

FORMALIZAÇÃO DE PROGRAMAS E SISTEMAS DE COMPUTAÇÃO SIMPLES

LÓGICA PARA COMPUTAÇÃO

Prof. Jonathan Gil Müller



FORMALIZAÇÃO DE PROGRAMAS

Quais os modelos computacionais (linguagens de programação) existentes?

- **Modelo imperativo (linguagens procedurais)**: são os programas que consistem em **sequências de comandos** que devem ser executados para a obtenção de um resultado. Ou seja, **o programador está dizendo ao computador como resolver o problema**, passo a passo.
 - >> Linguagens orientadas a procedimentos: Pascal, C, entre outras.
 - >> Linguagens orientadas a objetos: Smalltalk, Java , entre outras.



FORMALIZAÇÃO DE PROGRAMAS

Quais os modelos computacionais (linguagens de programação) existentes?

- **Modelo declarativo (linguagens declarativas ou descritivas)**: são os programas **não possuem comandos**, apenas "roteiros" que definem o que deve ser computado, de forma independente das manipulações que devem ser feitas para a obtenção dos resultados. Nesse caso, **o programa contém apenas hipóteses (premissas) e o mecanismo de inferências age por trás do pano para gerar uma conclusão**, ou seja, resposta a pergunta do usuário.

>> Linguagens funcionais: LISP (LISt Processing).

>> Linguagens de programação em lógica: **PROLOG** (PROgrammation en LOGique).



FORMALIZAÇÃO DE PROGRAMAS

- A linguagem de programação PROLOG (PROgramming in LOGic) é uma **linguagem de programação declarativa**.
- A solução de um problema em PROLOG é apresentada através de **fórmulas da lógica de predicados**.
- Assim como na lógica de predicados, um programa PROLOG permite **estabelecer relações entre objetos**, sendo que a representação de determinado problema ou situação é feita através de um conjunto finito de sentenças lógicas denominadas **cláusulas** ou **banco de dados PROLOG**.



FORMALIZAÇÃO DE PROGRAMAS

As **cláusulas** (ou **banco de dados**) podem ser de **dois tipos**:

- **Fatos**: denotam algum conhecimento (informação ou verdade incondicional). Os fatos são **expressos através de predicados** com argumentos constantes tomados do conjunto universo.

EXEMPLO:

homem(mário) .

genitor(mário, josé) .



FORMALIZAÇÃO DE PROGRAMAS

As **cláusulas** (ou **banco de dados**) podem ser de dois tipos:

- **Regras:** expressam como novos fatos podem ser deduzidos, isto é, definem as condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira.

EXEMPLO:

```
pai (X, Y) :- genitor (X, Y) , homem (X) .
```




FORMALIZAÇÃO DE PROGRAMAS

Para **traduzir fórmulas** da lógica dos predicados para programas PROLOG, deve-se usar a seguinte notação:

- **símbolos para constantes:** iniciam com letra minúscula, seguida por uma sequência qualquer de letras, dígitos ou `_`. Ou pode ser qualquer sequência de caracteres entre aspas duplas;
- **símbolos para variáveis:** iniciam com letra maiúscula ou `_`, seguido por uma sequência qualquer de letras, dígitos ou `_`;
- **símbolos para funções ou predicados:** iniciam com letra minúscula, seguida por uma sequência qualquer de letras, dígitos ou `_`;



FORMALIZAÇÃO DE PROGRAMAS

Para **traduzir fórmulas** da lógica dos predicados para programas PROLOG, deve-se usar a seguinte notação:

- todas as cláusulas terminam com ponto (.);
- a linguagem é *case sensitive*;

```
homem (mário) .
```

```
genitor (mário, josé) .
```

```
pai (X, Y) :- genitor (X, Y) , homem (X) .
```




FORMALIZAÇÃO DE PROGRAMAS

Uso de **conectivos** lógicos em PROLOG:

a) $(\neg\alpha)$ - negação

lógica de predicados:

$\neg(\text{homem}(\text{raquel}))$

em PROLOG:

not(homem(samuel))

b) $(\alpha \wedge \beta)$ - conjunção

lógica de predicados:

genitor(x,y) \wedge homem(x)

em PROLOG:

genitor(X,Y) , homem(X)



