DOCUMENTACIÓN DEL PROYECTO "NONOGRAMA"

"IMPLEMENTACIÓN EN PROLOG"

Alumnos

Latouquette, David Emanuel - López, David Estebán

Comisión N° 32

ÍNDICE

Introducción

En esta documentación se hablará sobre la implementación en Prolog del proyecto “Nonograma”. Dicha implementación será la parte lógica del mismo programa.

El enfoque de este documento se realizará sobre el archivo ***proylcc.pl*** ubicado en la ruta ***pengines_server\apps\proylcc***. Dicho archivo será el que contenga la mayoría de las operaciones que hacen posible el funcionamiento lógico del proyecto.

Para finalizar la introducción, en esta documentación se hará mayor énfasis en la implementación lógica del proyecto, en donde será explicado cuál es la función de cada operación implementada en el proyecto.

1. Observaciones y explicaciones de estrategias utilizadas

I. Servicio de Prolog

Para el correcto funcionamiento del proyecto, debe ser ejecutado el servicio de Prolog con el archivo ***run.pl*** que se encuentra en la ruta ***LCC-proyecto-nonograma\pengines_server***.

La ejecución de dicho archivo es la que facilita el acceso a las operaciones (predicados) al servidor de ***React***, el cual consulta y actualiza el tablero (la grilla) mediante las operaciones que dispone el proyecto dentro de la carpeta ***LCC-proyecto-nonograma\pengines_server\apps\proylcc***.

Una particularidad de este archivo, es que se mantendrá ejecutándose sin interacción del usuario (en ningún momento se requiere escribir algo sobre la consola de Prolog).

II. ¿Que representa la implementación en Prolog?

Los archivos que están ubicados dentro de la carpeta ***LCC-proyecto-nonograma\pengines_server\apps\proylcc*** representan la lógica del proyecto, es decir, donde se realiza la actualización del estado del juego.

III. Estrategias utilizadas

La implementación de los predicados del archivo ***proylcc.pl*** fueron implementados empleando como estrategia principal el uso de listas que son recorridas empleando recursividad.

Para cada uno de los predicados detallados en la sección "2. Predicados", se debe recorrer de dicha manera las listas.

2. Predicados

Aclaraciones

1). Cuando se menciona “encabezado actual” se hace referencia al primer elemento que se encuentra luego de realizar una llamada recursiva con una lista “más pequeña”.

2). En los predicados que tienen como parámetro de salida a ***FilaSat*** y ***ColSat***, estos dos retornan el grado de satisfacción de la fila y la columna en las que es insertado/modificado el elemento, devolviendo 1 si se satisfacen las pistas y 0 en caso contrario.

I. Predicados del archivo init.pl

El archivo ***init.pl*** contiene un único predicado el cual permite construir la estructura del juego (la grilla y las pistas del juego).

▸ *init/3*

El predicado ***init/3***, definido de la siguiente manera:

init(+PistasFilas, +PistasColumnas, +Grilla)

Principal función: La principal función del predicado es inicializar la estructura del juego, lo que implica construir una lista de pistas para las filas, una lista de pistas para las columnas y una grilla que contendrá los elementos de las celdas.

II. Predicados del archivo proylcc.pl

El archivo ***proylcc.pl*** contiene predicados que permiten la actualización y consulta del tablero, es decir, son los predicados que permiten actualizar y verificar el estado del juego.

▸ *replace/5*

El predicado ***replace/5***, definido de la siguiente manera:

replace(+X, +XIndex, +Y, -Xs, -XsY)

Principal función: La principal función del predicado es reemplazar la ocurrencia de ***X*** ubicada en la posición ***RowN*** dentro de la lista ***Xs*** por el elemento ***Y***, y almacena el resultado de dicha operación (la grilla actualizada) en ***XsY***.

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** Si **XIndex** es cero, y, además, el elemento de la lista coincide con **X** (esto es, el encabezado de la lista es igual a X) entonces se tiene que el encabezado de la lista de XsY (de la grilla) será **Y**.
- **CR:** Si **XIndex** no es cero, entonces disminuye en 1 el número de **XIndex** (almacenado en XIndexS) y se llama al predicado empleando los valores ***X e Y***, y los datos ***XIndexS***, la cola de la lista ***Xs*** y la cola de la lista **XsY**.

▸ *put/8*

El predicado ***put/8***, definido de la siguiente manera:

put(+Contenido, +Pos, +PistasFilas, +PistasColumnas, +Grilla, -GrillaRes, -FilaSat, -ColSat)

Principal función: La principal función del predicado consiste en actualizar el tablero (Grilla) en la posición ***pos*** con el elemento que está dentro de ***Contenido***.

