Ficha Prática 5 - Continuação

José Creissac Campos, António Nestor Ribeiro
{jose.campos, anr}@di.uminho.pt
PPIII (LESI)

2002/03

Objectivos

1. Resolução da Ficha 5.
2. Praticar a utilização de `findall`, `setof` e `bagof`.

Exercícios

`findall`, `setof` e `bagof`

1. Relembre a Secção *Factos, queries e regras* da Ficha Prática 1:

   (a) Escreva o predicado `alunos_de_ppiii/1` que define a lista de alunos inscritos a `ppiii`.

```
aluno(joao,ppi).
aluno(pedro,ppi).
aluno(maria,ppiii).
aluno(rui,ppiii).
aluno(manuel,ppiii).
aluno(pedro,ppiii).
aluno(rui,ppiv).

% alunos_de_ppiii/1 dá a lista dos alunos inscritos a ppiii

alunos_de_ppiii(L):- setof(A,aluno(A,ppiii),L).
```

(b) Escreva agora o predicado `alunos_de/2` que define a lista de alunos de uma cadeira[1].

```
% alunos_de/2 define a lista de alunos de uma cadeira

alunos_de(C,L):- setof(A,aluno(A,C),L).
```

(c) Escreva ainda o predicado `cadeirão/1` que define qual a cadeira com maior número de alunos inscritos.

```
% cadeirao/1 dá a cadeira com maior número de inscritos
% A estratégia consiste em criar uma lista do tipo
% [4/ppi, 6/ppii, 8/ppiii] e ao invertê-la obter na primeira
% posição a cadeira com mais inscritos.
% Atenção que o setof dá como resultado uma lista ordenada por ordem
% crescente.

cadeirao(C):- setof(Num/C,L^(alunos_de(C,L),length(L,Num)),R)
              ,reverse(R,[_/C|_]).
```

(d) Escreva os predicados `cadeiras_do_rui/1` (a que cadeiras está o `rui` inscrito), `cadeiras_de/2` (a que cadeiras está um aluno inscrito) e `atarefado/1` (quem é que está inscrito a maior número de cadeiras).

```
% cadeiras_do_rui/1 que dá as cadeiras a que o rui está inscrito

cadeiras_do_rui(L):- setof(C,aluno(rui,C),L).

% cadeiras_de/2 dá as cadeiras de um determinado aluno

cadeiras_de(A,L):- setof(C,aluno(A,C),L).

% atarefado/1 dá o aluno com maior número de cadeiras.
% A estratégia será semelhante à feita para determinar a cadeira
% com mais inscritos


atarefado(X):- setof(Num/A,L^(cadeiras_de(A,L),length(L,Num)),R)
               ,reverse(R,[_/X|_]).
```

[1] `alunos(C,L)` se `L` é a lista de alunos inscritos à cadeira `C`.

```
% mais_consultada/1 dá a página com mais consultas

mais_consultada(Pag):- setof(N/P,L^(bagof(A,consultou(A,P),L),length(L,N)),R)
                       ,reverse(R,[_/Pag|_]).

% O que queremos obter é
%
% ?- setof(N/P,L^(bagof(A,consultou(A,P),L),length(L,N)),R).
%
% N = _G151
% P = _G152
% L = _G159
% A = _G154
% R = [1/p3, 1/p4, 3/p2, 4/p1]
%
% e depois fazer o reverse.
```

2. Considere agora uma Base de Conhecimento onde são armazenados factos `consultou/2` com informação sobre as páginas Web que cada utilizador de um dado ISP consultou. Tome como exemplo a seguinte Base de Conhecimento:

```
consultou(util1, p1).          consultou(util2, p1).
consultou(util1, p2).          consultou(util2, p1).
consultou(util1, p3).          consultou(util2, p1).
consultou(util1, p4).          consultou(util3, p2).
                               consultou(util4, p2).
```

   (a) Escreva o predicado `mais_consultada/1` que define qual a página mais consultada[2].

   (b) Escreva o predicado `melhor_cliente/1` que define qual o utilizador que fez maior número de consultas.

   (c) Escreva o predicado `com_mais_clientes/1` que define a página com maior número de utilizadores diferentes.

   (d) Escreva o predicado `util_por_pagina/1` que define uma lista de pares página/lista de utilizadores que consultaram a página.