FORMALIZAÇÃO DE PROGRAMAS

Uso de **conectivos** lógicos em PROLOG:

c) $(a \vee b)$ - disjunção

lógica de predicados:

$((\text{pai}(y,x) \vee (\text{mae}(y,x)) \wedge \text{homem}(x)) \rightarrow \text{filho}(x,y)$

em PROLOG:

`filho(X,Y) :- pai(Y,X), homem(X).`

`filho(X,Y) :- mae(Y,X), homem(X).`

`filho(X,Y) :- (pai(Y,X) ; mae(Y,X)), homem(X).`

d) $(a \rightarrow b)$ - implicação

lógica de predicados:

$(\text{genitor}(x,y) \wedge \text{homem}(x)) \rightarrow \text{pai}(x,y)$

em PROLOG:

`pai(X,Y) :- genitor(X,Y), homem(X)`



FORMALIZAÇÃO DE PROGRAMAS

A execução de um programa PROLOG consiste em responder a várias **consultas/perguntas**.

Para tanto, o interpretador PROLOG aciona a máquina de inferência e aplica as regras de dedução para deduzir conclusões a partir dos fatos e das regras declarados no banco de dados. Tem-se que "o mecanismo de inferência age por trás do pano para construir uma sequência de demonstração" (GERSTING, 2001).

A resposta para uma consulta depende da pergunta feita e da base de fatos e regras. Pode ser: **true** (yes), **false** (no), ou um ou mais **valores**.



FORMALIZAÇÃO DE PROGRAMAS

O SWI-Prolog é
uma linguagem
importante para
**criação de
sistemas para
a área de IA.**

SWI-Prolog

<https://www.swi-prolog.org/>



SWI Prolog

Robust, mature, free. **Prolog for the real world.**

HOME

DOWNLOAD

DOCUMENTATION

TUTORIALS

COMMUNITY

USERS

WIKI

SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications. Join over a million users who have downloaded SWI-Prolog. [more...](#)

Download SWI-Prolog

Get Started

Try SWI-Prolog online



FORMALIZAÇÃO DE PROGRAMAS



SWI-Prolog downloads

[HOME](#)[DOWNLOAD](#)[DOCUMENTATION](#)[TUTORIALS](#)[COMMUNITY](#)[USERS](#)[WIKI](#)

Available versions

The **stable** release is infrequently updated. It is fine for running basic Prolog code without surprises. The **development** version is released roughly every two to four weeks. This is the recommended version for developers and users of applications such as [SWISH](#) or [ClioPatria](#). Finally, the **GIT** and **daily** versions are for developers that want to contribute or have immediate access to patches. These versions are generally fine, but occasionally suffer from regression.

- [Stable release](#)
- [Development release](#)
- [Daily builds for Windows](#)
- Browse GIT [repository](#).

Read more about

- Available SWI-Prolog [versions](#)
- Information on [Linux packages and building on Linux](#)



FORMALIZAÇÃO DE PROGRAMAS



Download SWI-Prolog stable versions

[HOME](#)[DOWNLOAD](#)[DOCUMENTATION](#)[TUTORIALS](#)[COMMUNITY](#)[USERS](#)[WIKI](#)

Linux versions are often available as a package for your distribution. We collect information about available packages and issues for building on specific distros [here](#). We provide a [PPA](#) for [Ubuntu](#) and [snap images](#)






Please check the [windows release notes](#) (also in the SWI-Prolog startup menu of your installed version) for details.



Examine the [ChangeLog](#).

