

Execução de programas em Prolog

Sumário

| | | |
|------------|--|------------|
| 1.1 | Objetivos | 1-1 |
| 1.2 | Conceitos | 1-1 |
| 1.2.1 | Prolog | 1-1 |
| 1.2.2 | Constantes | 1-2 |
| 1.2.3 | Variáveis | 1-2 |
| 1.2.4 | Cláusulas | 1-3 |
| 1.3 | Exemplo | 1-3 |
| 1.3.1 | SWI-Prolog | 1-3 |
| 1.3.2 | Interrogações à base de conhecimento | 1-5 |
| 1.4 | Exercícios | 1-7 |
| 1.4.1 | Socorro! | 1-7 |
| 1.4.2 | Fatos, queries e regras | 1-7 |

1.1 Objetivos

- Aprender a trabalhar com o ambiente interativo do Prolog.
- Escrever fatos e regras simples.
- Fazer consultas à base de conhecimento.

1.2 Conceitos

1.2.1 Prolog

Prolog (*PRO*gramming in *LOGic*) é uma linguagem *declarativa* para *computação simbólica*.

Diferentemente das linguagens imperativas, um programa em Prolog não descreve como obter a solução de um dado problema através de uma sequência de passos a ser seguidos pelo computador.

Um programa em Prolog consiste numa base de dados (também chamada de base de conhecimento) de fatos e regras (relações lógicas) que *descrevem* o problema.

Em vez de *executar* o programa para obter a solução, o usuário faz uma pergunta. Quando uma pergunta é colocada, o sistema efetua uma procura na base de dados de fatos e regras para determinar (por *dedução lógica*) a resposta.

Prolog oferece uma estrutura de dados uniforme, chamada *termo*, a partir da qual todos os dados, e também os próprios programas em Prolog, são construídos.

1.2.2 Constantes

Constantes são termos que nomeiam objetos específicos ou relações específicas. Podem ser átomos ou números.

Átomos podem ser construídos de três formas:

1. sequência de letras, dígitos e sublinhado começando por uma letra minúscula. Exemplos:

```
mario
livro_texto
cicl21
estruturas_de_dados_2
```

2. sequência de caracteres especiais: + - * / \ ~ ^ < > : . ? @ # \$ &. Exemplos:

```
+
<...>
===>
...:.
:::..
```

3. sequência de caracteres entre apóstrofes ('). Exemplos:

```
'george-smith'
'Beta'
'2304alpha'
'fim de arquivo'
```

Números em Prolog incluem os números inteiros e os reais. No entanto os números reais não costumam ser muito utilizados, uma vez que o Prolog é antes de mais nada uma linguagem adequada à computação simbólica. Exemplos:

```
-17
-2.67e2
0
1
99.9
512
6.02e-23
```

1.2.3 Variáveis

Variáveis são sequências de letras, dígitos e sublinhado começadas por letra maiúscula ou por sublinhado. Exemplos:

```
Resposta
X
_beta
NomeVar
Uma_variavel_muito_longa
```

Uma variável denota um objeto específico, porém desconhecido.

1.2.4 Cláusulas

Uma cláusula permite especificar uma relação entre objetos ou uma propriedade de um objeto. Quando a relação ou propriedade não depende de nenhuma condição, sendo considerada verdadeira sempre, a cláusula é um **fato**. Caso contrário ela é uma **regra**.

O fato de que Mário é pai de Manuel pode ser escrito em Prolog da seguinte forma:

```
pai(mario, manuel).
```

Neste exemplo `pai` é o nome da relação, `mario` e `manuel` são os argumentos.

Se o Mário é pai do Manuel, então o Manuel é filho do Mário. Podemos dizer que:

Para todo A e B , A é filho de B se B é pai de A .

