Programação Lógica

Prolog: controle de retrocesso, predicado especial corte (!), verificação de tipos, exemplos

Profa Heloisa de Arruda Camargo

Priscilla de Abreu Lopes (PESCD)
priscilla_lopes@dc.ufscar.br

Controle de Retrocesso

- O retrocesso (backtracking) é um processo pelo qual todas as alternativas de solução para uma dada consulta são tentadas exaustivamente.
- No Prolog, o retrocesso é automático.

Controle de Retrocesso

- É possível controlá-lo através de um predicado especial chamado corte, notado por !
- Visto como uma cláusula, seu valor é sempre verdadeiro. Sua função é provocar um efeito colateral que interfere no processamento padrão de uma consulta.

Controle de Retrocesso

- Pode ser usado em qualquer posição no lado direito de uma regra.
- O corte é adequado às situações onde regras diferentes são aplicadas em casos mutuamente exclusivos.
- Faz com que o programa se torne mais rápido e ocupe menos memória.

Controle de Retrocesso

- Construir um programa Prolog para implementar a função:

$$f(x) = \begin{cases} 0 & \text{se } x < 3 \\ 2 & \text{se } x >= 3 \text{ e } x < 6 \\ 4 & \text{se } x > 6 \end{cases}$$

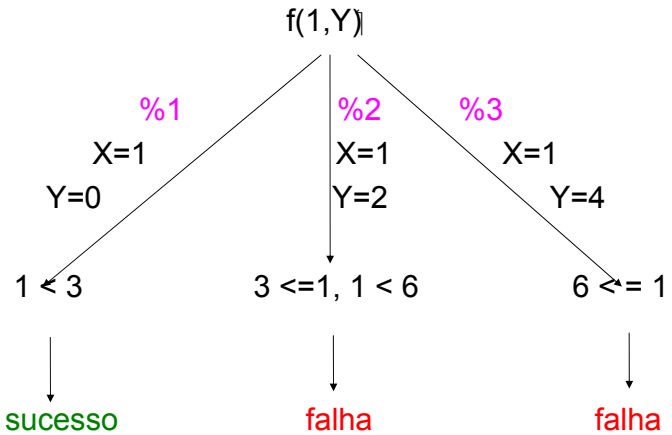
Controle de Retrocesso

```
f(X,0) :- X < 3.          %1
f(X,2) :- 3 =< X, X < 6 . %2
f(X,4) :- 6 =< X.         %3
```

Consulta:

```
?- f(1,Y).
   Y = 0 ;
   no
```


Controle de Retrocesso



Controle de Retrocesso

- Sabemos, no momento da programação, que as regras representam casos mutualmente exclusivos.
- O uso do corte torna-se conveniente, para evitar esforço de busca desnecessário e tornar a execução da consulta mais eficiente.

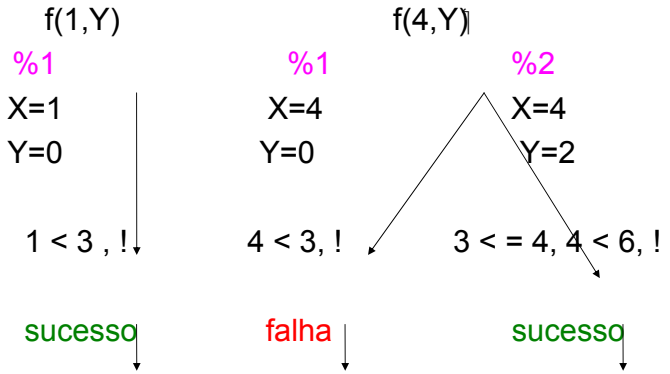
Controle de Retrocesso

```
f(X,0) :- X < 3,!.          %1
f(X,2) :- 3 =< X, X < 6, !. %2
f(X,4) :- 6 =< X.           %3
```