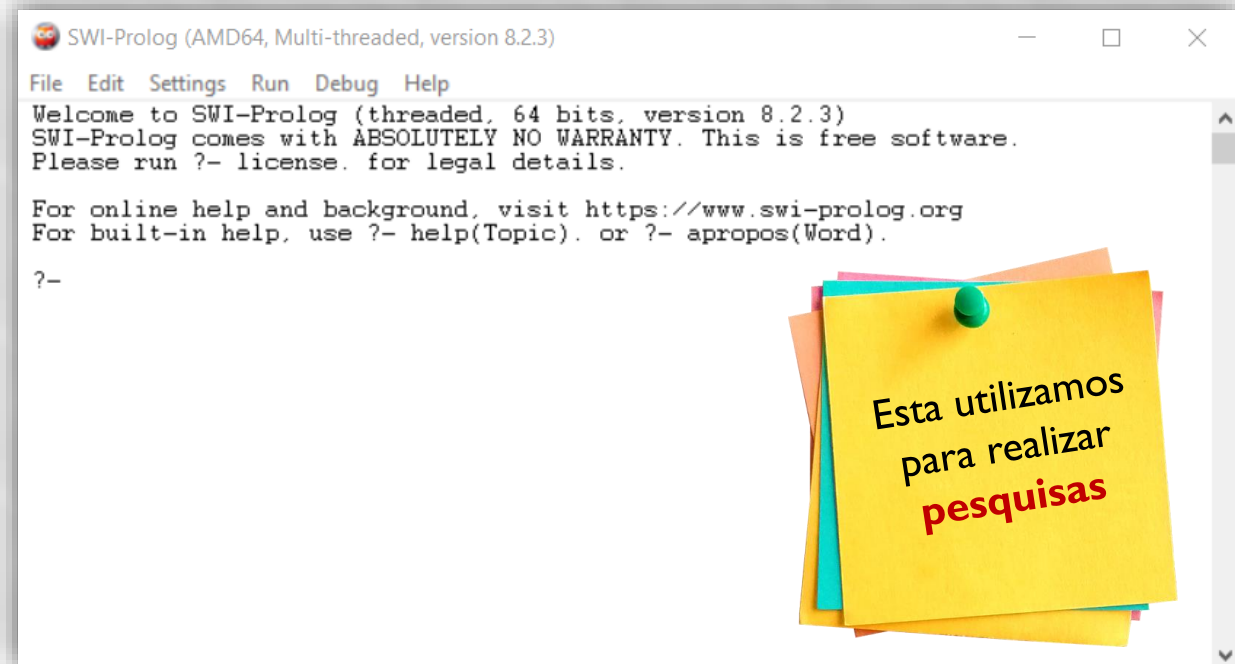
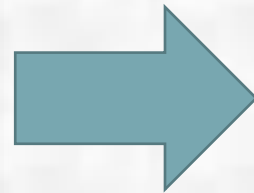
Binaries

	11,825,869 bytes	SWI-Prolog 8.2.3-1 for Microsoft Windows (64 bit) Self-installing executable for Microsoft's Windows 64-bit editions. Requires at least Windows 7. See the reference manual for deciding on whether to use the 32- or 64-bits version. This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: b4297d1d60ce050daf0d07764487370ef475cad70b54a96b074f83138d792cd6
	11,468,354 bytes	SWI-Prolog 8.2.3-1 for Microsoft Windows (32 bit) Self-installing executable for MS-Windows. Requires at least Windows 7. Installs swipl-win.exe and swipl.exe . This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: a05901d850c8e3138dab176ab10fab6b44ba452d35c7076b65d213dfcd89c209
	27,660,126 bytes	SWI-Prolog 8.2.3-1 for MacOSX 10.12 (Sierra) and later on intel Installer with binaries created using Macports . Installs <code>/opt/local/bin/swipl</code> . Needs xquartz (X11) and the Developer Tools (Xcode) installed for running the development tools SHA256: 235a08a3fcc4eecfa023a8e4f89a78ce54f7723120883f9944efaaaf89930f444