Este conhecimento pode ser descrito pela regra:

```
filho(A,B) :- pai(B,A).
```

A principal diferença entre fatos e regras é que os fatos expressam relações que são sempre verdadeiras, enquanto as regras definem uma relação que é verdadeira em determinadas condições. Neste caso se a condição `pai(B,A)` (o **corpo** da regra) for verdade, então podemos concluir que `filho(A,B)` (a **cabeça** da regra) é verdade.

Um conjunto de fatos e regras com o mesmo nome definem um **predicado** (ou procedimento). Neste caso definimos os predicados `pai/2` e `filho/2`. O número após a barra indica a **aridade** do predicado, isto é, o número de argumentos do predicado.

1.3 Exemplo

Consideremos a árvore genealógica representada na figura 1.1. A árvore representa informação sobre o grau de parentesco entre diversos indivíduos. Várias conclusões podem ser extraídas da figura: o Mário é pai do Manuel, o Mário é pai da Teresa, o João é filho do Manuel, o Manuel é avô da Maria, etc. Se escolhermos a relação *ser pai de* como a relação relevante a representar, o conhecimento adquirido a partir da figura leva à definição dos seguintes fatos:

```
pai(mario, manuel).
pai(mario, teresa).
pai(manuel, joao).
pai(joao, maria).
pai(joao, rui).
```

Este conhecimento é aquele que é explícito a partir da figura. É agora possível fazer perguntas e obter mais conhecimento a partir da informação existente. Comece por escrever os fatos no arquivo `bd.pl`¹.

1.3.1 SWI-Prolog

A implementação de Prolog que iremos utilizar é o SWI-Prolog (disponível em <http://www.swi-prolog.org>). SWI-Prolog é um ambiente de desenvolvimento em Prolog, de licença livre (GPL), largamente usado em pesquisa, educação, e desenvolvimento de aplicações comerciais. O seu manual de referência pode ser acessado no endereço <http://www.swi-prolog.org/pldoc/refman/>.

Ao instalar o SWI-Prolog em Windows preste atenção ao diretório de trabalho que define, será esse o diretório em que os arquivos serão procurados por omissão (é possível mudar de diretório de trabalho com o predicado `cd/1`).

¹Tradicionalmente a extensão de arquivos Prolog é `.pl`. No entanto, em alguns sistemas isso pode causar conflito com o Perl pelo que, durante a instalação, o SWI-Prolog permite configurar a extensão a utilizar. Nestes apontamentos iremos utilizar `.pl`.

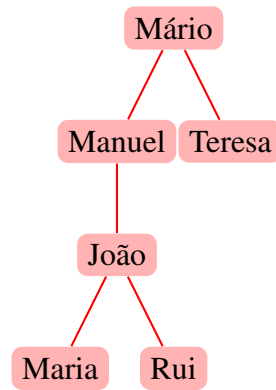


Figura 1.1: *Árvore Genealógica*

- Executando o interpretador:
 - Windows: procure a entrada apropriada no menu: Iniciar -> SWI-Prolog -> Prolog. A figura 1.2 mostra o SWI-Prolog sendo executado no Windows.

