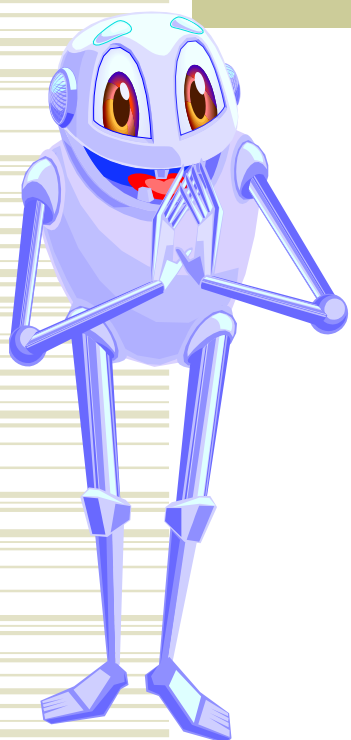


Introdução à Inteligência Artificial



Exercícios Resolvidos **Listas e Busca em Prolog**

Prof. Dr. Alneu de Andrade Lopes
LABIC - ICMC - USP São Carlos

Listas

% elemento de uma lista (elemento/2)

% elemento(X,L). X é elemento de L se é a cabeça da lista ou se é

% elemento do resto da lista.

elemento(X,[X|_]).

elemento(X,[Y|R]):-

 elemento(X,R).

% concatena duas listas (append/3)

% uma lista vazia concatenada com uma lista gera a própria lista.

% ou uma lista com cabeça X e resto R1 concatenada com uma lista L

% tem com resultado uma lista também com cabeça X e resto formado

% pela concatenação de R1 e L.

append([],L,L).

append([X|R1],L,[X|R2]):-

 append(R1,L,R2).

Cont.

```
% número de elemento de uma lista (cardinalidade/2)
% uma lista vazia tem 0 elementos,
% uma lista [X|R] tem 1 elemento (X) mais a quantidade de
% elementos em R
cardinalidade([],0).
cardinalidade([X|R],N) :-
    cardinalidade(R,N1),
    N is 1 + N1.

% remove uma ocorrência de X
% se X ocorre na cabeça da lista, remove a cabeça
% se não, remove X da cauda.
remove_um(X,[X|T],T).
remove_um(X,[Y|T],[Y|T1]) :-
    remove_um(X,T,T1).
```

- ◆ Considere o grafo representado pelos fatos Prolog abaixo.

aresta(a,b). aresta(a,c). aresta(a,d). aresta(a,e). aresta(d,j).
aresta(c,f). aresta(c,g). aresta(f,h). aresta(e,k). aresta(f,i).
aresta(x,y), aresta(y,z). aresta(z,x). aresta(y,u). aresta(z,v).

- ◆ a) escreva um predicado prolog *conectado*(No1, No2), que retorne verdadeiro se houver um caminho entre No1 e No2 no grafo.



conectado(A,B):-

aresta(A,B).

% A é conectado B por uma aresta.

conectado(A,C) :-

% A é conectado a C se houver

aresta(A,B),

% uma aresta de A para B e uma

conectado(B,C).

% conexão (caminho) entre B e C.

Busca em profundidade

- ♦ b) escreva um predicado que retorna um caminho entre dois nós, *caminho*(No1, No2, Caminho), usando busca em profundidade, evitando visitar um nó mais de uma vez.



profundidade(Caminho, No_meta, No_meta, [No_meta|Caminho]).

profundidade(Caminho, No, No_meta, Sol):-

aresta(No,No1),

not(pertence(No1, Caminho)),

profundidade([No|Caminho], No1, No_meta, Sol).

Exercícios Prolog (busca)


- ♦ Resolva o problema das 8 rainhas, isto é, como posicionar 8 rainhas em um tabuleiro de xadrez, de tal modo que não ocorra ataque entre elas.

8 rainhas

- ♦ como não se pode ter duas rainhas em uma mesma linha, o problema consiste em encontrar as colunas apropriadas.
- ♦ Representação (Prolog)
- ♦ ? – pos_rainhas([[1,Y1],[2,Y2],[3,Y3],[4,Y4],[5,Y5],[6,Y6],[7,Y7],[8,Y8]]).
- ♦ nao_ataca([X,Y],[[X1,Y1]|Outras]) % [X,Y] é a nova posição válida se já
% existem [[X1,Y1]|Outras] posicionadas.

- ♦ `pos_rainha([]).`
- ♦ `pos_rainha([[X,Y]|Outras):-` % A nova posição válida (X,Y)
- ♦ `pos_rainha(Outras),` % não pode atacar as que já
- ♦ `pertence(Y,[1,2,3,4,5,6,7,8]),` % estão no tabuleiro (Outras)
- ♦ `nao_ataca([X,Y],Outras).`
- ♦ `pertence(X,[X|_]).`
- ♦ `pertence(X,[Y|R]) :- pertence(X,R).`
- ♦

- ♦ `nao_ataca(_,[]).` % não ataca se não tem nenhuma
% no tabuleiro, ou
- ♦ `nao_ataca([X,Y],[[X1,Y1]|Outras]) :-` % [X,Y] é a nova posição válida
 `Y \= Y1,` % se já existe
- ♦ `T is Y1 - Y, T2 is X1 - X, T \= T2,` % [[X1,Y1]|Outras] posicionadas.
- ♦ `T3 is X - X1, T \= T3,`
- ♦ `nao_ataca([X,Y], Outras).`



caminho(No_inicial, No_meta, Solucao):-
profundidade([],No_inicial, No_meta, Sol_inv),
inverte(Sol_inv,Solucao).



Predicados Auxiliares



- ◆ bagof
- ◆ findall
- ◆ setof

Predicados auxiliares

- ♦ %ache todos X, onde uma condição Y é satisfeita e retorne uma lista Z com todos os X.
- ♦ `ache_todos(X,Y,Z) :-`
- ♦ `bagof(X,Y,Z),!`
- ♦ `ache_todos(_,_,[]).`

- ♦ `concatena([],L,L).`
- ♦ `concatena([X|Y],L,[X|Lista]):-`
- ♦ `concatena(Y,L,Lista).`
- ♦ `inverte([],[]).`
- ♦ `inverte([X|Y],Lista):-`
- ♦ `inverte(Y,Lista1),`
- ♦ `concatena(Lista1,[X],Lista).`

Busca em largura

- ◆ % estende a fila até um filho N1 de N, verificando se N1 não pertence a fila
- ◆
- ◆ `estende_ate_filho([N|Trajetoria], [N1,N|Trajetoria]):-`
- ◆ `aresta(N,N1),`
- ◆ `not(member(N1,Trajetoria)).`

% resolução por largura

- ◆ resolve_largura(No_Inicial,No_Meta, Solucao):-
- ◆ busca_em_largura([[No_Inicial]],No_Meta,Sol1),
- ◆ inverte(Sol1,Solucao).
- ◆
- ◆ busca_em_largura([[No_Meta|T]|T1],No_Meta,[No_Meta|T]).
- ◆
- ◆ busca_em_largura([T|Fila],No_Meta,Solucao):-
- ◆ ache_todos(Filho,estende_ate_filho(T,Filho),Lista),
- ◆ concatena(Fila,Lista,FilaExtendida),
- ◆ busca_em_largura(FilaExtendida,No_Meta,Solucao).