FORMALIZAÇÃO DE PROGRAMAS

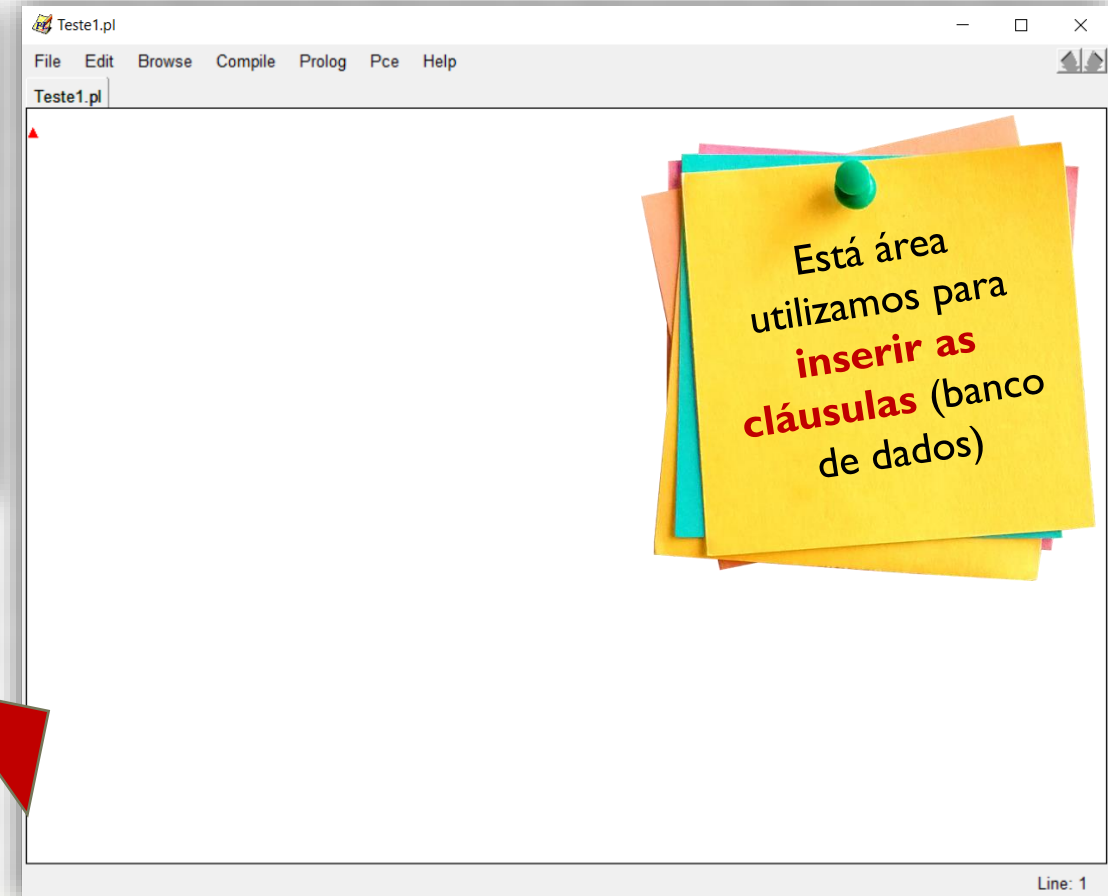
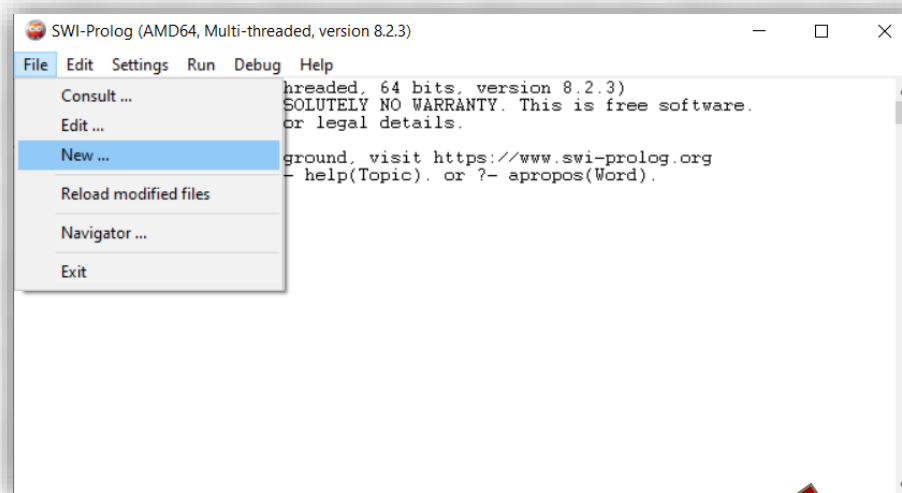
O SWI-Prolog tem **duas áreas de trabalho**. Uma para **inserir as cláusulas** do programa e outra para **realização de pesquisas**.





FORMALIZAÇÃO DE PROGRAMAS

Para acessar a segunda área de trabalho precisamos abrir um novo arquivo...





FORMALIZAÇÃO DE PROGRAMAS

Após inserir as informações do banco de dados clicar em **"Compile buffer"** para salvar as informações e sincronizar com a outra área de trabalho.

