

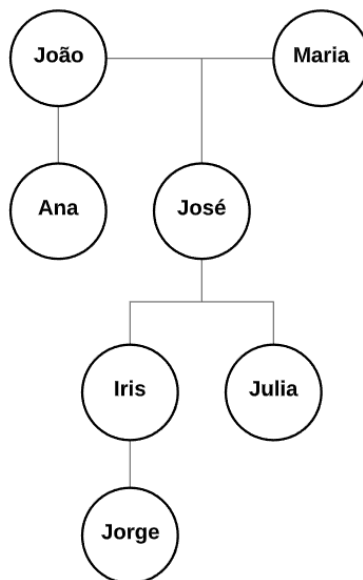
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Lógica Matemática
Trabalho Prolog - Árvore Genealógica

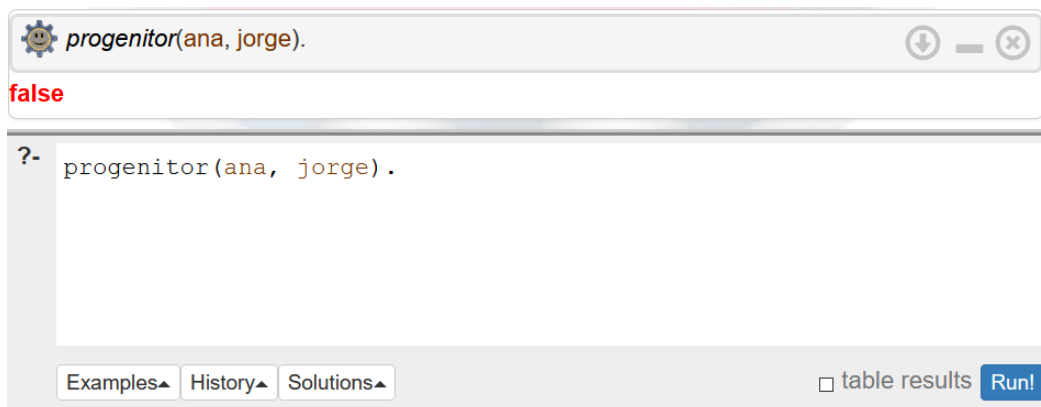
Aluno: Luís Guilherme Barbosa Custódio

Matrícula: 201905500

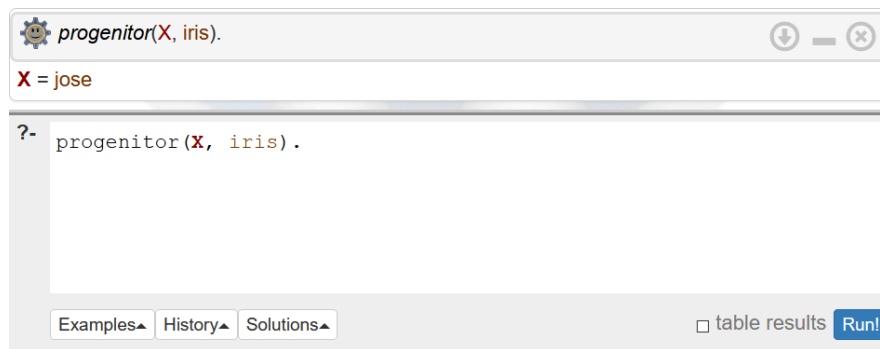
Exercício 1. Desenhe a árvore genealógica representada pela base de conhecimento.



Exercício 2. Escreva uma consulta para responder à seguinte pergunta: “Ana é progenitora de Jorge?”

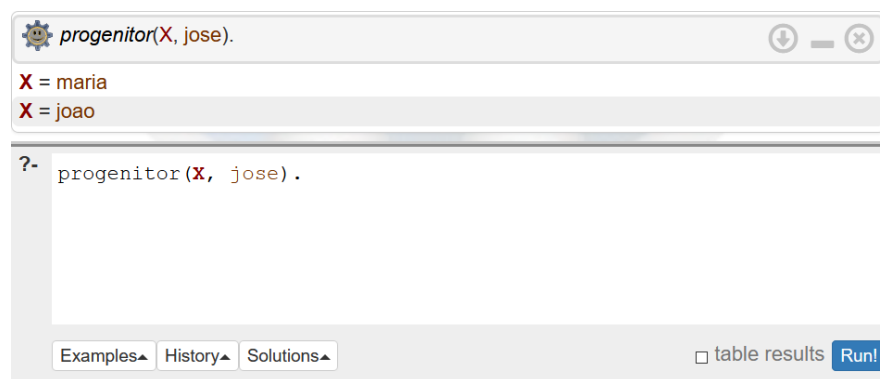


Exercício 3. Escreva uma consulta para retornar os progenitores de Íris.



