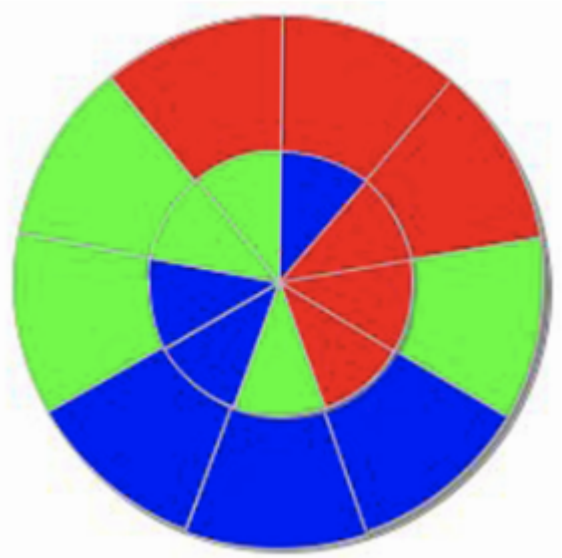
Trabalho Prático de Programação Declarativa

Carlos Valente n°33298
José Serra n°33289

31/12/2018

1 Introdução

Trabalho realizado no âmbito da disciplina de Programação Declarativa, no qual se pretende obter um algoritmo, que segundo diversas regras e restrições resulte na pintura de um disco com N coroas circulares e M sectores, usando K cores distintas.



2 Desenvolvimento

2.1 Sistema Utilizado:

O sistema utilizado para a realização do trabalho foi o swi-prolog.

2.2 Algoritmo:

O algoritmo enunciado pedia-nos a visualização de um disco composto por Ncoroas e Msectores, como tal, achámos que estas poderiam ser representadas como nested lists, em que as lists exteriores seriam as coroas, e as interiores os respetivos sectores do disco.
Este disco seria "pintado" através do predicado pintar/4.

A primeira tarefa a realizar foi a de inicializar a lista de sectores(matriz). Com o predicado inicializar/3 é criada uma lista com N elementos por instanciar (sendo N o número de sectores) e de seguida cada um destes elementos é instanciado com uma lista com M elementos por instanciar sendo M o número de coroas.
O resultado da execução deste predicado é a matriz do seguinte exemplo:

```
?- inicializar(9,2,M), write(M).
[[_5786,_5792],[_5798,_5804],[_5810,_5816],[_5822,_5828],
[_5834,_5840],[_5846,_5852],[_5858,_5864],[_5870,_5876],[
_5882,_5888]]
```

Após inicializada a estrutura, é utilizado o predicado pinta/4, que irá instanciar as coroas de cada sector (cada elemento da matriz), com uma das cores presentes na lista passada no segundo argumento deste predicado e comparar sector a sector. O predicado irá iterar a lista de sectores 2 a 2 comparando cada par de sectores até chegar aos casos base em que sobram apenas 2 ou 3 sectores por pintar.
O predicado pinta_sectores irá por sua vez iterar cada coroa do setor e pintar a coroa instanciando-a com uma das cores da lista de cores.

Nesta fase, apenas restam fazer 2 verificações para que as coroas sejam pintadas de acordo com as regras, isto porque até agora apenas os sectores imediatamente adjacentes foram comparados utilizando o predicado compara_sectores que verifica se 2 sectores adjacentes diferem apenas na cor de uma das suas coroas iterando ambos os sectores comparando coroa a coroa até encontrar uma coroa num dos sectores que seja a unica coroa com cor diferente entre estes dois sectores.
O passo seguinte é comparar o primeiro e o ultimo sectores presentes na lista de sectores, visto que estes são também considerados adjacentes. Para tal, basta obter o primeiro e último elementos da lista e utilizar uma última vez o predicado compara_sectores.
Resta apenas uma última verificação, que se trata de assegurar que todas as configurações de cores sejam únicas, ou seja, que cada sector(lista de coroas) seja único na sua configuração. Este último passo é feito pelo predicado configs_diferentes, que pega num elemento(sector) da lista de sectores e verifica se não o encontra na lista dos restantes elementos(sectores seguintes). Se isto acontecer, significa que todas as configurações/sectores são únicos.

No próximo exemplo é possível observar o resultado final com uma das soluções possíveis para um exemplo de disco que contém: 2 coroas, 9 sectores e as cores vermelho,verde e azul.

```
?- pintar(2, 9, [vermelho, verde, azul], Disco), write(Disco).
[[vermelho,vermelho],[vermelho,verde],[vermelho,azul],[verde,azul],[verde,vermelho],
[verde,verde],[azul,verde],[azul,azul],[azul,vermelho]]
```

3 Conclusão

A realização deste trabalho e consequente elaboração do algoritmo proposto, resultou numa aprendizagem importante relativamente ao pensamento recursivo inerente à utilização de uma linguagem declarativa como o Prolog. Achamos que cumprimos os objetivos enunciados e ficámos satisfeitos com o resultado do nosso trabalho apesar de não termos realizado a prova bónus.