

Paradigmas de Programação

Prof. Maicon R. Zatelli

Prolog - Programação Lógica
findall, bagof, setof

Universidade Federal de Santa Catarina
Florianópolis - Brasil

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.

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).
```

```
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.
```

Prolog

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(pessoa(Pessoa, Idade),  
    ( genitor(_,Pessoa),  
      not(genitor(Pessoa,_)),  
      idade(Pessoa, Idade)  
    ),  
    Pessoas).
```

Prolog

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(pessoa(Pessoa, Idade),  
    ( genitor(_,Pessoa),  
      not(genitor(Pessoa,_)),  
      idade(Pessoa, Idade)  
    ),  
    Pessoas).
```

```
Pessoas = [pessoa(ana, 12), pessoa(bill, 15), pessoa(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).
```

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).  
Genitor = bob,  
Filhos = [ana, pat]
```

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).  
Genitor = bob,  
Filhos = [ana, pat] ;  
Genitor = liz,  
Filhos = [bill]
```

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).  
Genitor = bob,  
Filhos = [ana, pat] ;  
Genitor = liz,  
Filhos = [bill] ;  
Genitor = pam,  
Filhos = [bob]
```


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).  
Genitor = bob,  
Filhos = [ana, pat] ;  
Genitor = liz,  
Filhos = [bill] ;  
Genitor = pam,  
Filhos = [bob] ;  
Genitor = pat,  
Filhos = [jim]
```

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).  
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  $\wedge$  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].
```

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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).
```

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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat]
```

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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill]
```

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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob]
```


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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob] ;  
Filhos = [jim]
```

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, mesmo utilizando _  
?- bagof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob] ;  
Filhos = [jim] ;  
Filhos = [bob, liz].
```

Prolog

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).
```

Prolog

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].
```

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).
```

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]
```

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]
```

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] ;  
Genitor = pam,  
Filhos = [bob]
```


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] ;  
Genitor = pam,  
Filhos = [bob] ;  
Genitor = pat,  
Filhos = [jim]
```

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] ;  
Genitor = pam,  
Filhos = [bob] ;  
Genitor = pat,  
Filhos = [jim] ;  
Genitor = tom,  
Filhos = [bob, liz].
```

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.

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

```
?- setof(Filho, Genitor  $\wedge$  genitor(Genitor,Filho), Filhos).
```

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.

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].
```

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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).
```

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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).  
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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill]
```

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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob]
```


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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob] ;  
Filhos = [jim]
```

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, mesmo utilizando _  
?- setof(Filho, genitor(_,Filho), Filhos).  
Filhos = [ana, pat] ;  
Filhos = [bill] ;  
Filhos = [bob] ;  
Filhos = [jim] ;  
Filhos = [bob, liz].
```

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.

```
%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).
```

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.

```
%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