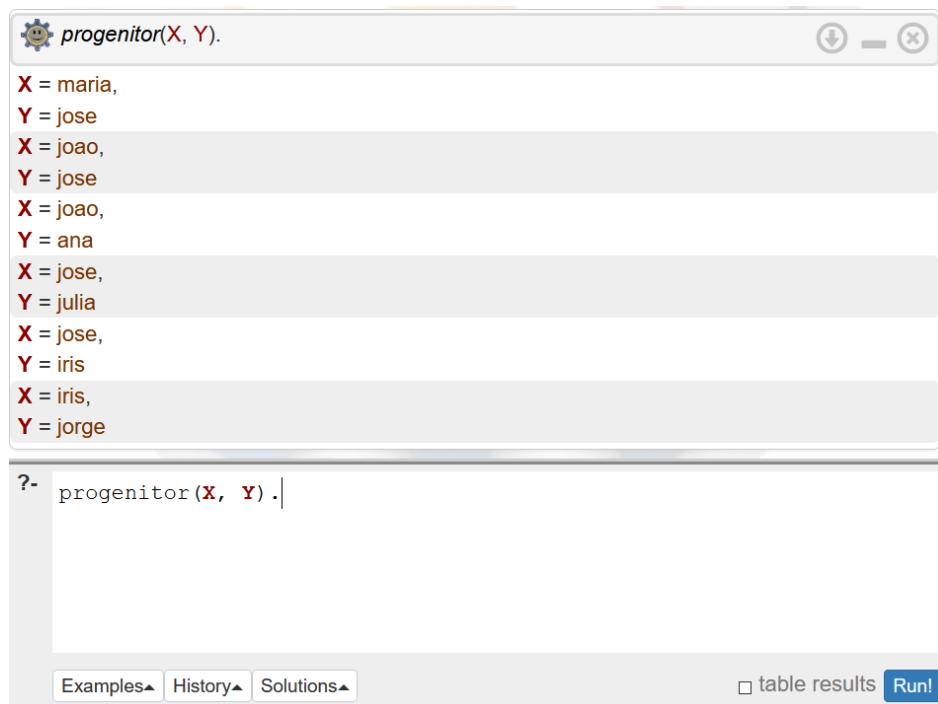
The screenshot shows a Prolog IDE window with the title `progenitor(X, iris).`. Below the title bar, the variable `X` is assigned the value `jose`. The main text area contains the query `?- progenitor(X, iris).`. At the bottom, there are buttons for `Examples`, `History`, and `Solutions`, along with a checkbox for `table results` and a `Run!` button.

Exercício 4. Escreva uma consulta para retornar os progenitores de José.



The screenshot shows a Prolog IDE window with the title `progenitor(X, jose).`. Below the title bar, the variable `X` is assigned the values `maria` and `joao`. The main text area contains the query `?- progenitor(X, jose).`. At the bottom, there are buttons for `Examples`, `History`, and `Solutions`, along with a checkbox for `table results` and a `Run!` button.

Exercício 5. Escreva uma consulta para retornar todos os pares progenitor/filho da base de conhecimento.



The screenshot shows a Prolog IDE window with the title `progenitor(X, Y).`. Below the title bar, the variable `X` is assigned the values `maria`, `joao`, and `jose`, and the variable `Y` is assigned the values `jose`, `ana`, `julia`, `iris`, and `jorge`. The main text area contains the query `?- progenitor(X, Y).`. At the bottom, there are buttons for `Examples`, `History`, and `Solutions`, along with a checkbox for `table results` and a `Run!` button.