- El tablero actualizado (con la celda actualizada) es almacenado en ***GrillaRes*** y luego, en ***FilaSat*** y ***ColSat*** se retorna el grado de satisfacción de las pistas de la fila y la columna a las que pertenece el elemento a ser actualizado, devolviendo 1 si se satisfacen las pistas y 0 en caso contrario.
- Se requiere de dos listas de “listas de números” que contienen las pistas del juego. Dichas pistas son las pistas de la fila ***(PistasFilas)*** y las pistas de la columna ***(PistasColumnas)***.

Funcionamiento: El predicado no es recursivo, pero emplea predicados adicionales que si son recursivos, y esto se detalla de la siguiente manera:

1. Empleando el predicado ***replace/5*** se procede a recuperar el elemento que se encuentra en la posición dada ***Row***.
2. Obtenido el resultado ***NewRow***, se procede a verificar si el resultado de sustituir la celda en la posición ***ColN*** de ***Row*** se produjo por un elemento sin instanciar o de reemplazar lo que se encontraba en la celda.

- Si el contenido de la celda Cell se quiere reemplazar por el mismo valor que ya almacenaba, entonces este se borra (se almacena una variable no instanciada)
- En caso contrario, entonces ***NewRow*** es el resultado de haber reemplazado lo que había en ***_Cell*** por ***Contenido*** en la ubicación previamente mencionada.

3. Finalizado este proceso de verificación del resultado de ***NewRow***, se procede a recuperar las listas que almacenan las pistas de la posición dada para ***put/8.*** Esto se hace empleando el predicado recursivo ***recuperar_pistas/3***, lo que se realiza tanto para la fila como para la columna.

4. Con las listas de pistas de la fila y de la columna, se procede a evaluar tanto en las filas como las columnas si las celdas de la fila y la columna dada satisfacen la lista de pistas. El resultado de ambas verificaciones son almacenadas en ***FilaSat*** y ***ColSat*** respectivamente, almacenando 1 si se satisfacen las pistas y 0 en caso contrario

▸ *recuperar_pistas/3*

El predicado auxiliar ***recuperar_pistas/3***, definido de la siguiente manera:

recuperar_pistas(+Pos, +Pistas, -E)

Principal función: La principal función del predicado consiste en obtener la lista de pistas **E** que le corresponde a la posición ***Pos*** a partir de la lista de listas de pistas ***Pistas.*** Encontrada la lista de pistas para la posición ***Pos***, dicha lista es almacenada en ***E.***

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** Si **Pos** es cero, entonces la lista de pistas ***E*** para la posición ***Pos*** corresponde al encabezado actual de la lista de listas de pistas ***Pistas***. Por lo tanto, la lista ***E*** es instanciada con el encabezado actual de ***Pistas***.

- **CR:** Si ***Pos*** es mayor a cero, entonces se sabe que aún no se ha llegado a la lista en cuestión, por lo tanto no importa el encabezado de ***Pistas***. Luego, la posición disminuye en 1 (almacenando dicho valor en Posaux) y se procede a realizar la llamada al predicado ***recuperar_pistas/3*** con Posaux, la cola de la lista ***Pistas*** y ***E*** (quien aún está sin instanciar).

▸ *verificar_pistas_filas/4*

El predicado auxiliar ***verificar_pistas_filas/4***, definido de la siguiente manera:

verificar_pistas_filas(+PistasFilas, +RowN, +NewGrilla, -FilaSat)

Principal funcionamiento: La principal función del predicado consiste en validar si la fila i-ésima ***RowN*** de la grilla satisface a la lista de pistas de dicha fila. En caso de satisfacerla, entonces ***FilaSat*** almacenará **1**, o 0 en caso contrario.

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** A continuación se detallan los casos base que tiene el predicado ***verificar_pistas_filas/4***, que en definitiva son los casos de corte de dicho predicado:
    - Si ***PistasFilas*** no está vacía, la fila ***RowN*** es cero y el encabezado de la lista fila perteneciente a ***NewGrilla*** es ***"#"***, entonces se disminuye en uno la pista del encabezado de ***PistasFilas*** y se emplea el predicado ***rafaga/3*** empleando ***PistasFilas*** con el encabezado disminuido en uno, la cola de la lista de fila de ***NewGrilla*** y ***FilaSat***.
    - Si el encabezado de ***PistasFilas*** es cero, la fila ***RowN*** es cero, y el encabezado de la lista de la fila perteneciente a ***NewGrilla*** es **"#"** entonces se tiene que se encontró un **"#"** extra, por lo que no se tiene la cantidad de celdas consecutivas con **"#"** que indica la pista de la fila, y por ende, no se satisface la fila, plasmando este resultado almacenando un cero en ***FilaSat***.
    - Si ***PistasFilas*** no está vacía, la fila ***RowN*** es cero y el encabezado de la lista fila perteneciente a ***NewGrilla*** es distinto de **"#"**, se emplea el predicado ***no_rafaga/3*** para comprobar si el resto de la fila no es un ráfaga de #'s. Esto último se realiza pasando como parámetro a dicho predicado la lista ***PistasFilas***, la cola de la lista de fila de ***NewGrilla*** y ***FilaSat***.
- **CR:** Si ***PistasFilas*** no es vacía, y ***RowN*** es un número mayor a cero, entonces se debe disminuir en uno el número de fila y llamar al predicado ***verificar_pistas_filas/4*** con la lista ***PistasFilas,*** el número de fila disminuido en uno, la cola de ***NewGrilla*** y ***FilaSat***.