[2]Sabendo que `setof` cria uma lista ordenada, como poderá escrever o predicado de modo a evitar ter que escrever um predicado que defina o máximo de uma lista?

```
% melhor_cliente/1 que dá o utilizador com maior número de consultas

melhor_cliente(Cl):- setof(Num/Cl,L^(bagof(P,consultou(Cl,P),L),length(L,Num)),R
                     ,reverse(R,[_/Cl|_]).

% Mais uma vez, o que queremos obter é:
%
% ?- setof(Num/Cl,L^(setof(P,consultou(Cl,P),L),length(L,Num)),R).
%
% Num = _G151
% Cl = _G152
% L = _G159
% P = _G155
% R = [1/util3, 1/util4, 3/util2, 4/util1]


% com_mais_clientes/1, dá a página com maior número de utilizadores diferentes.
% O facto de serem utilizadores diferentes obriga-nos a utilizar o setof.

com_mais_clientes(Pag):- setof(Num/P,L^(setof(Cl,consultou(Cl,P),L),length(L,Num
                         ,reverse(R,[_/Pag|_]).

% Mais uma vez, temos:
% ?- setof(Num/P,L^(setof(Cl,consultou(Cl,P),L),length(L,Num)),R).
%
% Num = _G151
% P = _G152
% L = _G159
% Cl = _G154
% R = [1/p3, 1/p4, 2/p1, 3/p2] ;


% util_por_pagina/1 dá uma lista de pares (página,lista de utilizadores)

util_por_pagina(R):- setof(P/L,setof(Cl,consultou(Cl,P),L),R).

% ?- setof(P/L,setof(Cl,consultou(Cl,P),L),R).
%
% P = _G151
% L = _G152
% Num = _G162
% Cl = _G154
% R = [p1/[util1, util2], p2/[util1, util3, util4], p3/[util1], p4/[util1]]
```

3. Utilizando os conhecimentos adquiridos anteriormente apresente **duas** implementações diferentes para o problema que a seguir se detalha. Uma das implementações deve recorrer aos meta-predicados (ex: setof, bagof, etc.) e outra deve ser feita utilizando apenas recursividade sobre listas.

   Considere que tem uma lista com informação relativa a livros e respectivos autores. Essa lista assume a forma que a seguir se exemplifica:

```
[bib(asterix,[uderzo, gosciny]),
 bib(maias,[eca_queiroz]),
 bib(farpas,[eca_queiroz,ramalho_ortigao]),
 bib(ubik,[philip_dick])
]
```

   O objectivo deste problema é construir, com a mesma informação, uma outra lista que permita mais facilmente responder à questão *"Que livros escreveu determinado autor?"*. A lista resultante deve ter o seguinte formato:

```
[auth(eca_queiroz,[maias,farpas]),
 auth(gosciny,[asterix]),
 auth(philip_dick,[ubik]),
 auth(ramalho_ortigao,[farpas]),
 auth(uderzo,[asterix])
]
```

Solução com a utilização de meta-predicados:

```
% inverte/2
% inverte(Lbib,Lauth) :- Lbib é uma lista de bib(entry,lista de autores) e Lauth
%                        autor(autor, lista de entries) correspondente

% com meta-predicados
inverte(Lbib, Lauth) :-
        is_list(Lbib),
        setof(autor(A,Le),
              setof(E,La^(member(bib(E,La),Lbib), member(A,La)),Le),
              Lauth).
```

Solução sem a utilização de meta-predicados:

```
inverte_alt(Lbib,Lauth) :-
        is_list(Lbib),
        inverte(Lbib, [], Lauth).

inverte([], Ac, Ac).
inverte([H|T], Ac, Lauth) :-
        insereEntrada(H, Ac, Ac1),
        inverte(T, Ac1, Lauth).

insereEntrada(bib(_,[]),Ac,Ac).
insereEntrada(bib(E,[A|T]),Ac,L):-
        append(L1,[autor(A,Le)|L2],Ac), !,
        append(L1,[autor(A,[E|Le])|L2],Ac1),
        insereEntrada(bib(E,T),Ac1,L).
insereEntrada(bib(E,[A|T]),Ac,L) :-
        insereOrd(A,[E], Ac, Ac1),
        insereEntrada(bib(E,T),Ac1,L).

insereOrd(A,Le,[],[autor(A,Le)]).
insereOrd(A,Le,[autor(H,Lh)|T],[autor(A,Le),autor(H,Lh)|T]) :-
        A@=<H, !.
insereOrd(A,Le,[autor(H,Lh)|T],[autor(H,Lh)|Taux]) :-
        insereOrd(A,Le,T,Taux).
```