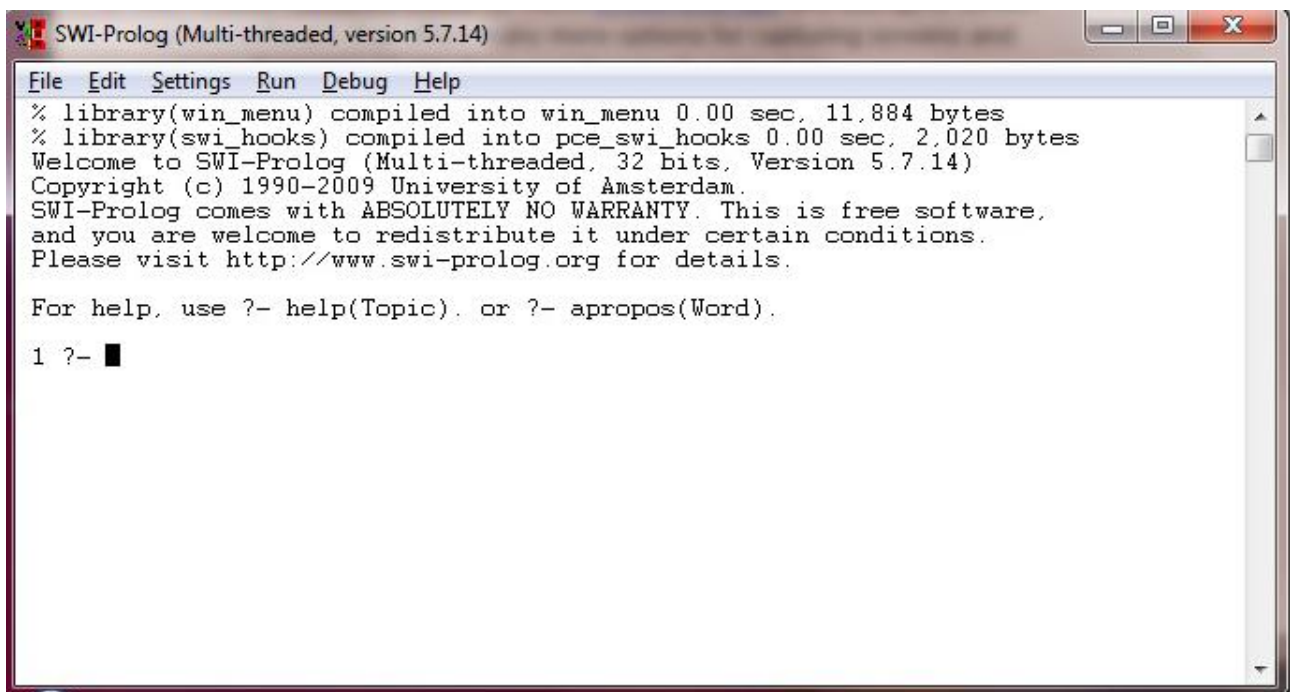


Figura 1.2: *SWI-Prolog*

- Linux: execute o comando `swipl` em um terminal. A figura 1.3 mostra o SWI-Prolog sendo executado em um terminal no Linux.

O prompt `?-` significa que o interpretador está à espera de instruções.

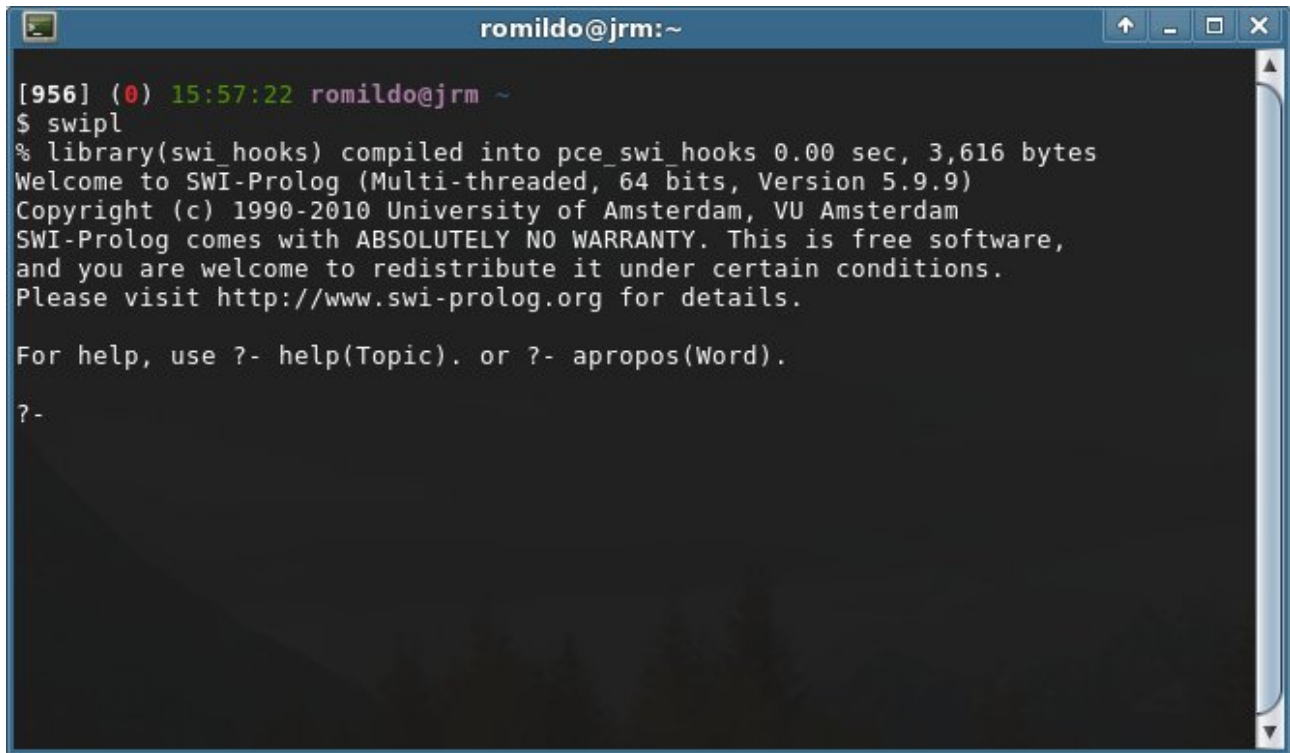
- Carregando um arquivo:

```
?- consult('bd.pl').
```