▸ *rafaga/3*

El predicado ***rafaga/3***, definido de la siguiente manera:

rafaga(+PistasFilas, +Fila, -FilaSat)

Principal funcionamiento: La principal función del predicado es verificar si una ráfaga de **"#"** se corresponde con la lista de pistas ***PistasFilas***, donde la lista ***Fila*** corresponde a la lista de celdas de la fila. En caso de satisfacer las pistas, se almacena en ***FilaSat*** un 1, de lo contrario, se almacena un 0.

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** A continuación se detallan los casos base que tiene el predicado ***rafaga/3***, que en definitiva son los casos de corte de dicho predicado:
    - Si la lista ***PistasFilas*** tiene de encabezado cero y la lista ***Fila*** está vacía, por lo tanto, se llegó al final de una ráfaga y se satisficieron las pistas, por lo que se almacena en ***FilaSat*** un 1.
    - Si la lista ***PistasFilas*** tiene encabezado cero, entonces el encabezado de ***Fila*** necesariamente debe ser distinto a ***"#"*** y esto se comprueba empleando el predicado ***no_rafaga/3*** para verificar si existe una ráfaga de X's o vacíos (variables sin instanciar) , lo que se realiza partiendo de la cola de ***PistasFilas***, la cola de ***Fila*** y ***FilaSat***.
    - Si la lista de ***PistasFilas*** tiene encabezado 0 y el encabezado de ***Fila*** es ***"#"*** entonces se encontró un elemento extra a la cantidad requerida, por lo que no satisface la pista y almacena en ***FilaSat*** un 0.
    - Si la lista ***PistasFilas*** tiene encabezado distinto de 0 y el encabezado de ***Fila*** es distinto de ***"#"***, entonces se tiene que no satisface la lista de pistas porque la ráfaga de #'s es más pequeña que lo indicado por la pista , por lo tanto, se almacena en ***FilaSat*** un 0.
    - Si la lista ***PistasFilas*** tiene encabezado igual a 0, y la lista ***Fila*** está vacía, y se tiene que la cola de ***PistasFilas*** no está vacía, entonces existe otra pista que debe ser satisfecha, cosa que no puede ser así, ya que la lista ***Fila*** está vacía. Por lo tanto, se almacena un 0 en ***FilaSat***.
    - Si la lista ***PistasFilas*** tiene encabezado distinto a 0 pero la lista ***Fila*** está vacía, entonces se tiene que no se tiene más celdas que recorrer en ***Fila***, por lo tanto, no se puede satisfacer la fila por que ya no hay más elementos que recorrer y se almacena un cero en ***FilaSat.***
- **CR:** Si el encabezado de ***Fila*** es ***"#"***, entonces disminuye en 1 el número de la pista (que es el encabezado por la lista ***PistasFilas*** y almacena dicho resultado en ***PFaux***) y llama a ***rafaga/3*** con ***PistasFilas*** actualizado su encabezado por ***PFaux***, la cola de la lista ***Fila*** y ***FilaSat***.

▸ *no_rafaga/3*

El predicado ***no_rafaga/3***, definido de la siguiente manera:

no_rafaga(+PistasFilas, +Fila, -FilaSat)

Principal funcionamiento: La principal función del predicado consumir una ráfaga de elementos distintos a #'s (X's o vacíos) y verificar que se corresponde con la lista de pistas ***PistasFilas***, donde la lista ***Fila*** corresponde a la lista de celdas de la fila. En caso de satisfacer las pistas, se almacena en ***FilaSat*** un 1, de lo contrario, se almacena un 0.