Consulta:

```
?- f(1,Y).
?- f(4,Y).
```


Controle de Retrocesso



Controle de Retrocesso

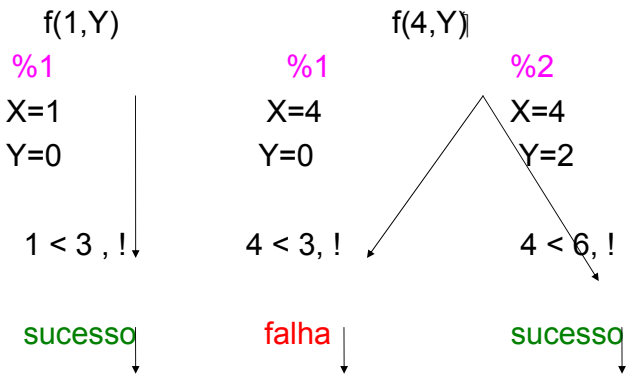
- Este tipo de corte é chamado de corte verde: se for retirado, o programa tem exatamente o mesmo significado.
- Altera-se apenas a eficiência da execução.

Controle de Retrocesso

- O corte pode também ser usado para tornar o programa mais compacto, sem ter que escrever explicitamente as condições de aplicação de cada regra.

```
f(X,0) :- X < 3, ! .   %1
f(X,2) :- X < 6, ! .   %2
f(X,4).                %3
```

Controle de Retrocesso



Controle de Retrocesso

- Este tipo de corte é chamado de corte vermelho: quando retirado, o programa tem significado diferente, geralmente produzindo resultados errados.

Controle de Retrocesso

- Com o uso do corte, vários programas já estudados podem ser modificados para ficarem mais eficientes.
- Para usar o corte, deve ser analisada a operação que se espera realizar com o predicado.

Exemplos

- Contar o número de ocorrências de um dado elemento no primeiro nível de uma lista

```
conta_ocorr(Elem,[ ],0):- !.
conta_ocorr(Elem,[Elem|Y],N) :-
    conta_ocorr(Elem,Y,N1),
    N is N1 + 1,
    !.
conta_ocorr(Elem,[Elem1|Y], N) :-
    Elem \== Elem1,
    conta_ocorr(Elem,Y,N).
```

Exemplos

- Eliminar todas as ocorrências de um elemento de uma lista

```
del_todas(Elem,[ ],[ ]) :- !.
del_todas(Elem, [Elem|Y], Z) :-
    del_todas(Elem,Y,Z),
    !.
del_todas(Elem,[Elem1|Y], [Elem1|Z]) :-
    Elem \== Elem1,
    del_todas(Elem,Y,Z).
```


Exercício

- Modificar o exercício de separação de lista de números utilizando o predicado corte.

```
?- separa_sz_corte([5,4,9,-8,-10,0,2,34,56,-77],P, N).
P = [5, 4, 9, 2, 34, 56],
N = [-8, -10, -77] .
```


Verificação de Tipos

- Termos para verificação de tipo
  - atom(X) - testa se X é átomo
  - atomic(X) - testa se X é átomo ou número
  - compound(X) - testa e X é um termo composto
  - float(X) - testa se X é número em ponto flutuante
  - integer(X) - testa se X é número inteiro
  - nonvar(X) - testa e X não é variável

Verificação de tipos

- Termos para verificação de tipo
  - number(X) - testa se X é número em ponto flutuante ou inteiro
  - simple(X) - verifica se X é átomo, número ou variável
  - var(X) - testa se X é variável não instanciada
  - type(X,Y) - retorna o tipo de um termo X(ver tabela abaixo)

Verificação de tipos

- Valores retornados por type(X,Y)
  - 0 variável
  - 1 inteiro
  - 2 número em ponto flutuante
  - 3 átomo
  - 4 string
  - 5 lista vazia
  - 6 lista
  - 7 n-upla
  - 8 conjunção
  - 9 disjunção

Exemplo

- Construir um programa para desparentizar uma lista, ou seja, eliminar os delimitadores das listas internas, exceto da lista vazia.

?- desparentize([a,[b],c,[d,[e]],f],L)

L = [a,b,c,d,e,f]

Exemplo

```
desparentize([ ], [ ]).
desparentize([X|Y], [X|Z]) :-
    not type(X,6),
    desparentize(Y,Z),
    !.
desparentize([X|Y], Z) :-
    desparentize (X, X1),
    desparentize (Y, Y1),
    conc(X1, Y1,Z).
```