obtenerCentro/2

Dado una lista que solo puede tener 3,4 o 5 elementos devuelve el “centro” de la misma. La lista representa un colapso de 3,4 o 5 muñecas respectivamente. Por ende este predicado retorna el centro del colapso, la mamuschka a la cual hay que agrandar.

siguienteEstado/2

Este predicado tiene dos parámetros, el primero representa el estado de cierta mamuschka y retorna el estado al cual tiene que cambiar la muñeca. Los estados en este caso son: sin cambios, borrar y agrandar.

obtenerFila/2

Este predicado recibe una lista de dos elementos, que representan las coordenas, y devuelve primer elemento de esta lista. De esta manera obtenemos el valor de la fila.

obtenerColumna/2

Este predicado recibe una lista de dos elementos, que representan las coordenas, y devuelve el segundo elemento de esta lista. De esta manera obtenemos el valor de la columna.

marcarColapsosYcentrosComunes/1

Este predicado recibe una lista, que adentro tiene las coordenadas de todas las celdas que han colapsado. Se recorre la lista para obtener las coordenadas de las celdas para poder marcar cada celda utilizando el predicado siguienteEstado. De esta manera se logra marcar las celdas que tienen que borrarse y en caso de que una celda se encuentre en dos o más colapsos se la marca para que esta celda tiene que agrandarse. Se puede observa que no marcamos los centros de aquellos colapsos que no “choquen” con otro colapso, solo los marcamos que tiene que ser borrados. El centro de estos colapsos será resuelto con marcarCentrosFinal/1.

tieneCentro/1

Este predicado recibe una lista de 5,4 o 3 elementos que representa un colapso de 5,4 o 3 muñecas respectivamente. En caso de que el estado de todas las muñecas del colapso sea borrar, esto quiere decir que todavia no se sabe cual es la mamushka a la cual hay que agrandarle el tamaño, se lllama al predicado ponerCentro/1 para realice lo anteriormente mencionado. En caso de que algunas de las muñecas no tenga el estado borrar significa que en ese colapso ya se sabe cual es la muñeca a la que hay que agrandar.

ponerCentro/1

Este predicado recibe una lista de 5,4 o 3 elementos que representa un colapso de 5,4 o 3 muñecas respectivamente, obtiene las coordenadas para saber cual es la celda que debería agrandarse y cambia el estado de esa mamushka por agrandar.

marcarCentrosFinal/1

Recibe una lista que representa todos los colapsos que hay en el tablero, y en aquellos colapsos que no se sepa cual es la mamushka que hay que agrandarle el tamaño, la busca y le setea el estado como agrandar.

marcar/1

Recibe una lista que representa todos los colapsos que hay en el tablero, la recorre y setea el estado de las celdas como borrar o agrandar, según corresponda.

aplicarEstados/0

Recorre todos los hechos de celda/4 revisando el cuarto parámetro de ellas. Si éste cuarto parámetro es "agrandar" entonces se agranda el valor de la mamushka (tercer parámetro). Si es "borrar" se reemplaza la mamushka por una "x". Si es "sincambios", no se hace ningún cambio.

actualizarMamushka/3

El primer parámetro es una mamushka, el segundo parámetro es la acción a aplicar. En caso de ser “agrandar” se retorna una mamushka del tamaño siguiente. En caso de ser “borrar” se retorna una “x”.

gravedad/0

Este predicado recorre todas las columnas y se encarga de “mover” los espacios vacíos hacia arriba, a la parte superior del tablero.

recorrerColumna/1

Este predicado recibe por parámetro un número de columna y trabaja en ella. Se encarga de “mover” los espacios vacíos hacia arriba, a la parte superior del tablero de una columna determinada.

listaOrdenada/2

Este predicado recibe una columna en forma de lista, y devuelve una lista de mamushkas ordenada de abajo hacia arriba.

rellenarColumna/2

Este predicado recibe una columna en forma de lista, la cual ya se encuentra ordenada y lo único que hace es completar la lista con ‘x’ hasta que esta lista tenga 5 elementos.

pasarListaGravedadAHechos/2

Recibe un número de columna y una lista de mamushkas pertenecientes a esa columna. Dicha lista ya tiene la gravedad aplicada. Lo que hace éste predicado es actualizar los hechos celda/4 en base a esa lista.

generarMamushkasRandom/0

Para cada mamushka que vale “x”, se cambia por una mamushka al azar.

borrarRepetidos/2

En caso de que no haya colapsos, la lista solamente tendrá el Tablero1, caso contrario tendrá los 4 tableros.

mostrarTableros/4, verTodosLosTableros/2,verPrimerTablero/2

Se recibe una lista de colapsos, y se retorna cada uno de ellos en un parámetro distinto, con el objetivo de poder visualizar la evolución del tablero.

Tests

Mostraremos una serie de tableros, consultas y la evolución del tablero.

1) **Tablero** =

a1	r2	v1	a1	a1
v2	r1	r1	a2	a1
v2	v1	a2	v1	v1
a2	a2	a1	v3	a1
r1	v1	r1	a1	r1

```
Tablero = [[a1,r2,v1,a1,a1],
           [v2,r1,r1,a2,a1],
           [v2,v1,a2,v1,v1],
           [a2,a2,a1,v3,a1],
           [r1,v1,r1,a1,r1]],
desplazar(abajo, 3, 1, Tablero, EvolTablero),!,
mostrarTableros(EvolTablero,Tablero1,Tablero2,Tablero3,Tablero4).
```