ou,

```
?- consult(bd). %% bd corresponde ao nome do arquivo sem extensão
```

ou,



```

romildo@jrm:~
[956] (0) 15:57:22 romildo@jrm ~
$ swipl
% library(swi_hooks) compiled into pce_swi_hooks 0.00 sec, 3,616 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.9.9)
Copyright (c) 1990-2010 University of Amsterdam, VU 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).

?-

```

Figura 1.3: SWI-Prolog

?- [bd].

- Visualizar o conhecimento existente na base de dados:

```

?- listing.
pai(mario, manuel).
pai(mario, teresa).
pai(manuel, joao).
pai(joao, maria).
pai(joao, rui).
yes

```

O SWI-Prolog-Editor (<http://lakk.bildung.hessen.de/netzwerk/faecher/informatik/swiprolog/indexe.html>) é um ambiente de programação que facilita o desenvolvimento de aplicações usando o SWI-Prolog. Ele está disponível apenas para o sistema Windows.

A figura 1.4 o SWI-Prolog-Editor sendo executado no Windows.

1.3.2 Interrogações à base de conhecimento

É então possível interrogar o sistema de maneira a extrair informação da base de conhecimento. Por exemplo:

- Verificar se o mario é pai do manuel.

```

?- pai(mario,manuel).
yes

```

- Verificar se o rui é pai do joao.
- Verificar se a mario é pai da teresa.

Ou ainda fazer perguntas mais complexas como:

