Conteúdo

1 Não determinismo:Geração e Teste

Não determinismo:Geração e Teste

Esquema geral:

```
procura(X):- gera(X), testa(X).
```

slowsort

```
slowsort(X,Y) :-  perm(X,Y), sorted(Y).
```

1.1 N rainhas

N rainhas

Colocar N rainhas num tabuleiro de xadrez $N \times N$ de modo a que nenhuma ataque qualquer outra.

Cada rainha pode mover-se

- na coluna
- na linha
- na diagonal

Solução $[R_1, \ldots, R_N]$

A rainha da coluna i está na linha R_i.

N rainhas

- `[1,...,N]` : garante que cada rainha está em linhas e colunas distintas
- permutar

- verificar se nenhuma rainha ataca outra, isto é, está na mesma diagonal

```
ataca(X,N,[Y|Ys]):- X is Y+N ;X is Y-N.
ataca(X,N,[Y|Ys]):- N1 is N+1, ataca(X,N1,Ys).
```

N rainhas:colocar sucessivamente

Melhoramento: em vez de permutar colocamos as rainhas uma a uma

```
rainhas1(N,Rs):- range(1,N,Ns), rainhas(Ns,[],rs).
rainhas([],Rs,Rs):-!.
rainhas(Naocoloc,Seguras,Rs):- delete(R,Naocoloc,Naocoloc1),
    not ataca(R,Seguras),
    rainhas(Naocoloc1,[R|Seguras],Rs).
```

Conjuntos de Expressões

- `findall/3` `findall(T,G,L)`
  coleciona os termos `T` que satisfazem o objectivo `G` e coloca-os em `L`
  Ex: `findall(X,rainhas(5,X),L).`
- `all/3` (o mesmo mas retira termos repetidos)
- `bagof/3`
- `setof/3`

1.2 Colorir um mapa

Colorir um mapa

Colorir um mapa de modo que duas regiões contíguas não tenham a mesma cor.
Para cada região do mapa

- escolher uma cor
- escolher ou verificar que as regiões vizinhas têm cores diferentes
- Mapa: lista de regiões
- Região: nome, cor, lista das cores de cada vizinho `regiao(N,C,Ns)`

Colorir um mapa

```
colorir([],_).
colorir([R|Rs],Cores):- colorir_reg(R,Cores),
              colorir(Rs,Cores).

colorir_reg(regiao(N,C,Ns),Cores):-
    delete(C,Cores,Cores1),
    members(Ns,Cores1).
```

2 Modificar a Base de Dados

Inspecionar e Modificar Programas

- predicados estáticos: `static`
- predicados dinámicos: `dynamic`

`clause/2`

Acrescentar/apagar cláusulas a um programa:

- `assert/1`, `asserta/1`, `assertz/1`,
- `retract/1`
- `abolish/2`

2.1 MasterMind

Master Mind

Dois jogadores

- o jogador A escolhe uma *chave* de 4 dígitos distintos de 0 a 9
- em cada iteração:
  - o jogador B adivinha uma sequência de 4 dígitos distintos e pergunta
  - o número de dígitos nas posições certas (brancos)
  - o número de dígitos que ocorrem também na chave mas em posições erradas (pretos)
- B ganha, se tem como resposta 4 brancos.

Master Mind

```
mmind(C):- limpa, tentativa(C), verifica(C), anuncio.

tentativa(C):-C=[X1,X2,X3,X4], selects(C,[1,2,3,4,5,6,7,8,9,0]).

verifica(C):- not inconsistente(C), pergunta(C).
```

Master Mind

```
inconsistente(C):-
     tenta(C1,B,P),
     not b_e_p(C1,C,B,P).

b_e_p(C1,C,B,P):-
   exactos(C1,C,B1),
   B=:=B1,
   comuns(C1,C,P1),
   P=:=P1-B.



exactos(X,Y,N):- mesmaposicao(X,Y,0,N).

comuns(X,Y,N):- comuns(X,Y,0,N).
```