PROLOG

Rafael D. Ribeiro, M.Sc.
rafaeldiasribeiro@gmail.com
http://www.rafaeldiasribeiro.com.br

Paradigmas de Programação

- **Paradigma imperativo (paradigma *procedural*)**

"Primeiro faça isso e depois faça aquilo."

O problema é analisado até que se encontre uma solução. Basicamente, é uma sequência de comandos que o computador executará, passo-a-passo, modificando dados e variáveis a fim de chegar ao resultado esperado.

Exemplo de linguagens que utilizam este paradigma: Basic, C , Pascal.

Paradigmas de Programação

- **Paradigma imperativo (paradigma procedural)**
  Ex:

```c
int InsertionSort(int *vetor, int tam){
        int i, j, elemento;

        for (i=1; i<tam; i++){
                elemento=vetor[i];
                j=i-1;
                while ((j >= 0) && (vetor[j] > elemento)){
                        vetor[j+1] = vetor[j];
                        j--;
                }
                vetor[j+1] = elemento;
        }

        return 0;
}
```

Exemplo em C do algoritmo de ordenação por inserção

Paradigmas de Programação

- **Paradigma funcional**

  " Subdividir o problema em outras funções e resolver cada uma separadamente, pois os resultados encontrados serão utilizados posteriormente."

  - Sobre o paradigma funcional, é fácil ilustrar através de um fluxograma. Cada bloco recebe no topo uma entrada de dados e retorna, na base, os dados de saída. A solução geral é dividida em várias funções que, no final, se associam para mostrar o resultado na tela.

Paradigmas de Programação

- **Paradigma funcional**

Ex:

```haskell
pertence :: Eq t => t -> [t] -> Bool
pertence x (c:r)
    | x == c = True
    | otherwise = pertence x r
pertence x [] = False
```

exemplo na linguagem Haskell

NOTA: Vários autores definem Programação Funcional dentro de Programação Descritiva.

Paradigmas de Programação

- **Paradigma orientado a objetos**

  " Um conjunto de classes faz a interação entre objetos (instâncias) e, com a troca de mensagens entre eles, forma-se o software como um todo."

- Praticamente tudo é objeto, cada qual com estrutura e comportamento próprios. Esses objetos são classificados em classes e comunicam entre si.
- Cada uma dessas representa um determinado fenômeno e seus objetos são organizados hierarquicamente. O conjunto de classes faz a interação entre objetos e a troca de mensagens entre eles forma o software como um todo.

Paradigmas de Programação

- **Paradigma orientado a objetos**

Ex:

```java
public class Caminhao extends Veiculo {

    private double capacidade;
    private int numEixos;

    public Caminhao(double capacidade, int numEixos) {
        this.capacidade = capacidade;
        this.numEixos = numEixos;
    }

    public double calcularIPVA() {
        return 0.05 * this.getValor();
    }
}
```

Exemplo de código orientado a objetos em Java

Paradigmas de Programação

- **Paradigma declarativo (Paradigma Imperativo)**

"Qual é o problema?"

- O paradigma declarativo caracteriza-se pelo método preciso de descrever um problema, sem se preocupar com qual algoritmo será utilizado para resolvê-lo.
- Baseia-se em cálculo de predicados. Não possui códigos para manipular a memória ou realizar desvios condicionais
- **Prolog** é uma linguagem lógica que baseia-se neste paradigma.

Paradigmas de Programação

- **Paradigma declarativo (Paradigma Imperativo)**
  - Este paradigma é mais adequado para solucionar problemas onde é necessário representar algum tipo de conhecimento .
  - Ex: Aplicações que realizem computação simbólica, compreensão de linguagem natural ou em sistemas especialistas.

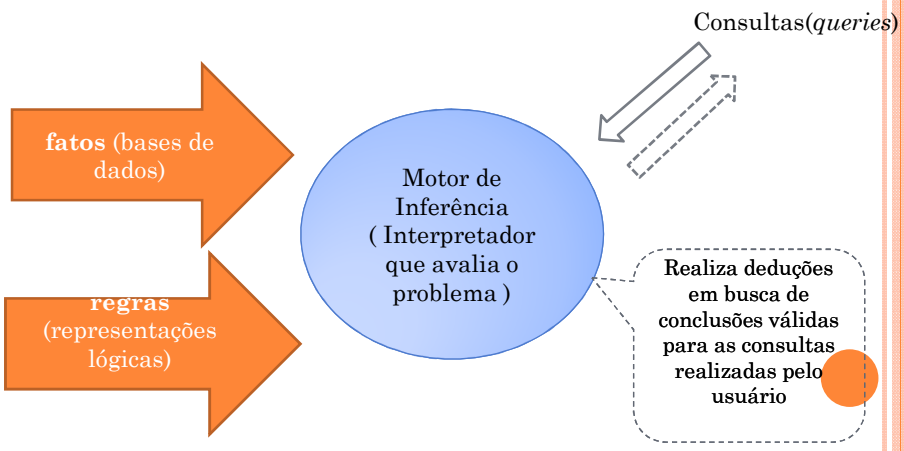
Paradigmas de Programação

- **Características do PROLOG**
  - É uma linguagem declarativa, ou seja, ao invés de o programa estipular a maneira de chegar à solução passo-a-passo, como acontece nas linguagens procedimentais ou orientadas a objeto, ele fornece uma descrição do problema que se pretende computar utilizando uma coleção de fatos e regras (lógica) que indicam como deve ser resolvido o problema proposto
  - Não possui estruturas de controle (if-else, do-while, for, switch) presentes na maioria das linguagens de programação. Para isso utiliza-se métodos lógicos para declarar como o programa deverá atingir o seu objetivo.

    Um programa em Prolog pode rodar em um modo interativo, o usuário poderá formular queries utilizando os **fatos** (bases de dados) e as **regras** (representações lógicas) para produzir a solução.

Paradigmas de Programação

- **Funcionamento do PROLOG**



- Os **fatos** de Prolog permitem a definição de predicados por meio da declaração de quais itens pertencentes ao universo (ou domínio) satisfazem os predicados.

predicado(arg 1[,arg 2,...,arg n]).

onde:

predicado = relação

arg i = objetos sobre os quais atuam a relação.

- Exemplo de fato com um argumento
  - homem (x).
    Define quais elementos do universo possuem tal predicado " x é homem "
- Exemplo de fato com dois argumentos seria a relação amiga
  - amiga(ana, paula).
    Define uma relação (amizade) entre dois argumentos (ana,paula)

Para manter uma coerência, é importante que todos os fatos sigam uma mesma estrutura de interpretação.

É importante observar que um conjunto de fatos forma um banco de dados. Ou seja, várias afirmações compõem os dados existentes no sistema. Os fatos são a célula básica do banco de dados Prolog.

