Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Lógica para Ciencias de la Computación

Proyecto N^{o} 2

Trizzle

Primer Cuatrimestre de 2020

El juego

El Trizzle es un juego que consta de una grilla de 5x5, llena con mamushkas (muñecas) de 3 tamaños diferentes (chicas, medianas, y grandes) y 3 colores diferentes (rojas, verdes y azules).

En cada momento del juego se puede desplazar una fila (hacia la derecha o izquierda) o columna (hacia arriba o abajo), y si 3 o más mamushkas exactamente iguales quedan alineadas y consecutivas, entonces colapsan generando una única mamushka del tamaño siguiente. Si las mamushkas alineadas consecutivamente fueran grandes, entonces simplemente colapsan en una muñeca grande. En cada movida más de un grupo de mamushkas podrían quedar alineadas y consecutivas, con lo que podría haber más de un colapso en simultáneo.

La mejor manera de entender las reglas del juego es jugándolo, y esta es una de las versiones disponibles en línea.

Simplificaciones respecto al juego original

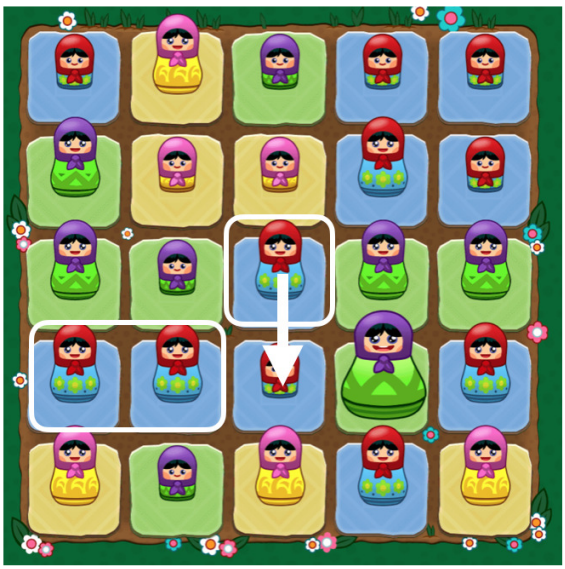
Para el presente proyecto omitiremos las siguientes dos características del juego a modo de simplificación:

- en el juego original alinear 3 mamushkas grandes se conoce como *megamatch*, y provoca además que todas las mamushkas que rodean a la resultante del colapso incrementen su tamaño al siguiente.
- en el juego original un colapso podría generar otros *colapsos en cadena*, cuando el tablero resultante luego de agregar las nuevas muñecas para rellenar los espacios vacíos presenta nuevos grupos de muñecas iguales alineadas y contiguas, que generarían un nuevo colapso. Este proceso puede repetirse varias veces hasta que el tablero vuelve a quedar estable.

Ubicación de la mamushka resultante del colapso

La mamushka resultante de colapsar un grupo de muñecas idénticas, alineadas y consecutivas ocupará el lugar que deje alguna de las mamushkas de dicho grupo. Para determinar cuál de los lugares del grupo ocupará se tendrá en cuenta la movida que originó el colapso, según las siguientes reglas:

- Quedará en aquella posición que forma parte de la fila o columna que se desplazó para causar el colapso.