Exercício 6. Escreva uma consulta para retornar todos os avós de Jorge. Dica: sua consulta será formada por dois termos separados por vírgula.

 `progenitor(_X, jorge), progenitor(Y, _X)`


`Y = jose`

`?- progenitor(_X, jorge), progenitor(Y, _X)`

Examples History Solutions

☐ table results Run!

Exercício 7. Escreva uma consulta para retornar todos os netos de João.

 `progenitor(joao, _X), progenitor(_X, Y)`

`Y = julia`


`Y = iris`

`?- progenitor(joao, _X), progenitor(_X, Y)`

Examples History Solutions

☐ table results Run!

Exercício 8. Escreva uma consulta para retornar todos os progenitores comuns de José e Ana.

 `progenitor(X, jose), progenitor(X, ana)`

`X = joao`

`?- progenitor(X, jose), progenitor(X, ana)`

Examples History Solutions

☐ table results Run!

Exercício 9. Pode-se definir o predicado `filho/2` como sendo o inverso de `progenitor/2`: se `X` é progenitor de `Y`, então `Y` é filho de `X`. Escreva uma regra para computar o predicado `filho/2` e teste com algumas consultas.

```
filho(Y, X) :- progenitor(X, Y) .
```

The screenshot shows a Prolog interpreter window with the following content:

- Query: `filho(jose, maria)` → Result: `true`
- Query: `filho(jose, joao)` → Result: `true`
- Query: `filho(ana, maria)` → Result: `false`
- Query: `filho(ana, joao)` → Result: `true`
- Query: `filho(ana, julia)` → Result: `false`
- Query: `filho(iris, jose)` → Result: `true`
- Interactive prompt: `?- filho(iris, jose)`
- Bottom bar: `Examples`, `History`, `Solutions`, `table results`, `Run!`

Exercício 10. Escreva regras para os predicados `mãe/2` e `pai/2`. Teste sua regra.

```
mae(X, Y) :- feminino(X), filho(Y, X) .  
pai(X, Y) :- masculino(X), filho(Y, X) .
```

The screenshot shows a Prolog interpreter window with the following content:

- Query: `pai(joao, jose)` → Result: `true`
- Query: `pai(joao, ana)` → Result: `true`
- Query: `mae(maria, jose)` → Result: `true`
- Query: `mae(maria, ana)` → Result: `false`
- Query: `mae(jose, iris)` → Result: `false`
- Query: `pai(jose, iris)` → Result: `true`
- Interactive prompt: `?- pai(jose, iris)`
- Bottom bar: `Examples`, `History`, `Solutions`, `table results`, `Run!`

Exercício 11. Escreva regras para os predicados `avô/2` e `avó/2`. Teste sua regra.

```
avô(X, Z) :- mae(X, Y), filho(Z, Y).  
avó(X, Z) :- pai(X, Y), filho(Z, Y).
```

The screenshot shows a Prolog interpreter window with the following content:

- Query: `avô(maria, iris)` → Result: `false`
- Query: `avó(maria, iris)` → Result: `true` (1 solution)
- Query: `avó(maria, ana)` → Result: `false`
- Query: `avó(jose, jorge)` → Result: `true` (1 solution)
- Query: `avó(ana, jorge)` → Result: `false`

At the bottom, there is a query prompt `?- avó(ana, jorge)` and buttons for `Examples`, `History`, `Solutions`, `table results`, and `Run!`.

Exercício 12. Escreva uma regra para o predicado `irmã/2`. Teste sua regra. Em particular, teste com a consulta `irmã(X, iris)`.

```
irmã(X, Z) :- feminino(X), filho(X, Y), filho(Z, Y).
```

The screenshot shows a Prolog interpreter window with the following content:

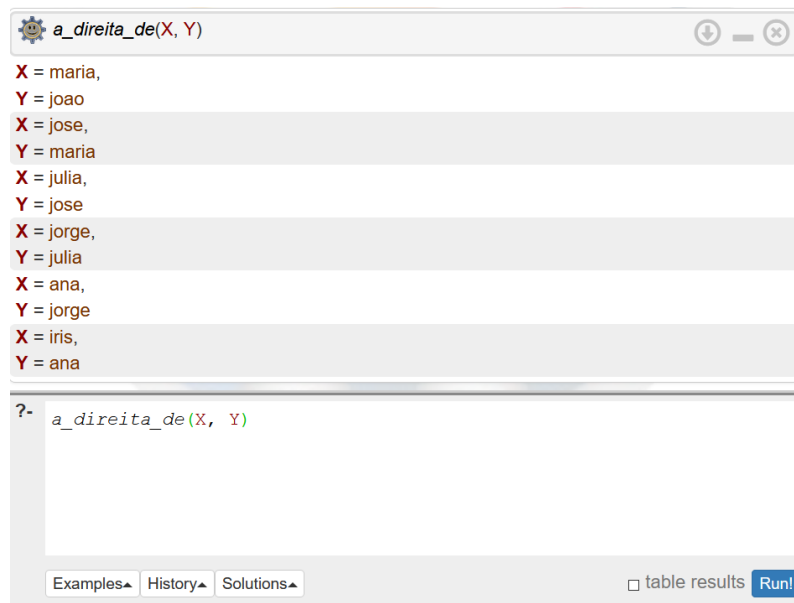
- Query: `irmã(ana, jose)` → Result: `true` (1 solution)
- Query: `irmã(julia, iris)` → Result: `true` (1 solution)
- Query: `irmã(iris, julia)` → Result: `true` (1 solution)
- Query: `irmã(X, iris)` → Results: `X = julia`, `X = iris`

At the bottom, there is a query prompt `?- irmã(X, iris)` and buttons for `Examples`, `History`, `Solutions`, `table results`, and `Run!`.

Exercício 13. Considere o predicado `a_direita_de(X, Y)`, indicando que `X` se senta imediatamente à direita de `Y`. Escreva uma base de conhecimento com esse predicado para representar a configuração de pessoas da mesa.

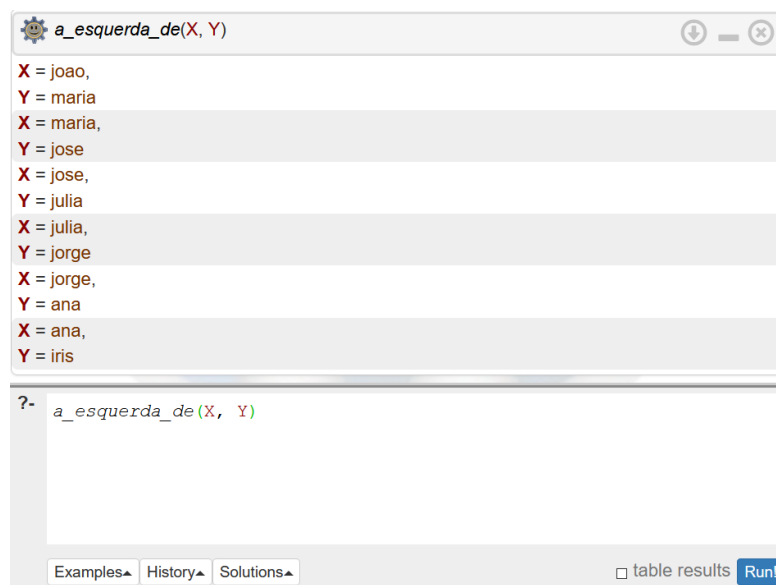
```
peessoas(joao, maria).
peessoas(maria, jose).
peessoas(jose, julia).
peessoas(julia, jorge).
peessoas(jorge, ana).
peessoas(ana, iris).

a_direita_de(X, Y) :- peessoas(Y, X).
```



Exercício 14. Escreva uma regra para o predicado `a_esquerda_de/2`, que é o inverso de `a_direita_de/2`.

```
a_esquerda_de(X, Y) :- peessoas(X, Y).
```



Exercício 15. Escreva uma regra para o predicado `sao_vizinhos_de(Esq, Dir, Meio)`, que indica que `Esq` e `Dir` são os vizinhos à esquerda e à direita de `Meio`, respectivamente.