É responsabilidade do programador manter a definição de um predicado consistente. Por exemplo, poderia ser criado o predicado come(x,y) para representar que x come y ou y come x, o programador é que deverá especificar os fatos de maneira consistente.

Em Prolog se fornece fatos e regras para uma base de dados, então se executam consultas (*queries*) a essa base de dados.

A unidade básica do Prolog é o predicado, que é postulado verdadeiro. Um predicado consiste de uma cabeça e um número de argumentos.

Por exemplo: homem(antonio).

Isso informa à base de dados o fato que 'antonio' é um 'homem'.

Formalmente, 'antonio' é a cabeça e 'homem' é o único argumento do predicado.

Alguns exemplos de consultas que podem ser feitas ao interpretador Prolog baseado nesse fato:

Antonio é um homem?

?- homem(antonio).

yes.

que coisas (conhecidas) são homem?

?- homem(X).

X = antonio;

yes.

Predicados são normalmente definidos para expressar algum fato sobre o mundo que o programa deve conhecer. Na maioria dos casos, o uso de predicados requer uma certa convenção. Por exemplo, qual das duas versões abaixo significaria que José é o pai de Ana?

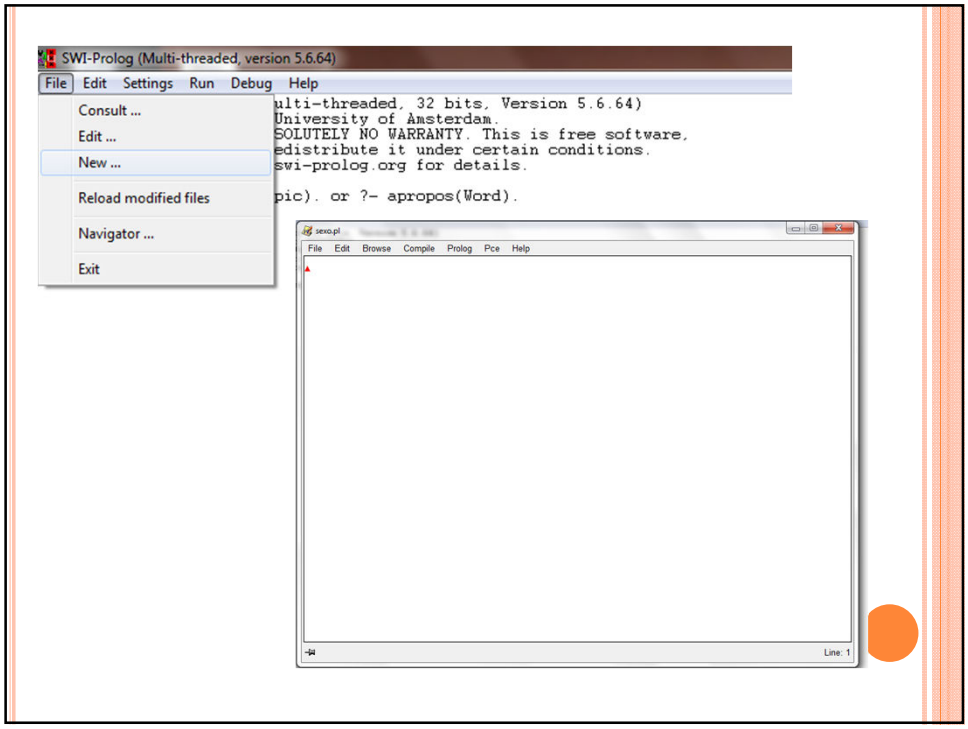
- pai(ana,jose).
- pai(jose,ana).

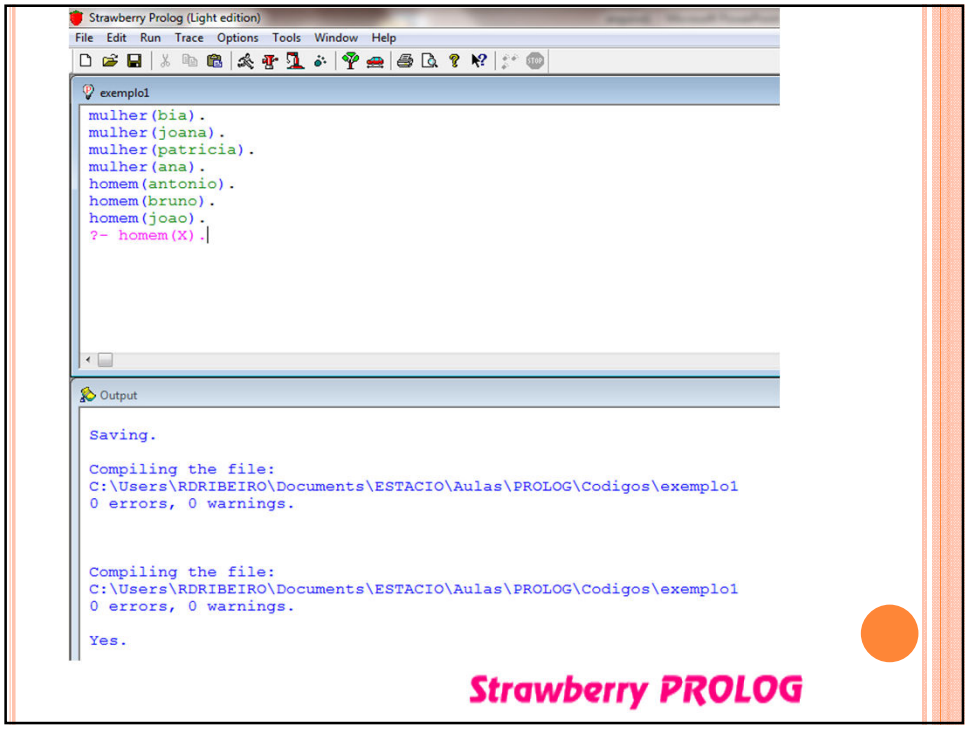
Em ambos os casos 'pai' é a cabeça e 'ana' e 'jose' são argumentos. Entretanto, no primeiro caso Ana vem primeiro na lista de argumentos, e no segundo, quem vem primeiro é José (a ordem nos argumentos é importante).