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** A continuación se detallan los casos base que tiene el predicado ***rafaga/3***, que en definitiva son los casos de corte de dicho predicado:
    - Si las listas ***PistasFilas*** y ***Fila*** están vacías, entonces se tiene que ya se procesó toda la fila y se verificaron todas las pistas, por lo que se almacena en ***FilaSat*** un 1.
    - Si la lista ***PistasFilas*** está vacía y ***Fila*** no está vacía, y el encabezado de ***Fila*** es un ***"#"*** entonces se encontró un elemento extra a la cantidad requerida , por lo tanto, se almacena en ***FilaSat*** un 0.
    - Si la lista de ***PistasFilas*** tiene encabezado 0 y el encabezado de ***Fila*** es ***"#"*** entonces se encontró un elemento extra a la cantidad requerida, por lo que no satisface la pista y almacena en ***FilaSat*** un 0.
    - Si la lista ***PistasFilas*** no está vacía y la lista ***Fila*** está vacía, entonces no satisface la lista de pistas , por lo tanto, se almacena en ***FilaSat*** un 0.
    - Si la lista ***PistasFilas*** no está vacía, la lista ***Fila*** tiene como encabezado a ***"#"***, entonces debe disminuir en 1 la pista del encabezado de ***PistasFilas*** y emplear el predicado ***rafaga/3*** con ***PistasFilas*** con su encabezado disminuido en 1, la cola de ***Fila*** y ***FilaSat*** para procesaruna rafaga de ***"#".***
- ***CR:***
    - Si ***PistasFilas*** no está vacío y el encabezado de ***Fila*** es distinto de ***"#"***, entonces se llama recursivamente al predicado **no_rafaga/3** con la misma ***PistaFila***, la cola de ***Fila*** y ***FilaSat***.

▸ *verificar_pistas_columnas/4*

El predicado ***verificar_pistas_columnas/4***, definido de la siguiente manera:

verificar_pistas_columnas(+PistasColumnas, +ColN, +NewGrilla, -ColSat)

Principal funcionamiento: La principal función del predicado consiste en validar si la columna i-ésima ***ColN*** de la grilla satisface a la lista de pistas de dicha columna. En caso de satisfacerla, entonces ***ColSat*** almacenará **1**, o 0 en caso contrario.

Funcionamiento: El predicado no trabaja recursivamente, sin embargo, los predicados que emplea si trabajan recursivamente, y esto se detalla de la siguiente manera:

1. Empleando el predicado ***recuperar_columna/3*** se recupera la lista de celdas que abarca toda la columna ***ColN*** que estará en ***Grilla*** y será almacenada en ***Columna.***
2. Partiendo de la lista ***Columna,*** se emplea el predicado ***verificar_pistas_filas/4*** para verificar si dicha columna satisface las pistas de la columna. Almacenará en ***ColSat*** un 1 en caso de que estén satisfechas las pistas, en caso contrario, almacena 0.

▸ *recuperar_columna/3*

El predicado ***recuperar_columna/3***, definido de la siguiente manera:

recuperar_columna(+ColN, +Grilla, -Columna).

Principal funcionamiento: La principal función del predicado consiste en recuperar y almacenar en ***Columna*** la lista ubicada en la columna ***ColN***, la cual está ubicada en la grilla.

Funcionamiento: El predicado trabaja de manera recursiva, y esto se detalla de la siguiente manera:

- **CB:** Si ***Grilla*** y ***Columna*** son listas vacías, entonces significa que se llegó al final del recorrido de las columnas de la Grilla.
- **CR:** Si las listas ***Grilla*** y ***Columna*** tienen elementos que recorrer, entonces se emplea ***recuperar_elemento/3*** para recuperar el elemento ***E*** ubicado en la columna ***ColN*** de la fila **G** en el encabezado de la Grilla.

  Luego, dicho elemento pasará a ser el encabezado de la lista ***Columna*** y la cola de la misma será el resultado que sea almacenado en la próxima llamada de ***recuperar_columna/3*** que es llamado manteniendo fijo a ***ColN*** y pasando la cola de ***Grilla*** y ***Columna***.

▸ *recuperar_elemento/3*

El predicado ***recuperar_elemento/3***, definido de la siguiente manera:

recuperar_elemento(+ColN, +Fila, -E).

Principal funcionamiento: La principal función del predicado consiste en buscar y almacenar en ***E*** el elemento ubicado en la columna ***ColN*** de la lista que representa a la fila.

Funcionamiento: El predicado trabaja de manera recursiva, y parte de la idea de que ***ColN*** es la cantidad de elementos que hay que recorrer hasta llegar al elemento. Esto se detalla de la siguiente manera:

- **CB:** Si ***ColN*** es 0, entonces esto significa que el encabezado actual de la lista ***Fila*** contiene al elemento en cuestión, por lo tanto, dicho elemento es almacenado en **E**.
- **CR:** Si ***ColN*** es distinto de 0, entonces el elemento aún no fue encontrado y, por ende, dicho elemento se encuentra a ***ColN-1*** elementos de distancia. Dicho de otra manera, se debe disminuir en 1 a ColN (almacenando el valor en ColNaux) y se llama al predicado ***recuperar_elemento/3*** con el contador disminuido, la cola de la lista Fila y E.