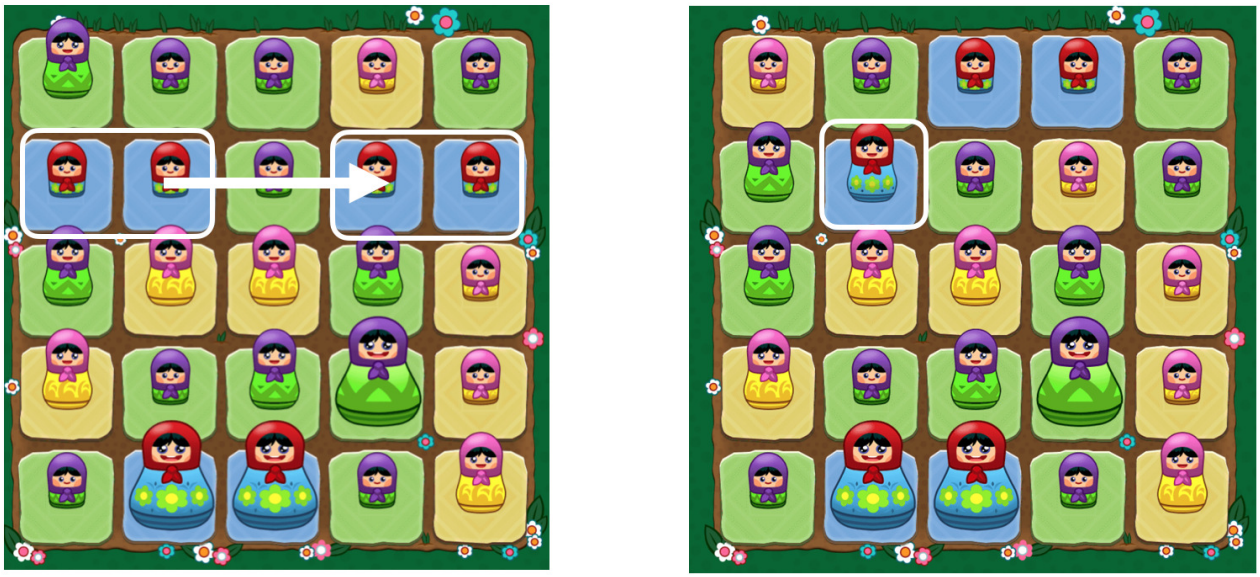
Desplazamiento de columna 3 hacia abajo 1 posición.

- Si hay más de una posición que forma parte de la fila o columna desplazada, entonces puede tratarse de un colapso *cruzado*, con lo que debe elegirse la de la intersección, es decir, aquella tal que toda otra posición del grupo de posiciones colapsadas comparte la fila o columna de dicha posición.

Desplazamiento de fila 4 hacia la derecha 2 posiciones.

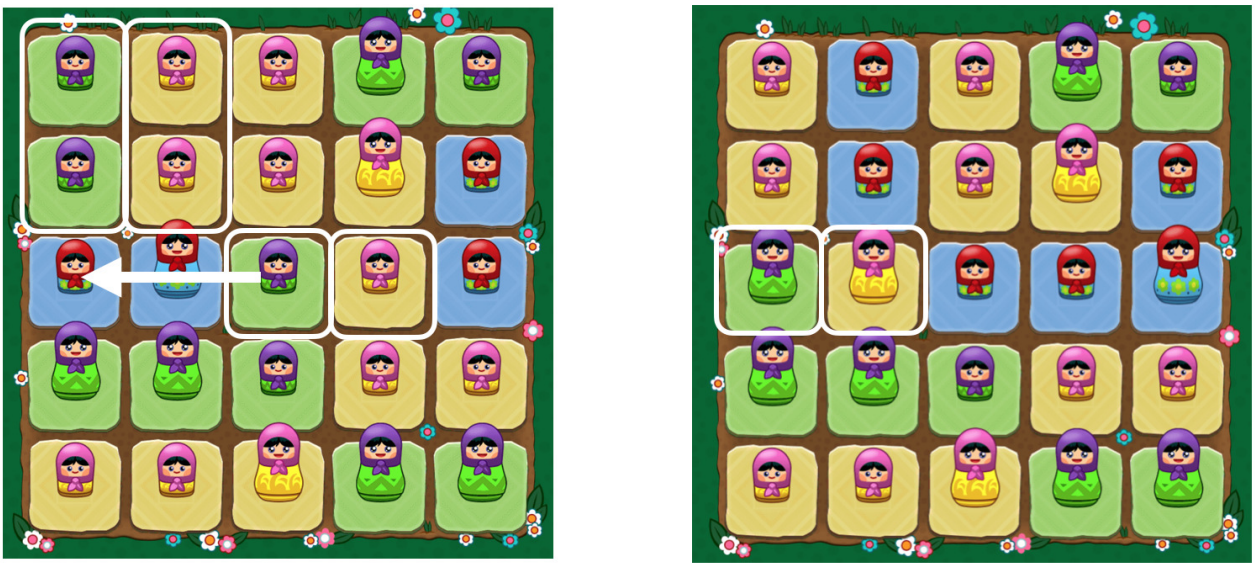
- Pero si todas las posiciones forman parte de la fila o columna desplazada, entonces se trata de un colapso por juntar muñecas que estaban en los extremos de una misma fila o columna, y por lo tanto desconectadas, que se conectan al desplazar la fila o columna,

en cuyo caso la posición elegida será la central, en caso de ser 3 muñecas, o la segunda posición (de izquierda a derecha para una fila, y de arriba hacia abajo para una columna) si son 4 muñecas.