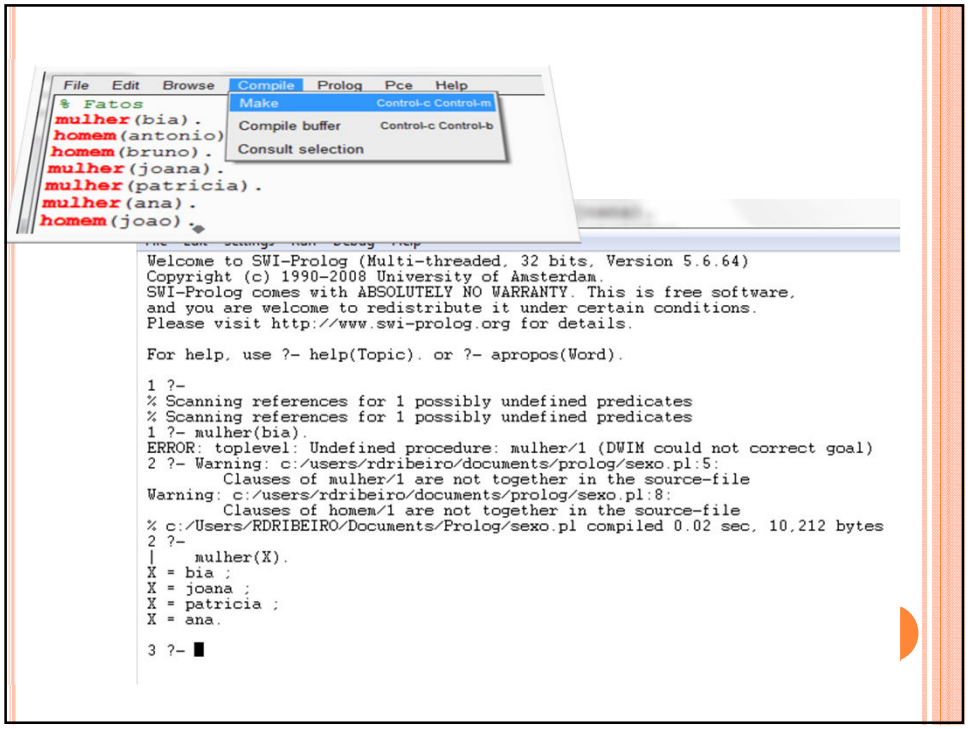
O primeiro caso é um exemplo de definição na ordem **Verbo Sujeito Objeto**, e o segundo, na ordem **Verbo Objeto Sujeito**.

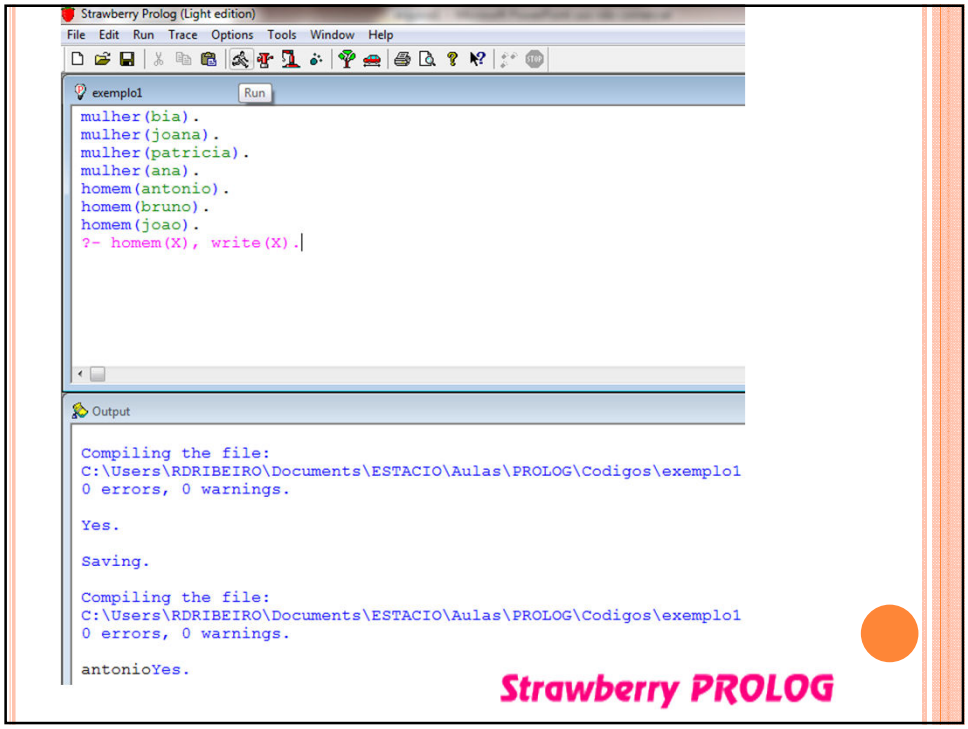
Como Prolog não entende português, ambas as versões estão corretas de acordo com seu escopo; no entanto é uma boa prática de programação escolher uma única convenção para ser usada no mesmo programa, para evitar escrever algo como:

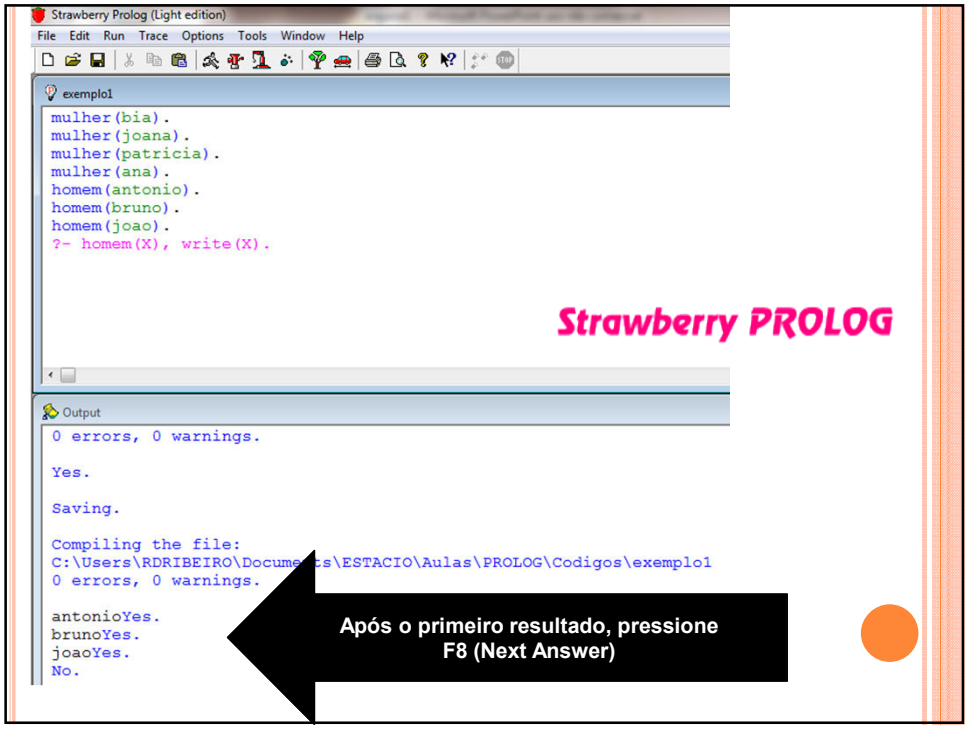
- pai(jose,ana).
- pai(maria,joao).

