Relatório Trabalho Prático de Programação Declarativa

Ricardo Mochila – 37762
Inês Veríssimo – 40102
Engenharia Informática

Descrição do Trabalho

O trabalho consistiu no desenvolvimento de um programa na linguagem de programação lógica Prolog.

De acordo com o input recebido, calcula o tamanho da grelha nxm do nonograma e a posição correta de todas as listas lidas, de modo a preencher a matriz final com "." e "X".

Desenvolvimento

Inicialmente lê-se a lista recebida. Esta terá mais 2 listas, a primeira correspondente às linhas da grelha, e a outra às colunas. O primeiro passo é contar quantos elementos(listas) fazem parte das linhas e colunas.

A função count recebe uma lista de listas e conta o número de elementos existentes. Este número de elementos será o número de linhas e colunas.

MakeRow é responsável por criar a matriz nxm e utiliza a constrain *fd_domain* para gerar valores entre 0 e 1 e preencher a grelha.

SumRowCol lê as linhas e as colunas e, caso exista uma lista que tenha mais que um elemento, isto é, em que as posições a serem preenchidas com "X" sejam separadas, soma os elementos da lista de modo à lista ter apenas 1 elemento.

A função applyConstrain verifica se, em cada lista, a repetição de "1" que foram somados em SumRowCol é igual aos existentes na matriz, através da constrain *fd_exactly.*

Transpose servirá para transpor a MatrixCol, que é a matriz que corresponde às colunas. Isto porque, as colunas são recebidas como listas e devem corresponder às posições verticais da matriz.

Separate é a função que por linha da matriz gerada, através da função applyConstrain, vai contar o número de "1"s seguidos e representá-los na forma original, recebida pelo nonogram. Estes números vão ser comparados com a lista das Rows inicial e, caso não sejam compatíveis, falha e volta a aplicar a constrain.

As funções Rewrite e Run vão passar os 1 a "X" e os 0 a ".".

Finalmente, imprime-se a matriz final e o tempo de execução.

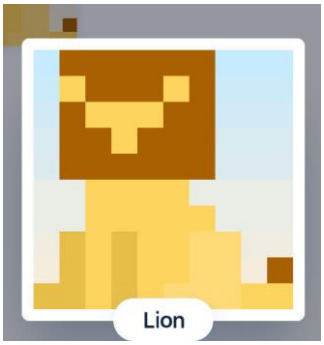
Funcionamento e Conclusão

O programa desenvolve vários nonograms, como por exemplo:

```
| ?- matrix([[[1],[3],[1,1,1],[1],[1]],[[1],[1],[5],[1],[1]]]).
 .  .  X  .  .
 .  X  X  X  .
 X  .  X  .  X
 .  .  X  .  .
 .  .  X  .  .
Time: 0.001ms
```

```
| ?- matrix([[[2], [2,4], [2,6], [8], [1,1],[2,2]],[[2], [3], [1], [2,1], [5], [4], [4,1], [5],[2]]]).
 .  .  .  .  .  X  X  .  .  .  .
 X  X  .  .  X  X  X  X  .  .  .
 X  X  .  X  X  X  X  X  X  .  .
 .  X  X  X  X  X  X  X  X  .  .
 .  .  .  .  X  .  .  X  .  .  .
 .  .  .  X  X  .  X  X  .  .  .
Time: 0.006000000000000001ms
```

```
 .  X  X  X  X  X  X  .  .  .
 .  .  X  X  X  .  X  .  .  .
 .  X  .  .  .  X  X  .  .  .
 .  X  X  .  X  X  X  .  .  .
 .  X  X  X  X  X  X  .  .  .
 .  .  X  X  X  X  .  .  .  .
 .  .  X  X  X  X  X  .  .  .
 .  X  X  .  X  X  X  X  .  .
 .  X  X  .  X  X  X  X  .  X
 X  X  X  .  X  X  X  X  X  X
Time: 0.003000000000000001ms
```



Este trabalho foi importante, pois foi necessário aplicar os conhecimentos adquiridos nas aulas para resolver situações que surgiram durante o desenvolvimento do trabalho, melhorando o nosso pensamento lógico em prolog.