Paradigmas de Programação

Prof. Maicon R. Zatelli

Prolog - Programação Lógica
findall, bagof, setof

Universidade Federal de Santa Catarina
Florianópolis - Brasil
2018/1

Prolog

findall, **bagof** e **setof** são chamados de metapredicados.

Eles recebem uma meta como parâmetro e retornam uma lista de resultados.

Metapredicados são equivalentes a funções de alta ordem em linguagens funcionais.

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
```

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat].
```

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat].

%Retorna uma lista de genitores com idade maior ou igual a 50 anos.
?- findall(G, (genitor(G,_), idade(G,I), I >= 50), Genitores).
```

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat].

%Retorna uma lista de genitores com idade maior ou igual a 50 anos.
?- findall(G, (genitor(G,_), idade(G,I), I >= 50), Genitores).
Genitores = [pam, tom, tom].
```

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat].

%Retorna uma lista de genitores com idade maior ou igual a 50 anos.
?- findall(G, (genitor(G,_), idade(G,I), I >= 50), Genitores).
Genitores = [pam, tom, tom].

%Retorna uma lista de genitores ordenada e sem repetições.
?- findall(G, genitor(G,_), Genitores),
     sort(Genitores, GenitoresOrdenado).
```

- Note que o **sort** remove também os duplicados. **msort** é equivalente ao **sort**, mas não remove os duplicados.
  list_to_set remove os duplicados, mas não ordena.

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna uma lista de genitores.
?- findall(G, genitor(G,_), Genitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat].

%Retorna uma lista de genitores com idade maior ou igual a 50 anos.
?- findall(G, (genitor(G,_), idade(G,I), I >= 50), Genitores).
Genitores = [pam, tom, tom].

%Retorna uma lista de genitores ordenada e sem repetições.
?- findall(G, genitor(G,_), Genitores),
     sort(Genitores, GenitoresOrdenado).
Genitores = [pam, tom, tom, bob, bob, liz, pat],
GenitoresOrdenado = [bob, liz, pam, pat, tom].
```

- Note que o **sort** remove também os duplicados. **msort** é equivalente ao **sort**, mas não remove os duplicados. **list_to_set** remove os duplicados, mas não ordena.

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna a quantidade de genitores.
?- findall(G, genitor(G,_), Genitores),
     list_to_set(Genitores, GenitoresNaoDup),
     length(GenitoresNaoDup, QuantidadeGenitores).
```

Prolog

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna a quantidade de genitores.
?- findall(G, genitor(G,_), Genitores),
     list_to_set(Genitores, GenitoresNaoDup),
     length(GenitoresNaoDup, QuantidadeGenitores).
Genitores = [pam, tom, tom, bob, bob, liz, pat],
GenitoresNaoDup = [pam, tom, bob, liz, pat],
QuantidadeGenitores = 5.
```

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna a lista de pessoas e idades que não possuem filhos.
?- findall(genitor(Pessoa, Idade),
           (  genitor(_,Pessoa),
              not(genitor(Pessoa,_)),
              idade(Pessoa, Idade)
           ),
           Pessoas).
```

findall(X,G,L) retorna em **L** uma lista de objetos **X** para os quais a meta **G** é satisfeita.

```
%Retorna a lista de pessoas e idades que não possuem filhos.
?- findall(genitor(Pessoa, Idade),
           (  genitor(_,Pessoa),
              not(genitor(Pessoa,_)),
              idade(Pessoa, Idade)
           ),
           Pessoas).
Pessoas = [genitor(ana, 12), genitor(bill, 15), genitor(jim, 1)].
```

Prolog

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob] ;
Genitor = pat,
Filhos = [jim]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos.
?- bagof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob] ;
Genitor = pat,
Filhos = [jim] ;
Genitor = tom,
Filhos = [bob, liz].
```

Prolog

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

O operador $\wedge$ permite ignorar variáveis. Assim, abaixo listamos todos os filhos, ignorando quem é o genitor.

```
?- bagof(Filho, Genitor ^ genitor(Genitor,Filho), Filhos).
```

Prolog

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

O operador $\wedge$ permite ignorar variáveis. Assim, abaixo listamos todos os filhos, ignorando quem é o genitor.

```
?- bagof(Filho, Genitor ^ genitor(Genitor,Filho), Filhos).
Filhos = [bob, bob, liz, ana, pat, bill, jim].
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob] ;
Filhos = [jim]
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- bagof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob] ;
Filhos = [jim] ;
Filhos = [bob, liz].
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Retorna a lista de filhos com genitores com idade de 50 anos ou mais.
?- bagof(Filho,
          Idade ^ Genitor ^
            (genitor(Genitor,Filho), idade(Genitor,Idade), Idade >= 50),
          Filhos).
```

bagof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G.

```
%Retorna a lista de filhos com genitores com idade de 50 anos ou mais.
?- bagof(Filho,
           Idade ^ Genitor ^
             (genitor(Genitor,Filho), idade(Genitor,Idade), Idade >= 50),
           Filhos).
Filhos = [bob, bob, liz].
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat]
```

Prolog

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob] ;
Genitor = pat,
Filhos = [jim]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos.
?- setof(Filho, genitor(Genitor,Filho), Filhos).
Genitor = bob,
Filhos = [ana, pat] ;
Genitor = liz,
Filhos = [bill] ;
Genitor = pam,
Filhos = [bob] ;
Genitor = pat,
Filhos = [jim] ;
Genitor = tom,
Filhos = [bob, liz].
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

O operador $\wedge$ permite ignorar variáveis. Assim, abaixo listamos todos os filhos, ignorando quem é o genitor.

```
?- setof(Filho, Genitor ^ genitor(Genitor,Filho), Filhos).
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

O operador $\wedge$ permite ignorar variáveis. Assim, abaixo listamos todos os filhos, ignorando quem é o genitor.

```
?- setof(Filho, Genitor ^ genitor(Genitor,Filho), Filhos).
Filhos = [ana, bill, bob, jim, liz, pat].
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob] ;
Filhos = [jim]
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Para cada genitor, retorna uma lista de filhos, mesmo utilizando _
?- setof(Filho, genitor(_,Filho), Filhos).
Filhos = [ana, pat] ;
Filhos = [bill] ;
Filhos = [bob] ;
Filhos = [jim] ;
Filhos = [bob, liz].
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Retorna a lista de filhos com genitores com idade de 50 anos ou mais.
?- setof(Filho,
         Idade ^ Genitor ^
           (genitor(Genitor,Filho), idade(Genitor,Idade), Idade >= 50),
         Filhos).
```

setof(X,G,L) retorna uma lista L de objetos X para cada instância das variáveis na meta G, mas sem repetições e de maneira ordenada.

```
%Retorna a lista de filhos com genitores com idade de 50 anos ou mais.
?- setof(Filho,
         Idade ^ Genitor ^
           (genitor(Genitor,Filho), idade(Genitor,Idade), Idade >= 50),
         Filhos).
Filhos = [bob, liz].
```

Prolog - Alguns Links Úteis

- http://www.swi-prolog.org/pldoc/doc_for?object=findall/3
- http://www.swi-prolog.org/pldoc/doc_for?object=setof/3
- http://www.swi-prolog.org/pldoc/doc_for?object=bagof/3

Prolog

Ver atividade no Moodle