The screenshot displays a Prolog development environment. The top window, titled 'Teste1.pl [modified]', contains a menu bar with 'File', 'Edit', 'Browse', 'Compile', 'Prolog', 'Pce', and 'Help'. The 'Compile' menu is open, showing options: 'Make' (Control-c Control-m), 'Compile buffer' (Control-c Control-b), and 'Consult selection'. The code in the editor is as follows:

```
%Fatos
genitor(maria, carmem).
genitor(maria, joao).
genitor(joao, carmem).
genitor(joao, pedro).
genitor(angélica, carla).
genitor(angélica, raul).
genitor(carmem, nicolas).
genitor(carmem, josé).
genitor(pedro, luisa).
genitor(raul, tom).
genitor(luisa, felipe).
```

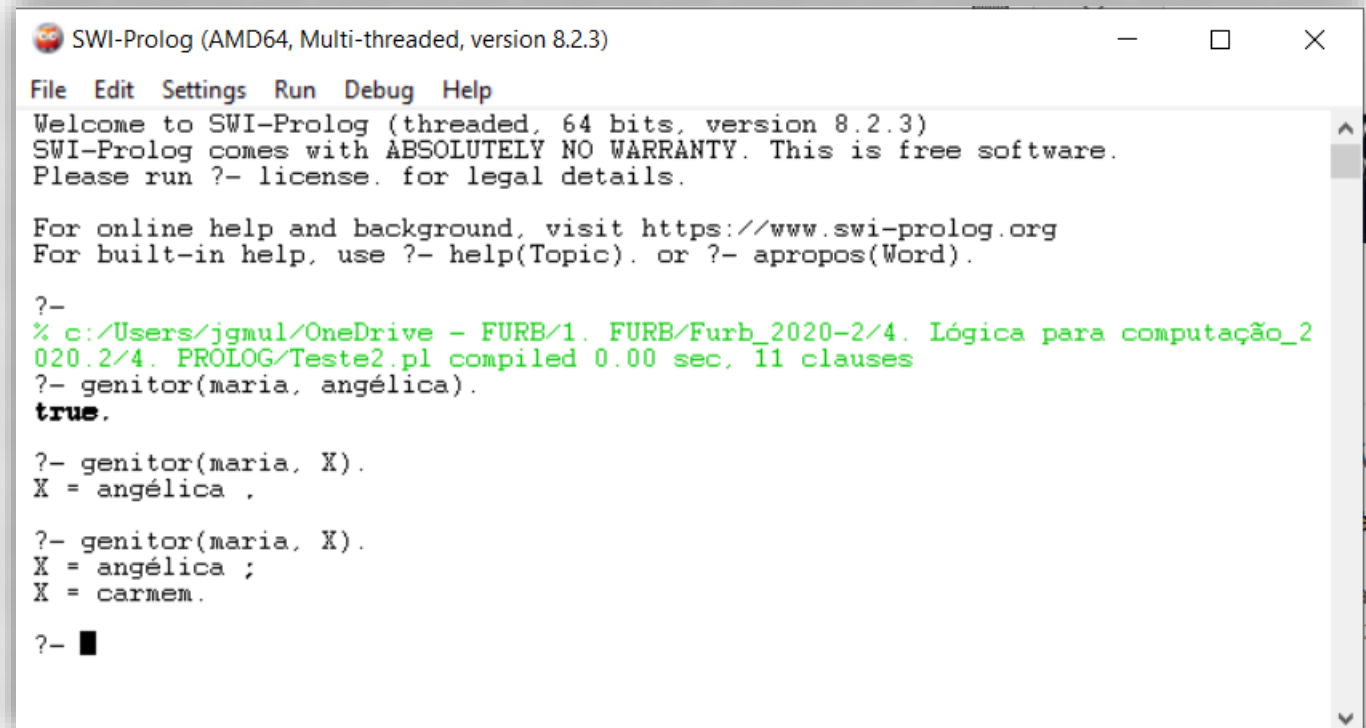
The bottom window, titled 'SWI-Prolog (AMD64, Multi-threaded, version 8.2.3)', shows the Prolog interpreter's output. It includes a welcome message and a list of commands. The last line of output, which is highlighted with a red box, indicates the successful compilation of a file:

```
?-
% c:/Users/jgmul/OneDrive - FURB/1. FURB/Furb_2020-2/4. Lógica para computação_2
020.2/4. PROLOG/Teste2.pl compiled 0.00 sec, 11 clauses
```