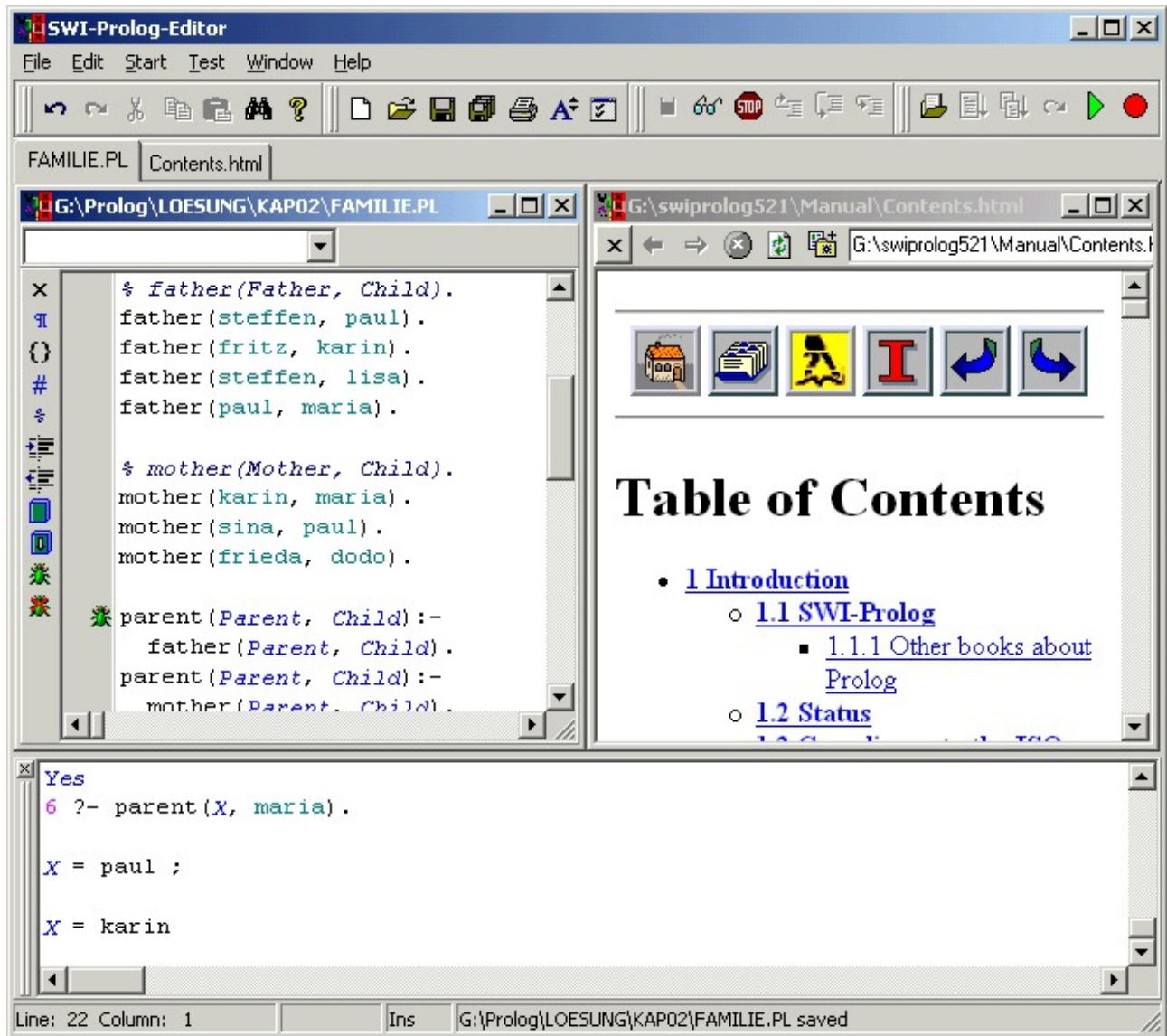


Figura 1.4: SWI-Prolog

- Determinar os filhos do mario.

```
?- pai(mario,X).
X = manuel ? ;
%% ";" dá a resposta seguinte no caso de existir
X = teresa ? ;
no
?-
```

- Quem é o pai da teresa?
- Os pares pai/filho existentes na base de conhecimento.
- Verificar se o pai do rui é o mesmo que o pai da maria.

```
?- pai(X,rui),pai(X,maria).
```

Repare que a , significa a conjunção de termos. Tem a mesma semântica que o AND (ou \wedge).

- Quem é o avô do joao.