Desplazamiento de fila 2 hacia la derecha 2 posiciones.

Estas reglas aplican para elegir la ubicación de la mamushka resultante de colapsar un grupo de mamushkas iguales, alineadas y contiguas, pero tener presente que como consecuencia de un desplazamiento puede originarse más de un grupo de mamushkas iguales, alineadas y contiguas. Por ejemplo:



Desplazamiento de fila 3 hacia la izquierda 2 posiciones.

Luego las reglas anteriores deben aplicarse para cada uno de los grupos, en este caso, para ambos grupos se aplica la primer regla.

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 número del 1 al 5.
- Cant, parámetro de entrada, especifica la cantidad de lugares a desplazar, y es un número del 1 al 4 (ya que desplazar 5 lugares nos dejaría en el mismo tablero).
- Tablero, parámetro de entrada, especifica el tablero sobre el cual se aplicará el desplazamiento.
- EvoTablero, parámetro de salida, es una lista de tableros que representa la evolución del tablero original como consecuencia de la movida realizada; dependiendo del desplazamiento realizado EvolTablero incluirá algunos de (posiblemente todos) los siguientes tableros:
  1. Tablero1, resultante de realizar el desplazamiento (Dir, Num, Cant) sobre Tablero,
  2. Tablero2, resultante de colapsar todos los grupos de mamushkas alineadas y contiguas en la mamushka de tamaño siguiente a partir de Tablero1; no se deben rellenar los espacios vacíos que se generaron, sino que deben ser reemplazados por la constante x,
  3. Tablero3, resultante de desplazar hacia abajo, por "gravedad", las muñecas por encima de los espacios vacíos en Tablero2, lo que "moverá' los espacios vacíos hacia arriba, a la parte superior del tablero,
  4. Tablero4, resultante de rellenar los espacios vacíos en Tablero3 con muñecas de tamaño chico y colores elegidos aleatoriamente.

Notar que si el desplazamiento no produce un colapso, entonces EvolTablero = [Tablero1].

Representación del tablero

Tanto el tablero de entrada, Tablero, como los tableros en la lista EvolTablero, deben representarse mediante una lista de filas, donde una fila es una lista de las siguientes constantes: r1, r2, r3 (mamushkas rojas de tamaño chico, mediano y grande, respectivamente), v1, v2, v3 (mamushkas verdes de tamaño chico, mediano y grande, respectivamente), a1, a2, a3 (mamushkas azules de tamaño chico, mediano y grande, respectivamente).

La razón de adoptar esta representación, como lista de listas, es que el SWI-Prolog cuenta con una librería para graficar listas de listas automáticamente como tablas. Para incluir la librería agregar la siguiente línea al principio del archivo .pl:

```
:- use_rendering(table).
```

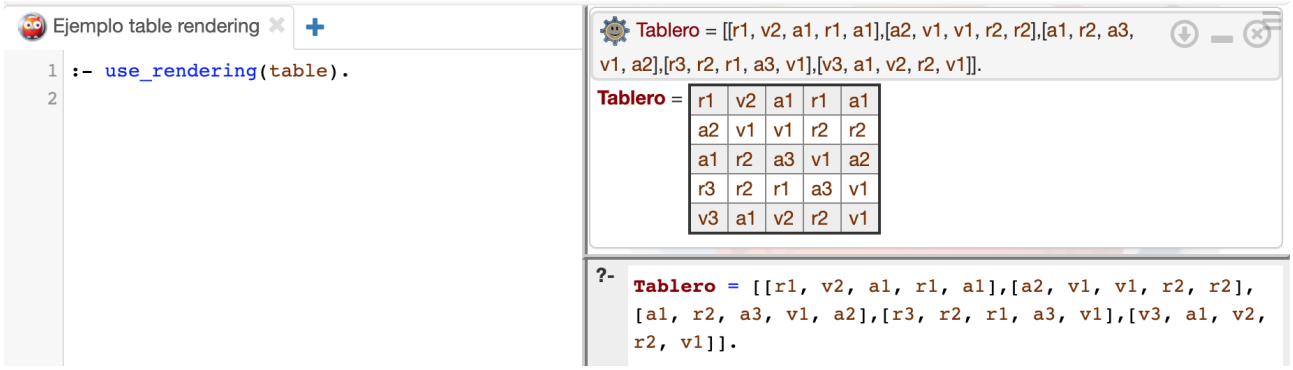
Luego todas las respuestas brindadas por PROLOG que sean listas de listas (donde esas listas tengan todas el mismo tamaño) serán graficadas como tablas. Por ejemplo, frente a la consulta PROLOG:

?- Tablero = [[r1, v2, a1, r1, a1],[a2, v1, v1, r2, r2],[a1, r2, a3, v1, a2],[r3, r2, r1, a3, v1],[v3, a1, v2, r2, v1]].

cuya respuesta involucra una lista de listas:

Tablero = [[r1, v2, a1, r1, a1],[a2, v1, v1, r2, r2],[a1, r2, a3, v1, a2],[r3, r2, r1, a3, v1],[v3, a1, v2, r2, v1]].,

el SWI-Prolog la graficará como una tabla, como se muestra en la siguiente captura de pantalla de SWISH.



La siguiente consulta puede resultar útil para graficar el tablero de entrada y uno a uno los tableros en EvolTablero mediante el pedido de soluciones alternativas:

```
?- _Tablero = [...],
   desplazar(abajo, 3, 1, _Tablero, _EvolTablero),!,
   member(TableroX, [_Tablero |_EvolTablero]).
```

Notar que todas las variables empleadas son anónimas, a excepción de TableroX que es la que queremos que se muestre como tabla en cada solución alternativa.

IMPORTANTE: **internamente, en la implementación, puede adoptar cualquier representación para el tablero**, mientras la interfaz del predicado respete la representación requerida anteriormente. Incluso puede resultar cómodo transformar el tablero de entrada, como lista de listas, a una representación como hechos PROLOG, por ejemplo, celda(Fila, Col, Mamushka), operar sobre dicho tablero mediante assert/1 y retract/1, e ir transformando cada tablero intermedio a la representación como lista de listas para construir la lista de tableros a retornar EvolTablero.

Requerimientos de Documentación

Se deberá realizar un informe que **explique** claramente la **implementación** en PROLOG realizada. Puede aprovechar el informe para destacar características positivas de la resolución, y documentar cualquier otra observación que considere pertinente. Además deberán incluir **casos de tests**, con sus respectivas respuestas, para el predicado que se requiere implementar.

Se recomienda estructurar el informe de manera top-down, comenzando con una descripción a alto nivel de la implementación.

Importante: en el desarrollo de software, la documentación de la implementación constituye un elemento fundamental. Es por esto que, para la evaluación del presente proyecto, se dará suma importancia a la calidad (claridad y completitud) del informe entregado.

Condiciones de Entrega

1. Las comisiones pueden estar conformadas por hasta 2 integrantes, y deben ser registradas en la página de la materia. A cada comisión se le asignará un integrante de la cátedra, quien hará el seguimiento y corregirá el proyecto de la comisión.

2. La fecha límite de entrega del presente proyecto se encuentra publicada en la página de la materia. Los proyectos entregados fuera de término recibirán una penalización en su calificación, la cual será proporcional al retraso incurrido.

3. La entrega del proyecto consiste del envío por mail de la resolución del proyecto y versión electrónica del informe.

   - Enviar por mail directamente al integrante de cátedra asignado a la comisión, con copia al asistente (en caso de no ser el asignado). Mails:
     - Federico: `fedeschmidt.10+LCC@gmail.com`
     - Diego: `dieorbe96+LCC@gmail.com`
     - Germán: `germang04+LCC@gmail.com`
     - Mauro (asistente): `mgomezlucero+LCC@gmail.com`
   - Asunto del mail: "Proyecto LCC - Comisión <`Ap.y Nom. Integrantes`>"
   - Adjunto: un .zip conteniendo 1) un archivo PR-1.pl con la implementación Prolog y 2) un pdf con la versión electrónica del informe.