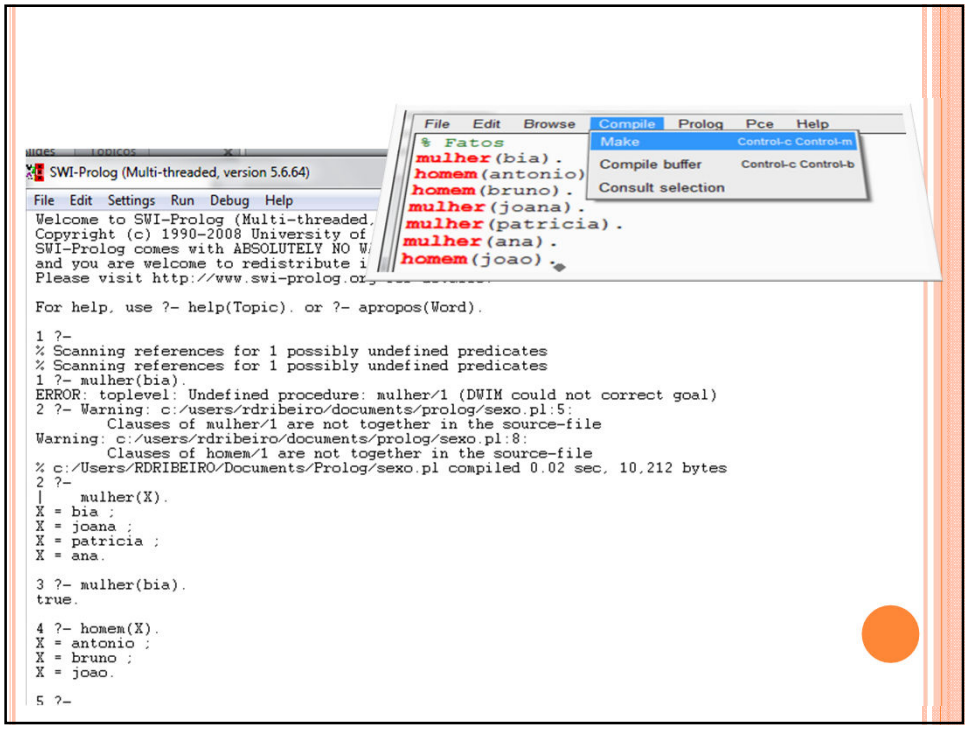

SWI-Prolog (Multi-threaded, version 5.6.64)
File Edit Settings Run Debug Help
Consult ...
Edit ...
New ...
Reload modified files
Navigator ...
Exit
ulti-threaded, 32 bits, Version 5.6.64)
University of Amsterdam.
SOLUTELY NO WARRANTY. This is free software,
edistribute it under certain conditions.
swi-prolog.org for details.
pic). or ?- apropos(Word).
sexo.pl
File Edit Browse Compile Prolog Pce Help
Line: 1


Strawberry Prolog (Light edition)
File Edit Run Trace Options Tools Window Help
exemplo1
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
?- homem(X).
Output
Saving.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo1
0 errors, 0 warnings.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo1
0 errors, 0 warnings.
Yes.
Strawberry PROLOG

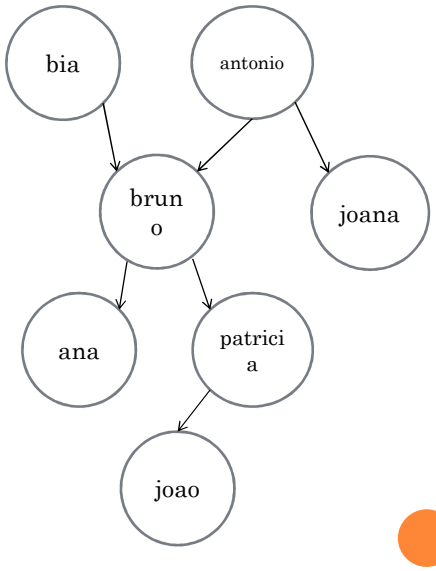

File Edit Browse Compile Prolog Pce Help
Make Control-c Control-m
Compile buffer Control-c Control-b
Consult selection
% Fatos
mulher(bia).
homem(antonio).
homem(bruno).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(joao).
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.64)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).
1 ?-
% Scanning references for 1 possibly undefined predicates
% Scanning references for 1 possibly undefined predicates
1 ?- mulher(bia).
ERROR: toplevel: Undefined procedure: mulher/1 (DWIM could not correct goal)
2 ?- Warning: c:/users/rdribeiro/documents/prolog/sexo.pl:5:
Clauses of mulher/1 are not together in the source-file
Warning: c:/users/rdribeiro/documents/prolog/sexo.pl:8:
Clauses of homem/1 are not together in the source-file
% c:/Users/RDRIBEIRO/Documents/Prolog/sexo.pl compiled 0.02 sec, 10,212 bytes
2 ?-
| mulher(X).
X = bia ;
X = joana ;
X = patricia ;
X = ana.
3 ?-


Strawberry Prolog (Light edition)
File Edit Run Trace Options Tools Window Help
exemplo1
Run
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
?- homem(X), write(X).
Output
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo1
0 errors, 0 warnings.
Yes.
Saving.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo1
0 errors, 0 warnings.
antonioYes.
Strawberry PROLOG


Strawberry Prolog (Light edition)
File Edit Run Trace Options Tools Window Help
exemplo1
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
?- homem(X), write(X).
Strawberry PROLOG
Output
0 errors, 0 warnings.
Yes.
Saving.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo1
0 errors, 0 warnings.
antonioYes.
brunoYes.
joaoYes.
No.
Após o primeiro resultado, pressione
F8 (Next Answer)