FORMALIZAÇÃO DE PROGRAMAS

Após inserir
as cláusulas
no programa
podemos
retornar a
primeira área
de trabalho
para
realização de
perguntas.



```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/jgmul/OneDrive - FURB/1. FURB/Furb_2020-2/4. Lógica para computação_2
020.2/4. PROLOG/Teste2.pl compiled 0.00 sec, 11 clauses
?- genitor(maria, angélica).
true.

?- genitor(maria, X).
X = angélica ;

?- genitor(maria, X).
X = angélica ;
X = carmen.

?-
```



FORMALIZAÇÃO DE PROGRAMAS

Para facilitar
a pesquisa em
nosso programa
podemos
inserir regras
em nosso banco
de dados!

```
teste2.pl
File Edit Browse Compile Prolog Pce Help
teste2.pl
% Fatos
genitor(maria,angélica).
genitor(maria, carmem).
genitor(joão,carmem).
genitor(joão, pedro).
genitor(angélica, carla).
genitor(angélica, raul).
genitor(carmem, nicolas).
genitor(carmem, josé).
genitor(pedro, luisa).
genitor(raul, tom).
genitor(luisa, felipe).

feminino(maria).
feminino(angélica).
feminino(carmem).
feminino(carla).
feminino(luisa).

masc(joão).
masc(pedro).
masc(raul).
masc(nicolas).
masc(josé).
masc(felipe).
masc(tom).

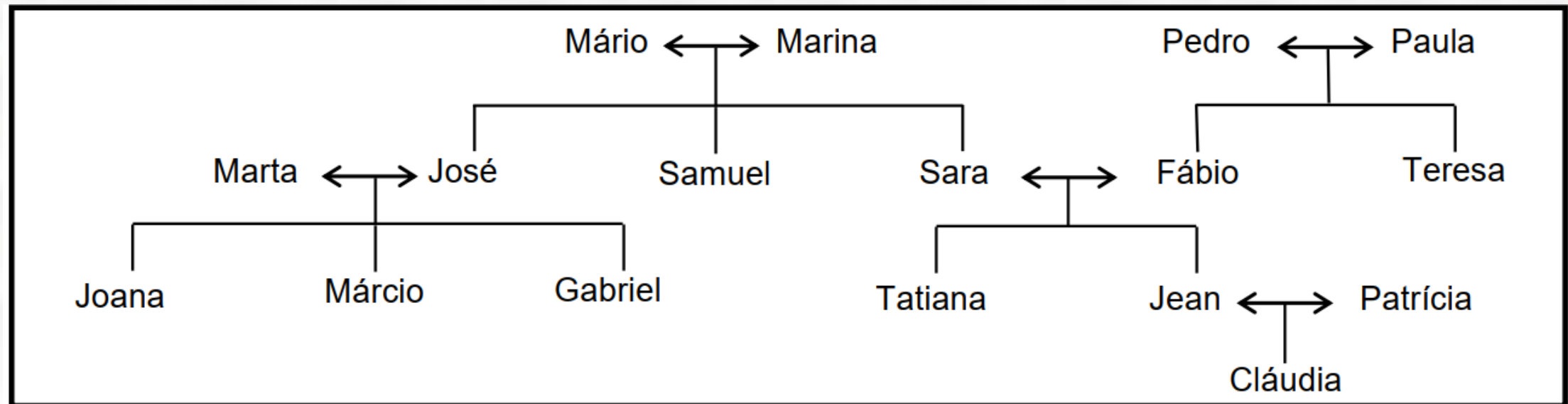
% Regras.

filho(Y,X):- genitor(X,Y),masc(Y) .
filha(Y,X):- genitor(X,Y),feminino(Y) .
```



