Programação Declarativa: Linguagem Prolog II

Licenciatura em Engenharia Informática

2014–2015

Vitor Beires Nogueira

Escola de Ciências e Tecnologia
Universidade de Évora

Cuts e negação

Outline

Cuts e negação

- Cuts verdes: expressando determinismo
- “Cuts” vermelhos
- Negação

Merge (sem cuts)

Exemplo

```
/* merge(Xs,Ys,Zs) :-
   Zs is an ordered list of integers obtained
   from merging the ordered lists of integers Xs
   and Ys.*/

merge(Xs, [], Xs).

merge([], Xs, Xs).

merge([X|Xs], [Y|Ys], [X|Zs]) :-
        X < Y,
        merge(Xs, [Y|Ys], Zs).

merge([X|Xs], [Y|Ys], [X,Y|Zs]) :-
        X =:= Y,
        merge(Xs, Ys, Zs).

merge([X|Xs], [Y|Ys], [Y|Zs]) :-
        X > Y,
        merge([X|Xs], Ys, Zs).
```

Consideração sobre o cut

Definição (Cut)

1. O cut "corta" todas as cláusulas abaixo dele.
2. O cut "corta" todas as soluções alternativas à conjunção de goals que aparecem à sua esquerda na cláusula.
3. O cut não afecta os goals à sua direita na cláusula.

O cut

- pode ser utilizado para expressar o determinismo
- pode aumentar a eficiência de um programa
- diz-se que é "verde" quando a sua inserção/remoção não altera o significado do programa.

Merge com cuts

Exemplo

```
/* merge(Xs,Ys,Zs) :-
   Zs is an ordered list of integers obtained
   from merging the ordered lists of integers Xs
   and Ys.*/

merge(Xs, [], Xs):- !.

merge([], Xs, Xs):- !.

merge([X|Xs], [Y|Ys], [X|Zs]) :-
        X < Y,
        !,
        merge(Xs, [Y|Ys], Zs).

merge([X|Xs], [Y|Ys], [X,Y|Zs]) :-
        X =:= Y,
        !,
        merge(Xs, Ys, Zs).

merge([X|Xs], [Y|Ys], [Y|Zs]) :-
        X > Y,
        !,
        merge([X|Xs], Ys, Zs).
```

Merge com cuts (gráfico)

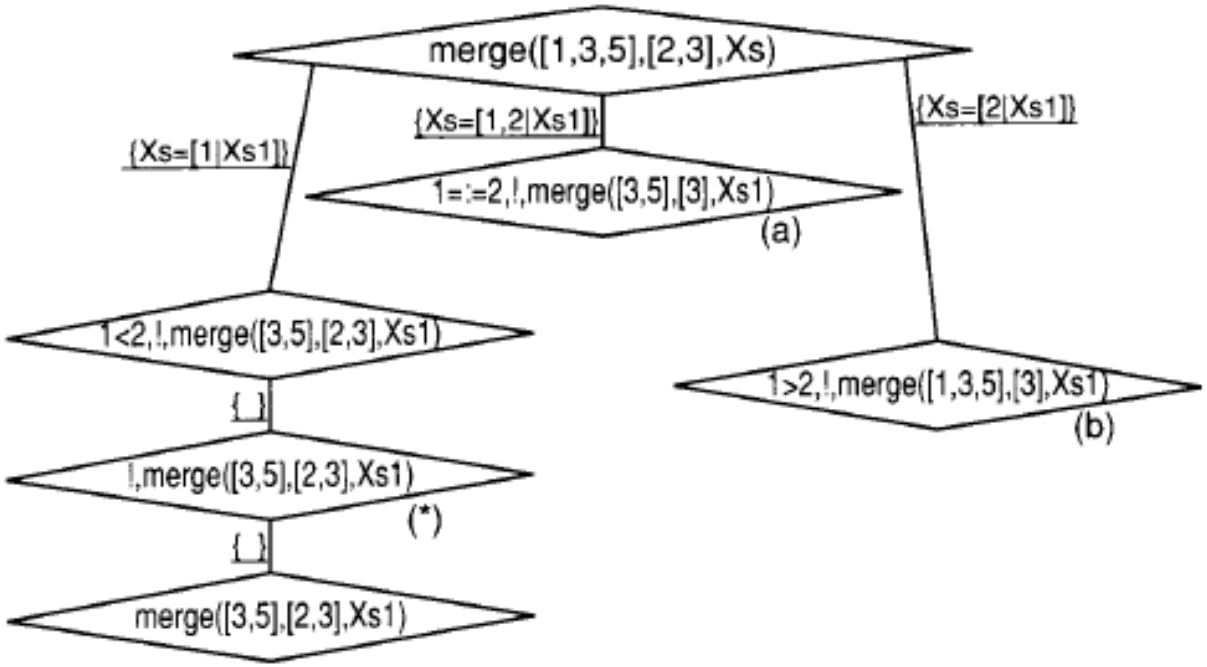

Figure 11.1 The effect of cut

Mínimo com cuts

Exemplo

```
/*
   minimum(X, Y, Min) :- Min is the minimum
   of the numbers X and Y.
*/
minimum(X, Y, X) :-
        X =< Y,
         !.

minimum(X, Y, Y) :-
        X > Y,
         !.
```

Omitindo condições implicitas

Exemplo

```
minimum(X,Y,X) :- X =< Y, !.
minimum(X,Y,Y).
```

- Qual é a resposta à query `?- minimum(2,5,5).`
- true

Exemplo

```
minimum(X,Y,Z) :- X =< Y, !, Z = X.
minimum(X,Y,Y).
```

“Cuts” vermelhos

- “Cuts” cuja presença/ausência altera o significado do programa são designados por “Cuts” vermelhos.
- Estes “cuts” são bastante utilizados quando pretendemos omitir condições implícitas.
- Muitas das vezes as condições implícitas corresponde à negação de condições prévias.
- Uma vez que alteram o comportamento procedimental do programa devem ser utilizados com muito cuidado.

Apagar um elemento de uma lista

Exemplo ("Cut" verde)

```
/* delete(Xs,X,Ys) :-  Ys is the result of
   deleting all occurrences of X from the
  list Xs. */
deleteg([X|Xs],X,Ys) :- !, deleteg(Xs,X,Ys).
deleteg([X|Xs],Z,[X|Ys]) :-
        Z \== X, !,
        deleteg(Xs,Z,Ys).
deleteg([],_X,[]).
```

Exemplo ("Cut" vermelho)

```
delete([X|Xs],X,Ys) :- !, delete(Xs,X,Ys).
delete([X|Xs],Z,[X|Ys]) :- !, delete(Xs,Z,Ys).
delete([],X,[]).
```

- Neste caso com o "cut" vermelho é ligeiramente mais eficiente mas é mais difícil de ler e de modificar

“Cut” vermelho para evitar a negação

If . . . Then . . . Else

```
/* if_then_else(P,Q,R) :- either P and Q
   or not P and R */

if_then_else(P,Q,R) :-
   P, !, Q.

if_then_else(P,Q,R) :-
   R.
```

“Cut” vermelho para modificar o significado

Member check

```
member(X,[X|Xs]) :- !.

member(X,[Y|Ys]) :-
   member(X,Ys).
```

Exemplo (Not)

```
/*
    \+ X :- X is not provable.
*/
:-op(900,fy,\+).

\+ X :-
  X,
  !,
  fail.

\+ X.
```

- A ordem das regras é fulcral no programa acima
- `\+ Goal` nunca instancia os argumentos de `Goal`
- Esta implementação é uma versão incompleta da negação por falha