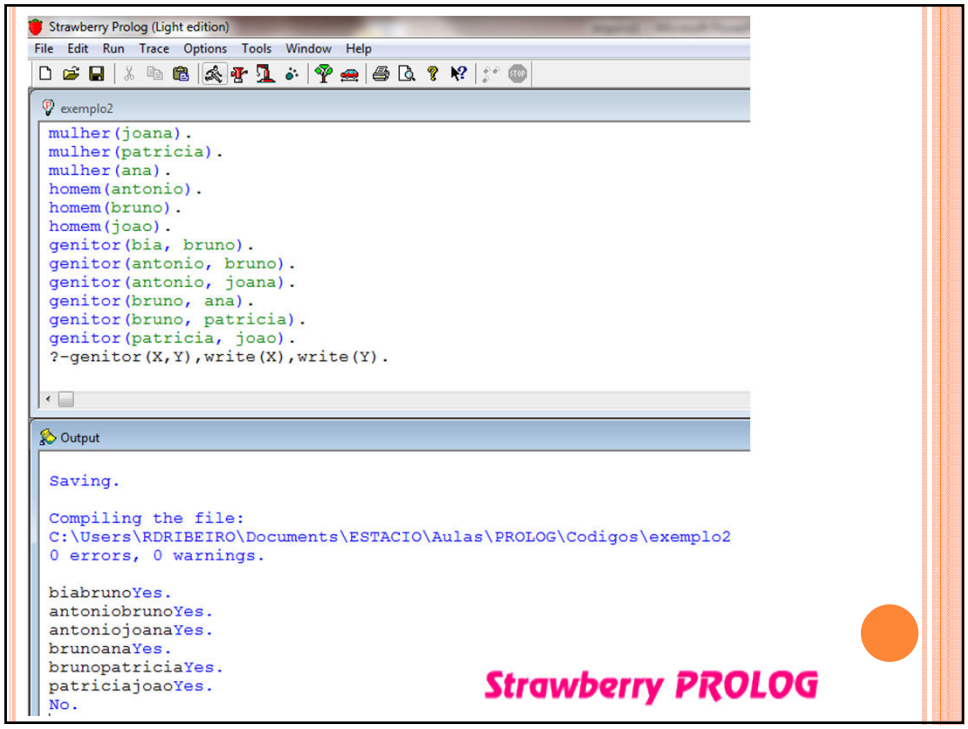

SWI-Prolog (Multi-threaded, version 5.6.64)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded,
Copyright (c) 1990-2008 University of
SWI-Prolog comes with ABSOLUTELY NO W
and you are welcome to redistribute i
Please visit http://www.swi-prolog.or
File Edit Browse Compile Prolog Pce Help
Make Control-c Control-m
Compile buffer Control-c Control-b
Consult selection
% Fatos
mulher(bia).
homem(antonio).
homem(bruno).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(joao).
For help, use ?- help(Topic). or ?- apropos(Word).
1 ?-
% Scanning references for 1 possibly undefined predicates
% Scanning references for 1 possibly undefined predicates
1 ?- mulher(bia).
ERROR: toplevel: Undefined procedure: mulher/1 (DWIM could not correct goal)
2 ?- Warning: c:/users/rdribeiro/documents/prolog/sexo.pl:5:
Clauses of mulher/1 are not together in the source-file
Warning: c:/users/rdribeiro/documents/prolog/sexo.pl:8:
Clauses of homem/1 are not together in the source-file
% c:/Users/RDRIBEIRO/Documents/Prolog/sexo.pl compiled 0.02 sec, 10,212 bytes
2 ?-
| mulher(X).
X = bia ;
X = joana ;
X = patricia ;
X = ana.
3 ?- mulher(bia).
true.
4 ?- homem(X).
X = antonio ;
X = bruno ;
X = joao.
5 ?-

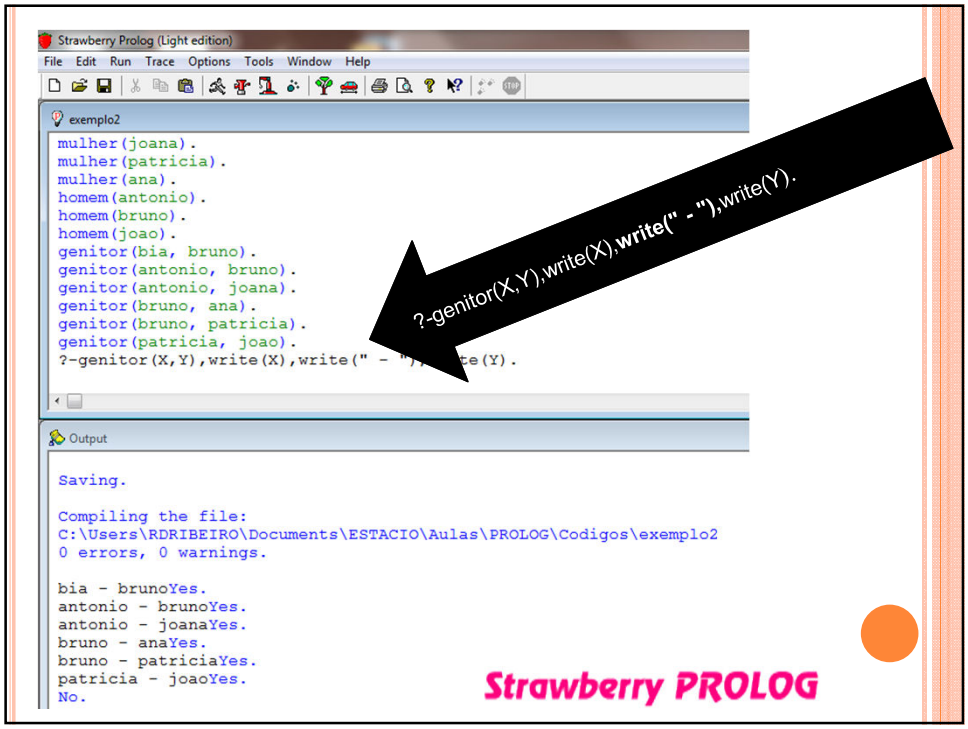
```
% fatos – arvore genealógica
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
```



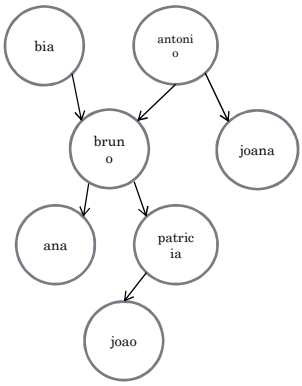

```
8 ?- genitor(antonio,bruno).
true .

9 ?- genitor(bruno,antonio).
false.

10 ?- genitor(X,Y).
X = bia,
Y = bruno ;
X = antonio,
Y = bruno ;
X = antonio,
Y = joana ;
X = bruno,
Y = ana ;
X = bruno,
Y = patricia ;
X = patricia,
Y = joao.

11 ?-
```

```
%fatos
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
```


Strawberry Prolog (Light edition)
File Edit Run Trace Options Tools Window Help
exemplo2
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
?-genitor(X,Y),write(X),write(Y).
Output
Saving.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo2
0 errors, 0 warnings.
biabrunoYes.
antoniobrunoYes.
antoniojoanaYes.
brunoanaYes.
brunopatriciaYes.
patriciajoaoYes.
No.
Strawberry PROLOG