FORMALIZAÇÃO DE PROGRAMAS

Base de dados para realização do exemplo no SWI-Prolog





FORMALIZAÇÃO DE PROGRAMAS

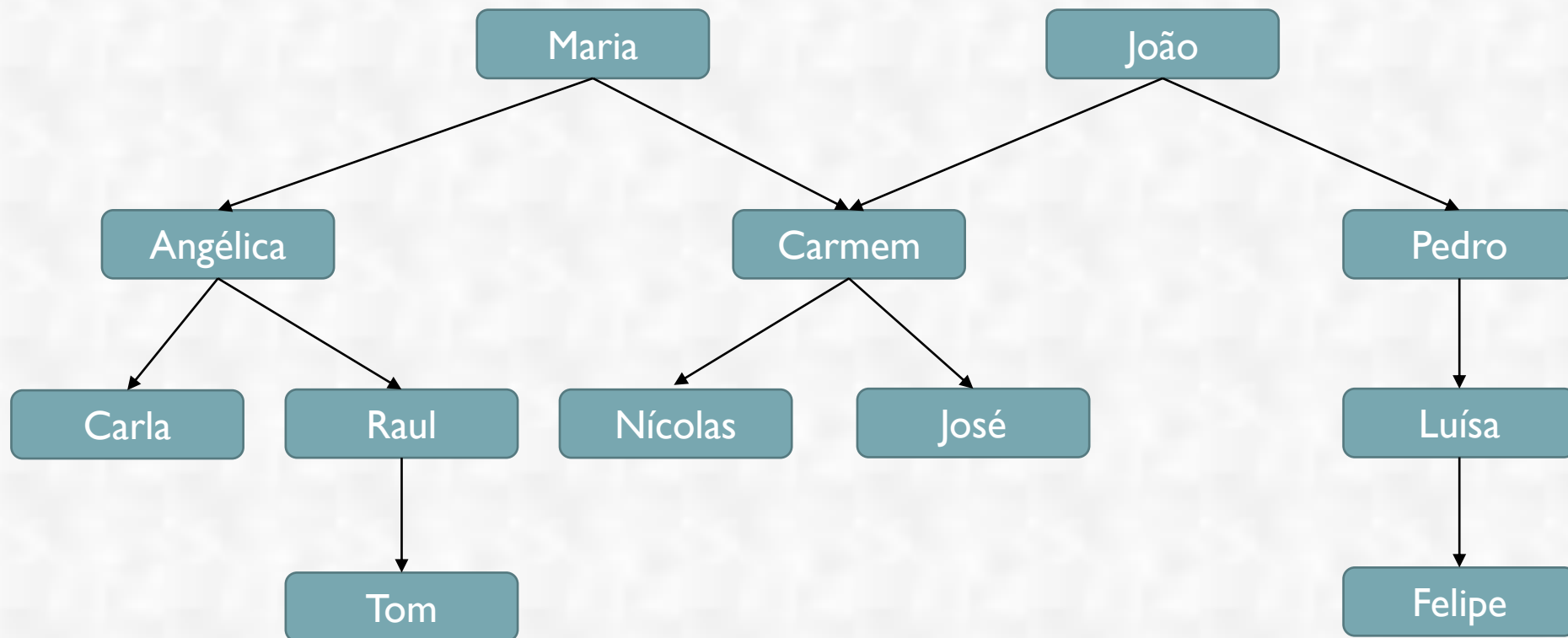
Vamos realizar as seguintes consultas:

- Verificar se Mário é um homem.
- Verificar se Joana é um homem.
- Verificar se Sara não é um homem.
- Verificar quem são as mulheres da família.
- Verificar se o Mário é pai do Samuel.
- Verificar quem são os filhos de Mário.
- Verificar quem são os genitores do Márcio.
- Verificar se Mário e Joana são um casal.
- Verificar quem é o casal de Mário.
- ...



FORMALIZAÇÃO DE PROGRAMAS

Base de dados para realização do exemplo no SWI-Prolog





DOCUMENTOS CONSULTADOS/RECOMENDADOS

1. BRATKO, I. Prolog programming for artificial intelligence. 2nd ed. Wokingham: Addison-Wesley, 1990.
2. CASANOVA, M. A.; GIORNO, F. A. C.; FURTADO, A. L. Programação em lógica e a linguagem PROLOG. São Paulo: E. Blucher, 1987.
3. GERSTING, J. L. Fundamentos matemáticos para a ciência da computação. 4.ed. Rio de Janeiro: LTC, 2001.
4. PRICE, A. M. A.; TOSCANI, S. S. Implementação de linguagens de programação: compiladores. Porto Alegre: Sagra Luzzatto, 2000.
5. STERLING, L.; SHAPIRO, E. The art of Prolog: advanced programming techniques. 2nd ed. Cambridge: MIT, 1994. p. 411-478.
6. WIELEMAKER, J. SWI Prolog. [S.l.], [2011]. Disponível em: <<http://www.swi-prolog.org/>>. Acesso em: 23 fev. 2013.



The End