`sao_vizinhos_de(Esq, Dir, Meio) :- a_direita_de(Dir, Meio), a_esquerda_de(Esq, Meio).`

```
sao_vizinhos_de(Esq, Dir, Meio)
Dir = jose,
Esq = joao,
Meio = maria
Dir = julia,
Esq = maria,
Meio = jose
Dir = jorge,
Esq = jose,
Meio = julia
Dir = ana,
Esq = julia,
Meio = jorge
Dir = iris,
Esq = jorge,
Meio = ana

?- sao_vizinhos_de(Esq, Dir, Meio)
```

Exercício 16. Escreva uma regra para o predicado `adjacente(X, Y)`, que indica se `X` e `Y` estão sentados um ao lado do outro.

`adjacente(X, Y) :- a_direita_de(X, Y), a_esquerda_de(Y, X).`

```
adjacente(X, Y)
X = maria,
Y = joao
X = jose,
Y = maria
X = julia,
Y = jose
X = jorge,
Y = julia
X = ana,
Y = jorge
X = iris,
Y = ana

adjacente(X, ana)
X = iris

?- adjacente(X, Y)
```

Exercício 17. Escreva uma regra para o predicado `esta_na_ponta(X)`, que indica que `X` está em uma das cabeceiras da mesa. Dica: mesmo quem está na cabeceira é vizinho de alguém.

```
esta_na_ponta(X) :- not(a_esquerda_de(_Y, X)), a_direita_de(_Z, X);  
a_esquerda_de(X, _Y), not(a_direita_de(X, _U));  
not(a_esquerda_de(X, _W)), a_direita_de(X, _Z); a_esquerda_de(_Y, X),  
not(a_direita_de(_Z, X)).
```

The screenshot shows a Prolog IDE interface. At the top, a code editor contains the definition of the `esta_na_ponta(X)` predicate. Below the editor, there are three query input fields, each with a gear icon and a close button. The first query is `esta_na_ponta(X)`, which has been executed, resulting in two solutions: `X = joao` and `X = iris`. The second query is `esta_na_ponta(joao)`, which returns `true`. The third query is `esta_na_ponta(iris)`, which also returns `true`. At the bottom, there is a large text area for a new query, currently containing `?- esta_na_ponta(iris)`. Below this area are three tabs: `Examples`, `History`, and `Solutions`. On the right side, there is a checkbox labeled `table results` and a blue button labeled `Run!`.

```
esta_na_ponta(X)
```

`X = joao`
`X = iris`

```
esta_na_ponta(joao)
```

true

```
esta_na_ponta(iris)
```

true

?- `esta_na_ponta(iris)`

Examples History Solutions ☐ table results Run!

=== Código 01 ===

```
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).

masculino(joao).
masculino(jose).
masculino(jorge).
feminino(maria).
feminino(julia).
feminino(ana).
feminino(iris).

filho(Y, X) :- progenitor(X, Y).

mae(X, Y) :- feminino(X), filho(Y, X).
pai(X, Y) :- masculino(X), filho(Y, X).

avó(X, Z) :- mae(X, Y), filho(Z, Y).
avô(X, Z) :- pai(X, Y), filho(Z, Y).

irmã(X, Z) :- feminino(X), filho(X, Y), filho(Z, Y).
```

=== Código 02 ===

```
peessoas(joao, maria).
peessoas(maria, jose).
peessoas(jose, julia).
peessoas(julia, jorge).
peessoas(jorge, ana).
peessoas(ana, iris).

%se senta imediatamente à direita

a_direita_de(X, Y) :- pessoas(Y, X).

a_esquerda_de(X, Y) :- pessoas(X, Y).

sao_vizinhos_de(Esq, Dir, Meio) :- a_direita_de(Dir, Meio),
a_esquerda_de(Esq, Meio).

adjacente(X, Y) :- a_direita_de(X, Y), a_esquerda_de(Y, X).

esta_na_ponta(X) :- not(a_esquerda_de(_Y, X)), a_direita_de(_Z, X);
a_esquerda_de(X, _Y), not(a_direita_de(X, _U)); not(a_esquerda_de(X,
_W)), a_direita_de(X, _Z); a_esquerda_de(_Y, X), not(a_direita_de(_Z,
X)).

%joao -> maria -> jose -> julia -> jorge -> ana -> iris
```