Strawberry Prolog (Light edition)
File Edit Run Trace Options Tools Window Help
exemplo2
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
?-genitor(X,Y),write(X),write(" - "),write(Y).
?-genitor(X,Y),write(X),write(" - "),write(Y).
Output
Saving.
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo2
0 errors, 0 warnings.
bia - brunoYes.
antonio - brunoYes.
antonio - joanaYes.
bruno - anaYes.
bruno - patriciaYes.
patricia - joaoYes.
No.
Strawberry PROLOG

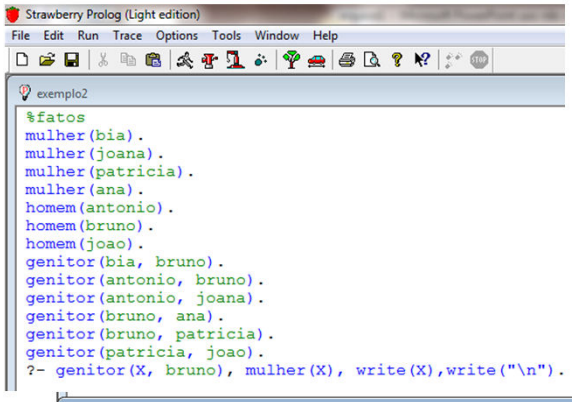
Quem é a mãe de bruno ?
?- genitor(X, bruno), mulher(X).




```
%fatos
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
```

```
5 ?-
|    genitor(X, bruno), mulher(X).
X = bia ;
false.
```

Quem é a mãe de bruno ?
?- genitor(X, bruno), mulher(X).



Strawberry PROLOG

Quem é a pai de bruno ?
?- genitor(X, bruno), not(mulher(X)).

```
6 ?- genitor(X, bruno), not(mulher(X)).
X = antonio ;
false.
```

```
%fatos
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
```

Quem é a pai de bruno ?
?- genitor(X, bruno), homem(X).

```
7 ?- genitor(X, bruno), homem(X).
X = antonio ;
false.
```

Quem é a pai de bruno ?

```
%fatos
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
?- genitor(X, bruno), not(mulher(X)), write(X),write("\n").
```

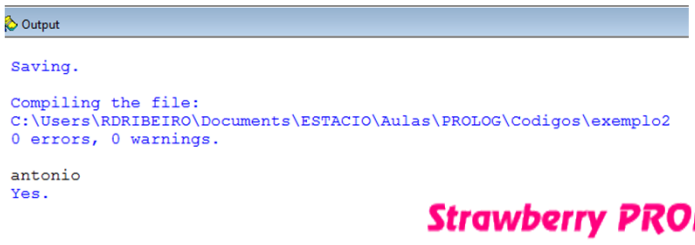
Output

```
Saving.

Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo2
0 errors, 0 warnings.

antonio
Yes.
```

Strawberry PROLOG

Quem é a pai de bruno ?

```
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
?-genitor(X, bruno), homem(X), write(X),write("\n").
```



Regras

Uma regra em Prolog é uma relação que utiliza outras mais elementares por meio do conectivo lógico "se...então..." (implicação lógica).

A conclusão é verdadeira se as suas condições também forem verdadeiras.

A verdade do fato representado por uma regra dependerá de suas condições.

Estrutura:

Conclusão :- Condição

Regras

Exemplo:

relação prole(y,x), signicando que y é prole de x.

Esta relação é a relação inversa de genitor(x,y), assim, pode-se armar que y é prole de x se x é genitor de y.

```
prole(Y,X) :- genitor(X,Y).
```


```
prole(Y,X) :- genitor(X,Y).
```


o símbolo :- pode ser lido como se.

A parte da regra a esquerda do símbolo :- é denominada de conclusão (ou cabeça), já a parte a direita deste é chamada de condição (ou corpo).

Assim, para responder a consulta o interpretador Prolog precisa satisfazer parte condicional da regra, uma vez que não existem fatos relacionados a prole, para então obter uma conclusão. Para isso, são utilizadas substituições, até que se satisfaça a parte condicional ou não existam mais possibilidades de substituição.

o símbolo **:-** pode ser lido como **se**.

A parte da regra a esquerda do símbolo :- é denominada de conclusão (ou cabeça), já a parte a direita deste é chamada de condição (ou corpo).

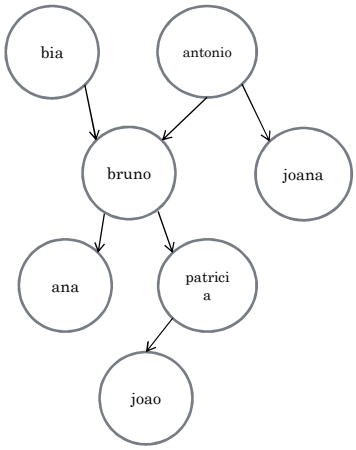
Assim, para responder a consulta o interpretador Prolog precisa satisfazer parte condicional da regra, uma vez que não existem fatos relacionados a prole, para então obter uma conclusão. Para isso, são utilizadas substituições, até que se satisfaça a parte condicional ou não existam mais possibilidades de substituição.

o símbolo **:-** pode ser lido como **se**.

A parte da regra a esquerda do símbolo :- é denominada de conclusão (ou cabeça), já a parte a direita deste é chamada de condição (ou corpo).

Assim, para responder a consulta o interpretador Prolog precisa satisfazer parte condicional da regra, uma vez que não existem fatos relacionados a prole, para então obter uma conclusão. Para isso, são utilizadas substituições, até que se satisfaça a parte condicional ou não existam mais possibilidades de substituição.

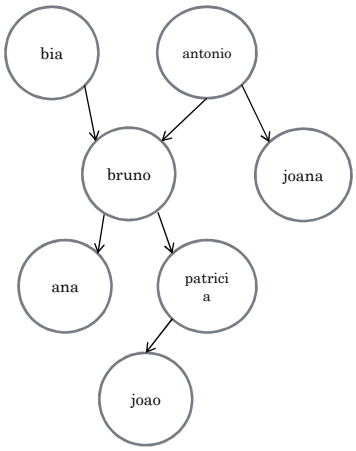
```
%fatos
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%regra
prole(Y,X) :- genitor(X,Y).
```




```
16 ?- prole(X,antonio).
X = bruno ;
X = joana.
```

prole(Y,antonio)

```
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).

?-prole(Y,antonio), write(Y).
```




```
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\exemplo2
0 errors, 0 warnings.

brunoYes.
joanaYes.
No.
```

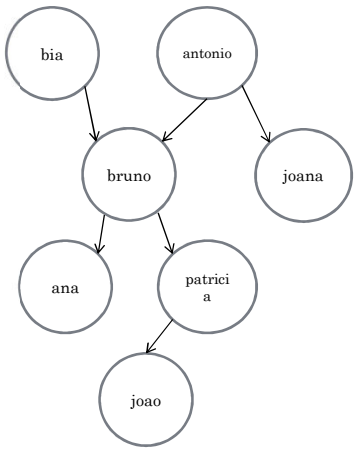
Strawberry PROLOG

Pode-se definir as relações mae(x,y) e avos(x,y).

