Programação Declarativa: Linguagem Prolog II

Licenciatura em Engenharia Informática

2013–2014

Vitor Beires Noguera

Escola de Ciências e Tecnologia
Universidade de Évora

Índice

Negação

Predicados extra-lógicos

- Input/Output
- Acesso e alteração do programa em runtime
- Programas interactivos
- Ciclos “failure-driven”

Negação

Negação: particularidades

Exemplo

A query `\+ (p(X),q(X))` falha relativamente ao programa

```
p(s(X)) :- p(X).
q(a).
```

Exemplo

A query `unmarried_student(X)` falha relativamente ao programa:

```
unmarried_student(X) :-
   \+ married(X),
   student(X).

student(bill).
married(joe).
```

O programador deve assegurar-se que os goals negados ficam “ground” antes de serem resolvidos.

Não unificável

Exemplo

```
/*
   X \\= Y :- X and Y are not unifiable.
*/
:- op(700, xfx, \\=).

X \\= X :-
        !,
        fail.

X \\= Y.
```

Negação: mais exemplos

Exemplo (Disjuntos)

```
disjoint(Xs,Ys) :-
   not(member(Z,Xs), member(Z,Ys)).
```

Exemplo (Cut-fail como "cuts" verdes)

Se adicionar a cláusula abaixo ao programa para `ground/1` não altera o seu significado:

```
ground(Term) :- var(Term), !, fail.
```

Negação

Predicados extra-lógicos

- Input/Output
- Acesso e alteração do programa em runtime
- Programas interactivos
- Ciclos “failure-driven”

read/1 e write/1

- O predicado `read(X)` lê um termo da stream de input actual e unifica com `X`
- O predicado `write(X)` escreve o termo `X` na stream de output actual.
- ambos não tem soluções alternativas em *backtracking*
- `write/1` sucede sempre ao passo que `read/1` pode falhar.

write_list/1

```
write_list([X|Xs]) :-
  write(X),
  write_list(Xs).

write_list([]) :-
  nl.
```

Predicados para aceder e alterar o programa

- O predicado `clause/1` pode ser utilizado para aceder às cláusulas de um programa.
- A directiva `dynamic` permite indicar predicados dinámicos, isto é, predicados que podem ser alterados.
- o predicado `asserta(Clausula)` (`assertz(Clausula)`) adiciona a `Clausula` como a primeira (última) do predicado correspondente.
- o predicado `retract(C)` remove a primeira cláusula do programa que unifica com `C`

Alterar um programa

Código

```
:- dynamic(membro/2).

membro(X,[X|_]).

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

Exemplo

Considerando o programa acima, qual o resultado de cada uma das queries:

1. `?- clause(membro(X,Y),Z).`
2. `?- retract(membro(X,[X|Y])).`
3. `?- clause(membro(X,Y),Z).`
4. `?- asserta(:-(membro(X, [X| _]), true)).`

Código

```
/*
    echo :- An interactive loop.
*/
echo :-
        read(X),
        echo(X).

echo(X) :-
        last_input(X),
        !.

echo(X) :-
        write(X),
        nl,
        read(Y),
        !,
        echo(Y).

last_input(exit).
```

Eco “failure-driven”

Código

```
echo :-
        repeat_,
        read(X),
        echo(X),
        !.

echo(X) :-
        last_input(X),
        !.

echo(X) :-
        write(X),
        nl,
        fail.

repeat_.
repeat_ :-
        repeat_.

last_input(exit).
```

Código

```
/* consult_(File) :- The clauses of the program
   in the file File are read and asserted.
*/

consult_(File) :-
        open(File,read,DD),
        consult_loop(DD),
        close(DD).

consult_loop(DD) :-
        repeat,
        read(DD,Clause),
        process(Clause,DD), !.

process(Clause,DD) :-
        at_end_of_stream(DD).
process(Clause,DD) :-
        assertz(Clause), fail.
```