Ficha Prática 8

8.1 Objectivos

1. Utilizar os meta-predicados `bagof`, `setof` e `findall`.

8.2 Conceitos

O Prolog possui três meta-predicados dedicados a encontrar todas as soluções para uma dada *query*:

- `findall`
- `bagof`
- `setof`

Considere a seguinte Base de Conhecimento:

```
% pais(Filho, Pai, Mae).
filho(manuel, mario, maria).
filho(teresa, mario, maria).
filho(joao, manuel, joana).
filho(lurdes, joao, joana).
filho(rui, manuel, teresa).
filho(alberto,manuel, teresa).
```

8.2.1 `findall`

A *query*:

```
?- findall(+T, +Objectivo, -L).
```

unifica `L` com a lista de todos os `T` que se obtêm ao procurar todas as soluções de `Objectivo` (`T` deverá conter variáveis que apareçam na expressão `Objectivo`).

Se não existirem instanciações para `T`, `L` unificará com lista vazia.

Considere que se pretende saber quem são os filhos do Manuel e da Teresa. Isso pode ser conseguido com as seguinte *query*:

```
?- findall(Filho,filho(Filho,manuel,teresa),L).

F = _G157
L = [rui, alberto]
```

Neste caso, o `findall` criou a lista `L` com todas as instanciações de `Filho` que satisfazem `filho(Filho,manuel,teresa)`.

Se existirem variáveis no objectivo que não estejam no padrão a lista conterá as instanciações do padrão para *qualquer valor dessas variáveis*. Por exemplo, a *query*

```
?- findall(F,filho(F,manuel, M),L).

F = _G157
M = _G159
L = [joao, rui, alberto]
```

coloca em `L` todas as instanciações de `F` para as quais também exista um `M` tal que se verifique `filho(F,manuel, M)`. Ou seja. calcula a lista de todos os filho de Manuel (independentemente de quem seja a mãe — note como a variável `M` não ficou instânciada).

8.2.2 `bagof`

A *query*:

```
?- bagof(+T, +Objectivo, -L).
```

unifica `L` com a lista de todos os `T` que se obtêm ao procurar todas as soluções de `Objectivo` (`T` deverá conter variáveis que apareçam na expressão `Objectivo`).

Se não existirem instanciações para `T`, o `bagof` falha.

Se existirem variáveis no objectivo que não estejam no padrão o `bagof` calculará uma lista para cada uma das instanciações possíveis para essas variáveis.

Por exemplo, a *query*

```
?- bagof(F,filho(F,manuel, M),L).

F = _G411
M = joana
L = [joao] ;

F = _G411
M = teresa
L = [rui, alberto] ;

No
```

calcula uma lista para `M=joana` (todos os filhos de Manuel e de Joana) e outra para `M=teresa` (todos os filho de Manuel e Teresa). Note que neste caso a variável M fica sempre unificada.

Considere agora a query:

```
?- bagof(F,filho(F,Pai, Mae),L).

F = _G157
Pai = mario
Mae = maria
L = [manuel, teresa] ;

F = _G157
Pai = manuel
Mae = joana
L = [joao] ;

F = _G157
Pai = joao
Mae = joana
L = [teresa] ;
```

```
F = _G157
Pai = manuel
Mae = teresa
L = [rui, alberto] ;

No
```

Neste caso obtemos listas de filhos para cada par de pais/mães.

Se na query anterior não pretendessemos que o Prolog considerasse as mães (se pretendessemos a lista de filhos de cada pai, independentemente das mães) utilizariamos a quantificação existêncial da variável `Mae`:

```
?- bagof(F,Maeˆfilho(F,Pai, Mae),L).

F = _G157
Mae = _G159
Pai = mario
L = [manuel, teresa] ;

F = _G157
Mae = _G159
Pai = joao
L = [teresa] ;

F = _G157
Mae = _G159
Pai = manuel
L = [joao, rui, alberto] ;

No
```

Neste caso as listas são calculadas com bases nas unificações de `F`, para qualquer valor de `Mae`.

Note que a expressao `findall(F, filho(F,Pai,Mae), L)` é equivalente à expressão `bagof(F, PaiˆMaeˆfilho(F,Pai, Mae), L)` excepto quando não exitam resultados a colocar em `L`. Nesse caso, a primeira expressão termina com `L` unificado com a lista vazia, e a segunda expressão falha.

8.2.3 `setof`

Semelhante a bagof, mas a lista é ordenada e sem duplicados:

```
setof(T, G, L) :-
        bagof(T, G, Laux),
        sort(Laux, L).
```

8.3 Exercícios

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`.
   (b) Escreva agora o predicado `alunos_de/2` que define a lista de alunos de uma cadeira[10].

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

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

(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).

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

(a) Escreva o predicado `mais_consultada/1` que define qual a página mais consultada[11].

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

(c) Escreva o predicado `com_mais_clintes/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.

[11] Lembre-se que o `setof` cria uma lista ordenada.