# Primeiro Trabalho de IA (Busca Heurística em Grafos $OU$) (Prof. Alexandre Direne - 2011/1)

**Atenção:**

Este trabalho é obrigatório e deverá ser entregue, impreterivelmente, até o dia 06 de maio 2011 (sexta-feira). 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 "grs00" então o nome o arquivo seria "grs00.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 grs00.pl` antes de efetuar a cópia com a preservação das permissões (`cp -p grs00.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.

**Enunciado:**

Implementar um predicado binário em Prolog, chamado `completada`, o qual realiza o trabalho de algoritmo de busca heurística por Satisfação de Restrições. O predicado deverá ser adaptado para preencher molduras de palavras cruzadas. Para ilustrar a aplicação do predicado `completada`, deve bastar a carga de seu código e a execução da seguinte consulta:

```
?- completada([[ x1, x2,  x3,  * ,  x4 ],
               [ x5, * ,  x6,  x7,  x8 ],
               [ x9, * , x10, x11, x12 ],
               [ * , * ,  * , x13,  *  ]],
         [pal([x1,x2,x3]), pal([x6,x7,x8]), pal([x10,x11,x12]), pal([x1,x5,x9]),
          pal([x3,x6,x10]), pal([x7,x11,x13]), pal([x4,x8,x12]),
          dif([[x1,x2,x3],[x6,x7,x8],[x10,x11,x12],[x1,x5,x9],
               [x3,x6,x10],[x7,x11,x13],[x4,x8,x12]])]).

d o m * v
a * a t a
o * r i o
* * * a *
Tempo de execucao = 0.13 segundo(s).
Instanciacoes = 21.
yes
```

O primeiro termo de `completada` representa uma moldura retangular de $N$ linhas por $M$ colunas ($4 \times 5$ para o exemplo) da palavra cruzada. As lacunas $x_1, x_2, \ldots, x_k$ ($k = 13$ para o exemplo) deverão ser preenchidas pelo algoritmo de busca para formar palavras válidas do dicionário (ver abaixo). Onde há definição de lacuna (variável) no cruzamento entre uma linha e uma coluna, a letra a ser preenchida precisa coincidir para ambas as palavras (vertical e horizontal). Adicionalmente, as ocorrências dos asteriscos (`*`) indicam que não haverá preenchimento de letra na posição.

O segundo termo de `completada` representa restrições sobre os elementos da moldura. Elas podem ser de dois tipos:

- $pal(x_a, x_b \ldots, x_p)$ em que necessariamente a sequência de letras $x_a, x_b \ldots, x_p$ deve formar uma palavra do dicionário;
- $dif([[x_a, x_b, \ldots, x_p], \ldots, [x_i, x_j \ldots, x_q]])$ em que necessariamente cada sequência de letras é diferente de qualquer outra sequência da lista.

Tais restrições completam as restrições básicas que derivam dos cruzamentos de palavras da moldura.

Note ainda que o domínio (valores possíveis) de cada uma das lacunas $x_1, x_2, \ldots, x_k$ compreende apenas às letras minúsculas do alfabeto (`a` até `z`, incluindo `k`, `w` e `y`).

O dicionário de palavras é bem pequeno e só contém itens de 3 (três) letras para facilitar o trabalho de desenvolvimento. Ele é composto apenas de fatos baseados no predicado `pal` e pode ser encontrado em:

`http://www.inf.ufpr.br/alexd/ia/dicionario_T1_2011_1.pl`

O desempenho de seu predicado será avaliado por meio tanto do consumo de tempo como pela quantidade de instanciações de lacunas da moldura que ele realizar durante a execução. Para medir o consumo de tempo, veja o seguinte código aplicado ao cálculo do fatorial de um número:

```
:- prolog_language('pop11').
false -> popmemlim;
false -> pop_prolog_lim;
:- prolog_language('prolog').

fat(0,1).
fat(X,Y) :-
  Z is X-1,
  fat(Z,Fz),
  Y is X*Fz.

tempo(Total) :-
  Inicio is apply(valof(systime)),
  fat(20000, _ ),
  Fim is apply(valof(systime)),
  Total is (Fim - Inicio)/100.0,!.

?- tempo(T).
T = 4.23 ?
yes
```

Para o cálculo da quantidade de instanciações, veja o seguinte código genérico:

```
iniciadas_as_instanciacoes:-
  retractall(instanciacoes(_)),
  assertz(instanciacoes(0)).

mais_uma_instanciacao :-
  instanciacoes(N),
  retractall(instanciacoes(_)),
  N2 is N+1,
  assertz(instanciacoes(N2)).
```

**Obsevações finais:**

- Pode ser bem interessante conhecer os detalhes da buaca por Satisfação de Restrições apresentados em:

  `http://aima.cs.berkeley.edu/newchap05.pdf`

- Use apenas o compilador POPLOG-Prolog;
- Componha seu programa apenas com predicados criados por você (evite as bibliotecas do Prolog), com exceção dos predicados de tempo de execução;
- Não troque o nome do predicado `completada` pois isto dificultará a correção do trabalho.