Tablero1 =

a1	r2	r1	a1	a1
v2	r1	v1	a2	a1
v2	v1	r1	v1	v1
a2	a2	a2	v3	a1
r1	v1	a1	a1	r1

Tablero3 =

x	r2	x	a1	a1
a1	r1	r1	a2	a1
v2	v1	v1	v1	v1
v2	a3	r1	v3	a1
r1	v1	a1	a1	r1

Tablero2 =

a1	r2	r1	a1	a1
v2	r1	v1	a2	a1
v2	v1	r1	v1	v1
x	a3	x	v3	a1
r1	v1	a1	a1	r1

Tablero4 =

v1	r2	r1	a1	a1
a1	r1	r1	a2	a1
v2	v1	v1	v1	v1
v2	a3	r1	v3	a1
r1	v1	a1	a1	r1

2)

Tablero =

r2	v1	a1	v1	r1
r2	a2	r2	a1	v2
v1	a2	v2	v2	r1
a2	v2	r1	a2	a2
v2	v1	a3	v1	r2

```
Tablero = [[r2,v1,a1,v1,r1],
           [r2,a2,r2,a1,v2],
           [v1,a2,v2,v2,r1],
           [a2,v2,r1,a2,a2],
           [v2,v1,a3,v1,r2]],
desplazar(der, 4, 2, Tablero, EvolTablero),!,
mostrarTableros(EvolTablero,Tablero1,Tablero2,Tablero3,Tablero4).
```

Tablero1 =

r2	v1	a1	v1	r1
r2	a2	r2	a1	v2
v1	a2	v2	v2	r1
a2	a2	a2	v2	r1
v2	v1	a3	v1	r2

Tablero2 =

r2	v1	a1	v1	r1
r2	x	r2	a1	v2
v1	x	v2	v2	r1
x	a3	x	v2	r1
v2	v1	a3	v1	r2

Tablero3 =

x	x	x	v1	r1
r2	x	a1	a1	v2
r2	v1	r2	v2	r1
v1	a3	v2	v2	r1
v2	v1	a3	v1	r2

Tablero4 =

a1	a1	a1	v1	r1
r2	a1	a1	a1	v2
r2	v1	r2	v2	r1
v1	a3	v2	v2	r1
v2	v1	a3	v1	r2

3)

Tablero =

v2	v1	v1	r1	v1
a1	a1	v1	a1	a1
v2	r2	r2	v2	r1
r2	v1	v2	v3	r1
v1	a3	a3	v1	r2

```
Tablero = [[v2,v1,v1,r1,v1],
           [a1,a1,v1,a1,a1],
           [v2,r2,r2,v2,r1],
           [r2,v1,v2,v3,r1],
           [v1,a3,a3,v1,r2]],
desplazar(der, 2, 2, Tablero, EvolTablero),!,
mostrarTableros(EvolTablero,Tablero1,Tablero2,Tablero3,Tablero4).
```

Tablero1 =

v2	v1	v1	r1	v1
a1	a1	a1	a1	v1
v2	r2	r2	v2	r1
r2	v1	v2	v3	r1
v1	a3	a3	v1	r2

Tablero3 =

x	v1	x	x	v1
v2	a2	v1	r1	v1
v2	r2	r2	v2	r1
r2	v1	v2	v3	r1
v1	a3	a3	v1	r2

Tablero2 =

v2	v1	v1	r1	v1
x	a2	x	x	v1
v2	r2	r2	v2	r1
r2	v1	v2	v3	r1
v1	a3	a3	v1	r2

Tablero4 =

r1	v1	a1	v1	v1
v2	a2	v1	r1	v1
v2	r2	r2	v2	r1
r2	v1	v2	v3	r1
v1	a3	a3	v1	r2

4)

Tablero =

v1	a3	r1	r3	r2
a3	r3	v1	v2	v3
a3	r3	a1	a1	a1
a3	r2	v1	r2	v1
a3	v1	r2	v3	r1

```
Tablero = [[v1,a3,r1,r3,r2],
           [a3,r3,v1,v2,v3],
           [a3,r3,a1,a1,a1],
           [a3,r2,v1,r2,v1],
           [a3,v1,r2,v3,r1]],
desplazar(izq, 1, 1, Tablero, EvolTablero),!,
mostrarTableros(EvolTablero,Tablero1,Tablero2,Tablero3,Tablero4).
```

Tablero1 =

a3	r1	r3	r2	v1
a3	r3	v1	v2	v3
a3	r3	a1	a1	a1
a3	r2	v1	r2	v1
a3	v1	r2	v3	r1

Tablero3 =

x	r1	x	r2	x
x	r3	r3	v2	v1
x	r3	v1	a2	v3
x	r2	v1	r2	v1
a3	v1	r2	v3	r1

Tablero2 =

x	r1	r3	r2	v1
x	r3	v1	v2	v3
a3	r3	x	a2	x
x	r2	v1	r2	v1
x	v1	r2	v3	r1

Tablero4 =

r1	r1	r1	r2	r1
v1	r3	r3	v2	v1
r1	r3	v1	a2	v3
a1	r2	v1	r2	v1
a3	v1	r2	v3	r1

5)

Tablero =

a1	v2	v3	a2	a2
v1	r3	r1	r1	a2
a1	v2	v1	a1	a1
v2	r2	v1	r1	r1
r1	a1	r2	v2	a1

,

```
Tablero = [[a1,v2,v3,a2,a2],
           [v1,r3,r1,r1,a2],
           [a1,v2,v1,a1,a1],
           [v2,r2,v1,r1,r1],
           [r1,a1,r2,v2,a1]],
desplazar(arriba, 5, 4, Tablero, EvolTablero),!,
mostrarTableros(EvolTablero,Tablero1,Tablero2,Tablero3,Tablero4).
```

Tablero1 =

a1	v2	v3	a2	a1
v1	r3	r1	r1	a2
a1	v2	v1	a1	a2
v2	r2	v1	r1	a1
r1	a1	r2	v2	r1