Regras:

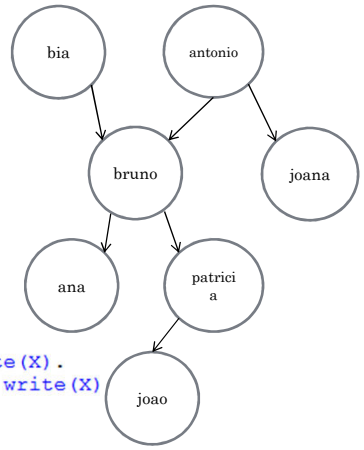
mae(X,Y) :- g e n i t o r (X,Y) , mulher (X) .
avos (X, Z) :- g e n i t o r (X,Y) , g e n i t o r (Y, Z) .

```
%fatos
mulher(bia).
homem(antonio).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%regra
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).
```




```
17 ?- mae(X,joao).
X = patricia.

18 ?- avos(X,joao).
X = bruno ;
false.

19 ?- avos(X,patricia).
X = bia ;
X = antonio ;
false.
```

```
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).

%Consultas
?-mae(Y,joao), write(Y).
?-write("--------- \n"),avos(X,joao), write(X).
?-write("--------- \n"),avos(X,patricia), write(X)
```



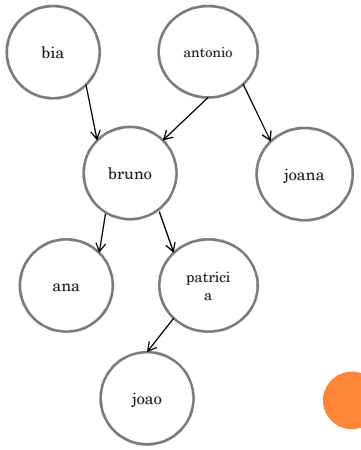

```
patriciaYes.
---------
brunoYes.
---------
biaYes.
antonioYes.
No.
```

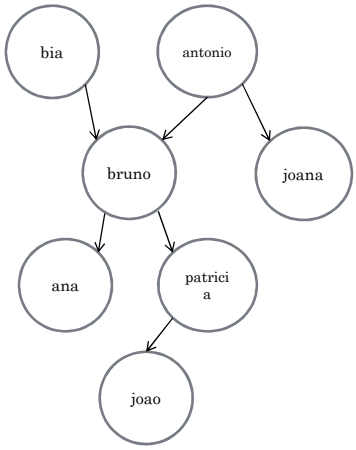
Strawberry PROLOG

Relação irma(x,y), signicando x é irmã de y.

Note que deve-se ter cuidado na definição deste relacionamento, pois, pode-se definir este através da seguinte regra:

```
irma (ana,patricia) :-
g e n i t o r (bruno ,ana) ,
g e n i t o r (bruno ,patricia) ,
mulher (X) .
```

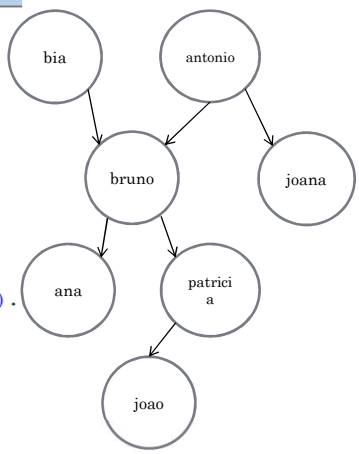




irma (X,Y) :- g e n i t o r (Z ,X) , g e n i t o r (Z ,Y) , mulher (X) .

exemplo2*

```
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).
irma(X,Y):-genitor(Z,X),genitor(Z,Y),mulher(X).

%Consultas
?-irma(Y,X), write(Y),write(X).
```




```
joanabrunoYes.
joanajoanaYes.
anaanaYes.
anapatriciaYes.
```

Strawberry PROLOG

exemplo2

Strawberry PROLOG

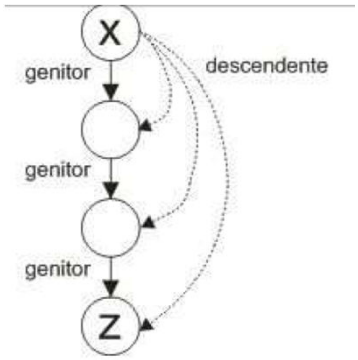
```
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).
irma(X,Y):-genitor(Z,X),genitor(Z,Y),mulher(X),not(X=Y)
```

```
joanabrunoYes.
anapatriciaYes.
patriciaanaYes.
No.
```

A recursão é um dos elementos mais importantes da linguagem Prolog, este conceito permite a resolução de problemas significativamente complexos de maneira relativamente simples.

Relação descendente(x,y), significando que y é um descendente de x.

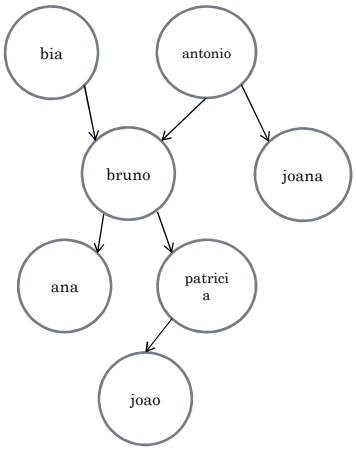
```
descendente(X,Z) :- genitor(X,Z).
```

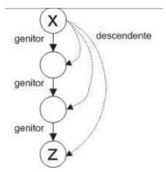


```
%fatos
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).
irma(X,Y):-genitor(Z,X),genitor(Z,Y),mulher(X),not(X=Y).
descendente(X,Z):-genitor(X,Z).

%Consultas
?-descendente(X,Z), write(X),write(Z).
```

```
Compiling the file:
C:\Users\RDRIBEIRO\Doc
0 errors, 0 warnings.

biabrunoYes.
antoniobrunoYes.
antoniojoanaYes.
brunoanaYes.
brunopatriciaYes.
patriciajoaoYes.
No.
```

