Primeiro Trabalho de Inteligência Artificial

(Prof. Alexandre Direne - 2013/2)

Atenção:

Este trabalho é obrigatório e deverá ser entregue, impreterivelmente, até o dia 6 de outubro 2013 (domingo). A solução é individual e deverá ser arquivada no diretório "~alexd/IA/" onde o nome do arquivo terá como prefixo o seu nome-de-usuário no sistema do laboratório e, como extensão, ".pl" para indicar que seu conteúdo possui um programa em Prolog. Assim, por exemplo, se o seu nome de usuário no sistema fosse "grs09" então o nome o arquivo seria "grs09.pl" (dentro do diretório "~alexd/IA/"). Não se esqueça de proteger completamente o arquivo criado, de maneira a permitir a leitura do mesmo apenas por você! Isso pode ser feito aplicando `chmod og-rwx grs09.pl` antes de efetuar a cópia com a preservação das permissões (`cp -p grs09.pl ~alexd/IA/`). Não se preocupe com as permissões do professor que irá corrigir o trabalho. A correção dos trabalhos será parcialmente automatizada, sendo assim, é importante que todos os arquivos com as soluções individuais estejam no diretório citado acima, dentro do prazo estipulado. Não será permitida a entrega do arquivo por e-mail.

O Jogo Four-in-a-line (Connect-four - Quatro em linha):

No jogo chamado *Four-in-a-Line*, 2 competidores se enfrentam por meio de um tabuleiro retangular constituído por 6 linhas e 7 colunas, utilizando fichas de 2 cores diferentes, uma para cada jogador. A Figura 1 apresenta o esquema do referido tabuleiro em sua posição de início de jogo, onde nenhuma peça foi posicionada ainda.

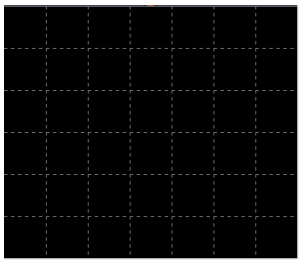

Figura 1: Tabuleiro inicial do jogo Four-in-a-line.

O do jogador **"a"** preenche o tabuleiro empilhando peças brancas (as de cor vermelha nas Figuras 2 e 3) ao nas colunas e o jogador **"b."**, empilhando peças pretas (azuis nas mesmas Figuras). Tente ver detalhes do jogo e preticar um pouco em:

`http://www.mathsisfun.com/games/connect4.html`

Os dois jogadores se alternam a cada movimento. Para efeturar um movimento, um jogador empilha uma, e somente uma, peça de sua cor no topo (máximo de 6 em cada coluna) de qualquer uma das 7 colunas em formação. Não há captura neste jogo. Ganha a partida o jogador que primeiro colocar 4 peças de sua cor justapostas, ou na horizontal, ou na vertical, ou em diagonal. Nas Figuras 2 e 3, por exemplo, vemos o que acontece quando o jogador "b" escolhe a casa $(4, 4)$ para empilhar sua peça (linha = 4, coluna = 4).

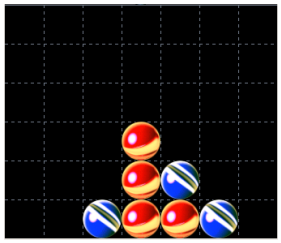

Figura 2: Tabuleiro antes de efetuado o movimento.

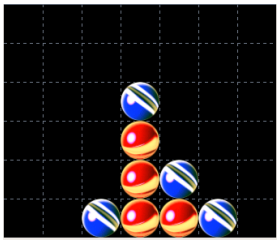

Figura 3: Tabuleiro depois de efetuado o movimento.

Enunciado:

Fazer um predicado ternário (aridade igual a 3) em Prolog capaz de relacionar uma configuração de peças do tabuleiro a um jogador (**"a"** ou **"b"**) assim como com cada uma de suas configurações alcançáveis diretamente de forma lícita a partir de apenas um movimento durante uma partida de Four-in-a-line. Para esgotar completamente todas as possibilidades, o predicado deverá ter capacidade retroativa inter-cláusulas, ou seja, cada uma das configurações possíveis será gerada a cada 1 (um) cálculo do valor de verdade. A representação que deverá ser de uma lista constituída de duas listas dinâmicas. A primeira lista dinâmica

contém a representação das peças do jogador **"a"** ao passo que a segunda, das peças do jogador **"b"**. Cada lista dinâmica é constituída de zero ou mais listas binárias. Cada lista binária representa as coordenadas (linha,coluna) de uma peça do jogador. Por exemplo, a configuração do tabuleiro da Figura 2 é representada da seguinte forma:

```
[ [ [1,4] , [2,4] , [3,4] , [1,5] ] , [ [1,3] , [2,5] , [1,6] ] ]
```

O nome do predicado principal deverá ser `adjacente`. Um exemplo do comportamento de cada execução retroativa do predicado `adjacente` é o seguinte:

```
?- adjacente([[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6]]], b, Proximo).
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[1,1]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[1,2]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[2,3]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[4,4]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[3,5]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[2,6]]] ? ;
Proximo = [[[1,4],[2,4],[3,4],[1,5]],[[1,3],[2,5],[1,6],[1,7]]] ? ;
no
```

Cada um dos termos do predicado `adjacente` representa o seguinte:

- Termo 1 – configuração do tabuleiro antes do próximo movimento (neste exemplo, é a da Figura 2;
- Termo 2 – constante literal que indica qual jogador ("a" ou "b") efeturá o próximo movimento (neste exemplo será a do jogador "b";
- Termo 3 – configuração do tabuleiro alcançado diretamente por um movimento lícito (neste exemplo é a da Figura 3).

Vale a pena notar que há uma situações em que um jogador pode não ter nenhum movimento possível a ser feito (no caso de esgotamento de todas as casas - típico de uma situação de empate, mas que não precisa ser verificada formalmente se ela é mesmo um empate). Diante de tal situação, o comportamento do predicado deverá ser o de levar o Prolog a calcular o valor de verdade `falso` (*i.e.*, no). Um resumo de ativação do predicado nessa situação é o que segue:

```
?- adjacente([[...],[...]], a, Proximo).
no
```

Se você optar pelo Prolog do ambiente Poplog, adicione as seguintes instruções no início do seu arquivo com a solução:

```
:- prolog_language('pop11').
false -> popmemlim;
false -> pop_prolog_lim;
10e7 -> pop_callstack_lim;
true -> popdprecision;
12 -> pop_pr_places;
:- prolog_language('prolog').
```

Obsevações finais:

- Use apenas os compiladores SWI-Prolog ou o Prolog do Poplog;
- Não troque o nome do predicado `adjacente` pois isto dificultará a correção do trabalho.