Mergulho em PROLOG

O objetivo deste exercício é estudar o funcionamento do PROLOG, além de implementar um demonstrador automático de teoremas nesta linguagem.

O exercício começa instalando uma versão freeware de PROLOG e depois prossegue executando alguns exercícios da aula passada (muito mais fácil, agora). Depois, algumas construções lógicas em prolog. Obviamente é apenas um mergulho na linguagem. Fica o desafio para aprofundar-se na mesma.

Instalando

Procure o SWI-prolog em www.swi-prolog.org, baixe-o e instale-o. Ele já associa os arquivos terminados por `.pl` como arquivos fonte prolog. Note que o pacote é razoavelmente documentado. Dê uma navegada pelos manuais. Instale também o editor embutido, já que ele facilita a operação do ambiente.

Alo mundo

Note a inclinaçao das barras.

1. Chame o notepad
2. escreva: `alo:-write('alo mundo').`
3. salve como `c:\temp\alo.pl`
4. Chame o prolog
5. execute `consult('c:/temp/alo.pl').`
6. escreva: `alo.`

Demonstrador automático

Exemplo 1 Seja o problema dado na aula passada,

> Fox Mulder e Dana Scully trabalham no Arquivo X. Quem trabalha no arquivo X e é crédulo acredita em discos voadores. Quem é cientista não é crédulo. Quem chefia funcionários crédulos é crédulo. Skinner chefia Mulder e Scully. Fox Mulder acredita em DV. Pode-se afirmar que Skinner é crédulo ?

Para responder, crie um arquivo chamado MULDER.PL com qualquer editor e coloque nele as cláusulas acima, já em PROLOG

```
trabalhax(mulder).
trabalhax(scully).
chefia(skinner,mulder).
chefia(skinner,scully).
credulo(X):-chefia(X,Y),credulo(Y).
acreditadv(mulder).
acreditadv(X):-trabalhax(X),credulo(X).
cientista(X):-not(credulo(X)).
```

Chame o PROLOG e carregue (comando FILE-CONSULT) o arquivo acima. A seguir pergunte: Skinner é crédulo ? (em prolog `credulo(skinner).`

Resposta:_____.

Exemplo 2 Outro exemplo, também da aula passada,

> Fox Mulder e Dana Scully trabalham no Arquivo X. Quem trabalha no arquivo X e é crédulo acredita em discos voadores. Quem é cientista não é crédulo. Quem chefia funcionários que acreditam em DV é crédulo. Skinner chefia Mulder e Scully. Fox Mulder é crédulo. Pode-se afirmar que Skinner é crédulo ?

Crie o arquivo MULDER2.PL com o seguinte conteúdo

```
trabalhax(mulder).
trabalhax(scully).
chefia(skinner,mulder).
chefia(skinner,scully).
credulo(X):-chefia(X,Y),acreditadv(Y).
credulo(mulder).
acreditadv(X):-trabalhax(X),credulo(X).
cientista(X):-not(credulo(X)).
```

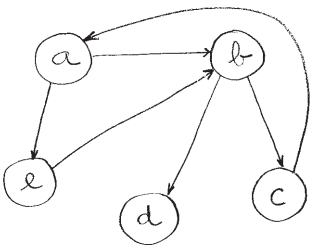
Chame o PROLOG e carregue (comando FILE-CONSULT) o arquivo acima. A seguir pergunte: Skinner é crédulo ? (em prolog `credulo(skinner).`

Resposta:_____.

Pergunte também *Quem é crédulo ?*, em prolog `credulo(X).` Veja o que o PROLOG responde. (Não esqueça de retrucar com ; a cada resposta do PROLOG).

Resposta:_____.

Exemplo 3 Agora, vamos começar a construir uma aplicação. Seja um grafo



Informemos ao PROLOG (arquivo GRAFO.PL)

```
aresta(a,b).
aresta(a,e).
aresta(b,d).
aresta(b,c).
aresta(c,a).
aresta(e,b).
aresta(S,S).
```

Pergunte agora: Existe aresta de a a c?

Resposta:_____.

Quais as arestas que saem de b ? (em prolog `aresta(b,X).`)

Resposta:_____.

Existe um vértice entre a e c? (em prolog `aresta(a,X),aresta(X,c).`)

Resposta:_____.

Qual o significado do comando `aresta(a,X);aresta(X,c).`? (Note a substituição de , por ;.

Resposta:_____.

Interprete o significado das regras `caminho(S,T) :- aresta(S,X),aresta(X,T).`

Resposta:_____.

e `conectado(S,T) :- aresta(S,T);aresta(T,S).`

Responda (e interprete o resultado)

`caminho(a,X).`

Resposta:_____.

Interpretação: _____.

_____.

_____.

_____.

Seja agora uma regra recursiva: `caminho(S,T):-aresta(S,T).` `caminho(S,T):-aresta(S,X),caminho(X,T).` Se o grafo não fosse cíclico funcionaria sem problemas. Como ele é cíclico, e considerando que o PROLOG faz busca em profundidade, ele entra em loop e aborta o programa.

Exemplol 4: Outra regra recursiva

```
ancestral(X,Y) :- mae(X,Y).
ancestral(X,Y) :- pai(X,Y).
ancestral(X,Y) :- mae(X,Z),ancestral(Z,Y).
ancestral(X,Y) :- pai(X,Z),ancestral(Z,Y).
```

Informe as seguintes cláusulas:

```
pai(ana,alfredo).
pai(roberto,alfredo).
mae(eduardo,ana).
pai(francisco,roberto).
pai(alfredo,gustavo).
mae(alfredo,debora).
```

Pergunte quem são os ancestrais de eduardo:

_____.

_____.

Listas

Listas são nativas em PROLOG e têm o mesmo conceito daquele visto em LISP. Uma lista é representada por elementos entre colchetes, por exemplo `[1, 3, 6, 8, 9]` ou `[maria, [jose, berta], carlos]` ou ainda `[]`. O operador `|` separa a cabeça (o primeiro elemento) da cauda de uma lista (a lista que sobra quando se retira a cabeça). Acompanhe a definição do predicado membro:

```
membro(X,[X|T]).
membro(X,[Y|T]):-membro(X,T).
```

Uma maneira alternativa de definição (usando variável anônima) é

```
membro(X,[X|_]).
membro(X,[_|T]):-membro(X,T).
```

Exemplo 5 Defina a função `membro` e depois pergunte se a pertence à lista [b,c,d,e,a,s,f]

Resposta:_____.

Pergunte se josé pertence à lista [maria, [josé, berta], carlos]

Resposta:_____.

A seguir, a função que calcula a quantidade de elementos em uma lista

```
elems([],0).
elems([H|T], X) :- elems(T, Y), X is Y + 1.
```

Exemplo 6: Hanoi Finalmente, a torre de Hanoi

```
hanoi(N) :- move(N, left, center, right).
move(0, _, _, _) :- !.
move(N, A, B, C) :-
  M is N-1,
  move(M, A, C, B), inf(A, B), move(M, C, B, A).
inf(X, Y) :-
  write('mova do pino '),write(X),
      write(' p/ pino '), write(Y),nl.
```

Rode o programa com a configuração padrão de 4 discos. Responda quantos movimentos foram feitos:

_____.