Strawberry PROLOG

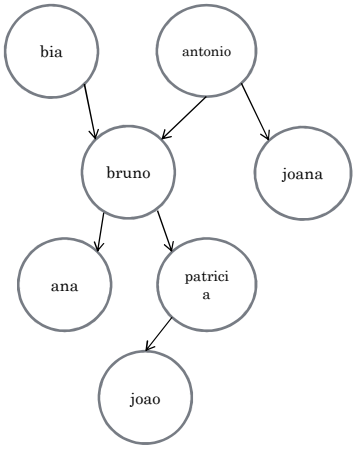


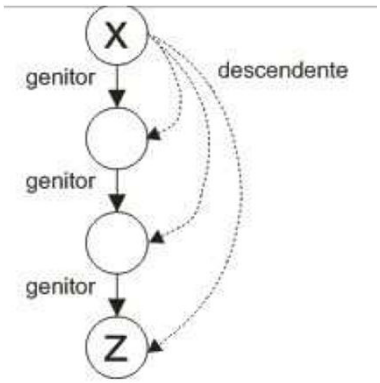


Strawberry PROLOG

```
Compiling the file:
C:\Users\RDRIBEIRO\Doc
0 errors, 0 warnings.

biabrunoYes.
antoniobrunoYes.
antoniojoanaYes.
brunoanaYes.
brunopatriciaYes.
patriciajoaoYes.
No.
```

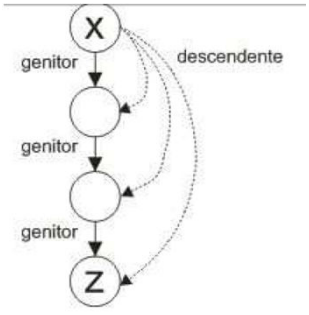
Apenas descendência direta !





Como definir regra para descendência indireta direta ?

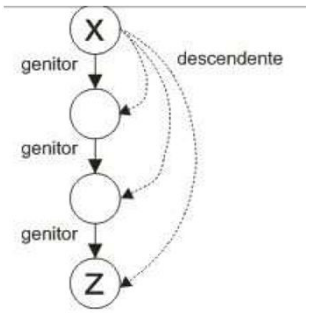
Como definir regra para descendência indireta direta ?




```
descendente(X,Z) :- genitor(X,Y), genitor(Y,Z).
descendente(X,Z) :- genitor(X,Y), genitor(Y,W), genitor(W,Z).
...
```

Solução trabalhosa e limitada !

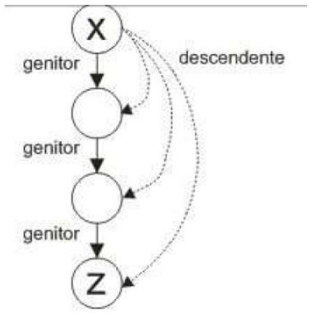
Como definir regra para descendência indireta direta ?



É necessário definir a seguinte afirmação:

“ x é um descendente de z se existe um y, tal que, x seja genitor de y e y seja um descendente de z “

“ x é um descendente de z se existe um y, tal que, x seja genitor de y e y seja um descendente de z “



descendente(X,Z):-genitor(X,Y), descendente(Y,Z)

As duas regras são necessárias, pois o uso somente da primeira regra só seria suficiente para casos de descendência direta (equivalente à relação genitor), enquanto o uso exclusivo da segunda regra levaria a uma busca infinita de descendência.

descendente(X,Z):-genitor(X,Z).

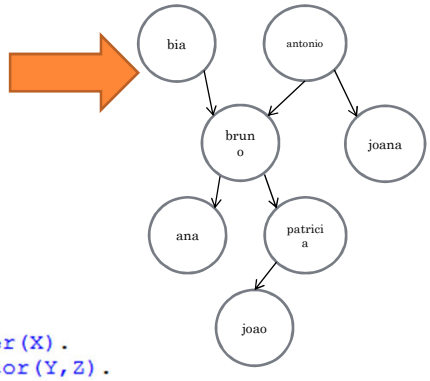
descendente(X,Z):-genitor(X,Y), descendente(Y,Z).

- Um conjunto de regras utilizados para descrever uma única relação é, em geral, chamada de procedimento (procedure). Assim, as regras relacionadas à descendência, ilustradas no exemplo, podem ser denominadas de procedimento.

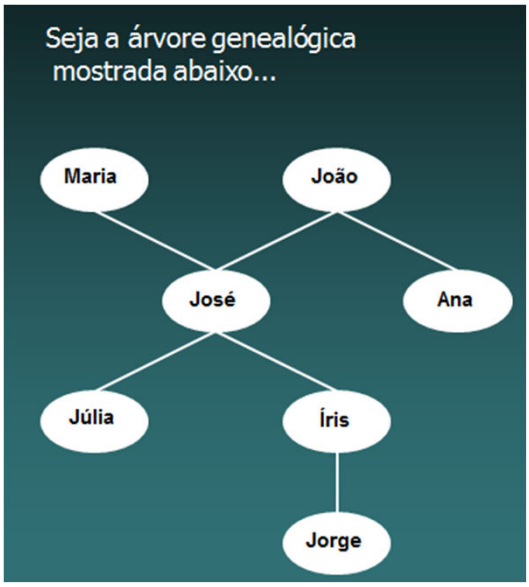
- Um conjunto de regras utilizados para descrever uma única relação é, em geral, chamada de procedimento (procedure). Assim, as regras relacionadas à descendência, ilustradas no exemplo, podem ser denominadas de procedimento.

```
%fatos
mulher(bia).
mulher(joana).
mulher(patricia).
mulher(ana).
homem(antonio).
homem(bruno).
homem(joao).
genitor(bia, bruno).
genitor(antonio, bruno).
genitor(antonio, joana).
genitor(bruno, ana).
genitor(bruno, patricia).
genitor(patricia, joao).
%Regras
prole(Y,X) :- genitor(X,Y).
mae(X,Y) :- genitor(X,Y), mulher(X).
avos(X,Z) :- genitor(X,Y),genitor(Y,Z).
irma(X,Y):-genitor(Z,X),genitor(Z,Y),mulher(X),not(X=Y).
descendente(X,Z):-genitor(X,Z).
descendente(X,Z):-genitor(X,Y),descendente(Y,Z).

%Consultas
?-descendente(bia,Z),write(Z).
```

Strawberry PROLOG




```
brunoYes.
anaYes.
patriciaYes.
joaoYes.
No.
```



Pode ser representada pelo seguinte programa Prolog:

```
progenitor(maria, josé).
progenitor(joão, josé).
progenitor(joão, ana).
progenitor(josé, júlia).
progenitor(josé, íris).
progenitor(íris, jorge).
```

O programa pode ser ampliado acrescentando-se novos fatos que inclusive podem estabelecer novas relações.

No exemplo abaixo acrescentou-se ao programa original as relações masculino/1 e feminino/1.

Estas relações, que possuem aridade 1, podem ser pensadas como sendo atributos dos objetos a que se aplicam.

```
masculino(joão).
masculino(josé).
masculino(jorge).
feminino(maria).
feminino(ana).
feminino(júlia).
feminino(íris).
```

Exercícios:

Defina as relações:

A) Pai

B) Mãe

C) Irmão

D) Irmã

E) Avô

F) Tia

G) Prima