Inteligência Artificial 04/05 — Aula 8

Como as turmas P6, P7 e P8 não têm esta aula, não irá ser dada nova matéria. As aulas servirão essencialmente para dar apoio ao trabalho prático, mas para os que gostam realmente de Prolog ficam aqui dois novos desafios, dois problemas que saíram em concursos de programação.

1. Divisões

(Este problema saiu no CENPL'2003)

O PROBLEMA

Como estão certamente fartos de saber, os números racionais são aqueles que se podem obter a partir da divisão de dois números inteiros. Quando passado para a forma decimal, um número racional nem sempre pode ser representado com um número finito de casas decimais. Por exemplo: `1/3 = 0.33333...` Mas, para os números racionais, essa sequência (mesmo que infinita) de casas decimais não pode ser qualquer: a partir de certa altura há sempre uma sequência de algarismos que se repete indefinidamente. Por isso mesmo, até é usual representar esses números na forma decimal com a dita sequência no fim e entre parêntesis.

Por exemplo:

```
1/3 = 0.(3)              8/15 = 0.5(3)
1/7 = 0.(142857)        10/7  = 1.(428571)
```

TAREFA

Escrever um predicado Prolog que, dados dois números inteiros positivos `n` e `m`, calcule a sequência de algarismos que se repete indefinidamente na representação decimal do número racional `n/m`.

OS RESULTADOS

Deverá implementar o predicado `ciclo(N,M,C)` que, quando chamado com `N` e `M` instanciados com números inteiros positivos, devolve em `C` uma lista com a sequência de algarismos que se repete indefinidamente na representação decimal de `N/M`.

Se `N/M` se conseguir representar com um número finito de casas decimais, então em `C` deve ser devolvido `[0]`. Note que, em rigor, essa é uma sequência que se repete indefinidamente: por exemplo, `1/4 = 0.25(0)`.

No caso de haverem várias sequências possíveis (por exemplo `1/11 = 0.(09) = 0.0(90)`), o seu predicado deverá devolver aquela que

dá origem a uma representação mais curta do número (o `0.(09)` é mais curto que o `0.0(90)`, pelo que deverá ser devolvido `[0,9]` e não `[9,0]`).

EXEMPLOS

```
?- ciclo(1,3,C).
C = [3] ? ;
no

?- ciclo(8,15,C).
C = [3] ? ;
no

?- ciclo(1,7,C).
C = [1,4,2,8,5,7] ? ;
no

?- ciclo(10,7,C).
C = [4,2,8,5,7,1] ? ;
no

?- ciclo(1,11,C).
C = [0,9] ? ;
no

?- ciclo(1,4,C).
C = [0] ? ;
no

?- ciclo(7,7,C).
C = [0] ? ;
no
```

2. Autómatos Celulares

(Este problema saiu no CENPL'2003)

INTRODUÇÃO

Um Autómato Celular (AC) é um sistema dinâmico determinístico que consiste num array A de células idênticas que repetidamente mudam o seu estado ou cor de acordo com uma determinada regra de actualização ra. Esta regra é aplicada simultaneamente a todas as células de A em unidades de tempo discretas. Quando ra é aplicada a uma célula particular $x \in A$, a nova cor para x é determinada pelo valor corrente das células numa vizinhança V_x de x. Embora existam muitas escolhas interessantes para A, vamos restringir a nossa atenção a arrays uni-dimensionais de inteiros. A vizinhança de raio r (r é um inteiro positivo) de uma célula x é então o intervalo $V_x = \{y \in Z : |x-y| \leq r\}$. De um modo geral, as células de um AC tomam uma cor de um conjunto de k cores diferentes. No nosso problema tomamos $r = 1$ e $k = 2$. As duas cores possíveis são representadas por 0 e 1.

Para os ACs uni-dimensionais com $r = 1$ e $k = 2$, a regra de actualização ra de uma célula x depende das cores correntes das três células na vizinhança de x, pelo que podemos representar a nova cor de x por $ra(p, q, r)$, onde p, q, r denotam as cores correntes respectivamente das células $x - 1$, x e $x + 1$. Por exemplo, a regra $ra(p, q, r) = r$ desloca a sequência de cores (a sequência de zeros e uns) de uma unidade para a esquerda em cada actualização.

TAREFA

A sua tarefa consiste em escrever um programa *Prolog* que dada a configuração inicial das células; a regra de actualização; e o número N de actualizações a realizar; mostra na primeira linha do ecrã a configuração inicial e nas N linhas seguintes as N configurações que lhe sucedem.

DADOS

O array uni-dimensional A é representado por uma lista de zeros e uns. A regra $ra(p, q, r)$ é representada da maneira óbvia. Por exemplo, a regra

```
p + (1-2p) (q+r-qr)
```

é codificada como

```
p + (1-2*p)*(q+r-q*r)
```

À esquerda da primeira célula (à esquerda do primeiro elemento na lista) e à direita da última célula (à direita do último elemento na lista) supomos um zero para efeitos de aplicação da regra (supomos, respectivamente, que $p = 0$ e $r = 0$).

OS RESULTADOS

O programa deve ser invocado através do predicado:

```
ca(+ConfiguracaoInicial,+RegraActualizacao,+NumActualizacoes)
```

Seguem-se dois exemplos ilustrativos:

```
?- ca([0,0,0,0,1],r,6).
   0 0 0 0 1
   0 0 0 1 0
   0 0 1 0 0
   0 1 0 0 0
   1 0 0 0 0
   0 0 0 0 0
   0 0 0 0 0

?- ca([0,0,0,1,0,0,0],p+(1-2*p)*(q+r-q*r),13).
   0 0 0 1 0 0 0
   0 0 1 1 1 0 0
   0 1 1 0 0 1 0
   1 1 0 1 1 1 1
   1 0 0 1 0 0 0
   1 1 1 1 1 0 0
   1 0 0 0 0 1 0
   1 1 0 0 1 1 1
   1 0 1 1 1 0 0
   1 0 1 0 0 1 0
   1 0 1 1 1 1 1
   1 0 1 0 0 0 0
   1 0 1 1 0 0 0
```

Note que no output não existem espaços no início ou final de linha.