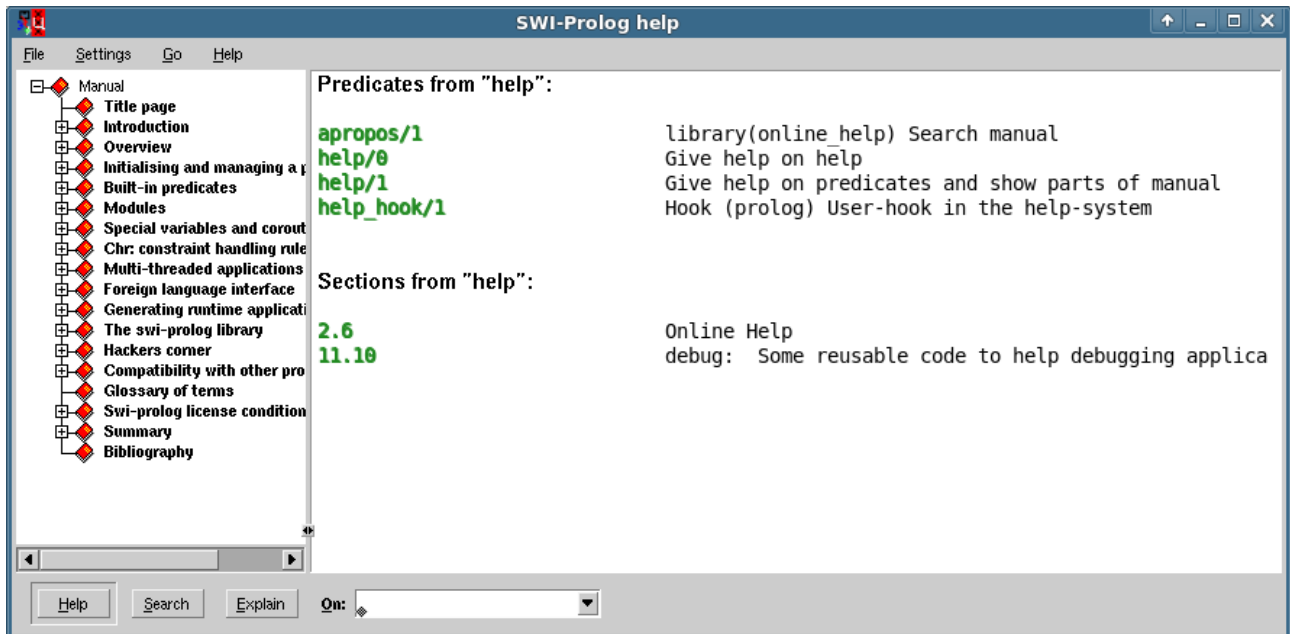


Figura 1.5: Janela de Help do SWI-Prolog

```
?- pai(X,joao),pai(Y,X).
X = manuel,
Y = mario ? ;    %% Y é o avô
no
```

1.4 Exercícios

1.4.1 Socorro!

Experimentar os predicados `help` e `apropos`:

1. Depois de iniciar o interpretador, escreva a query `apropos(help)`. (não esqueça o ponto — ‘.’.)
Note que o predicado `apropos` lhe fornece uma lista de predicados e seções do manual relacionados com o assunto indicado como parâmetro. (Se estiver utilizando a versão 4.0 ou superior do SWI Prolog, essa lista aparecerá numa nova janela — ver figura 1.5.)
2. Experimente agora a query `help(help)`. (Se estiver a utilizar a versão 4.0 ou superior, poderá utilizar a área de diálogo que aparece na janela de help.) Note que o predicado `help` lhe fornece ajuda sobre os predicados indicados como parâmetro.
3. Experimente as queries `help(1)`. e `help(2-1)`. Note que quando o parâmetro do predicado `help` é um número, é apresentada a seção do manual com esse número. Na verdade tem todo o manual disponível online para consulta!
4. Finalmente, experimente a query `help(halt)`.

1.4.2 Fatos, queries e regras

1. Escreva os seguintes fatos (pode utilizar `[user]`. ou então escrever os fatos num arquivo e utilizar `consult(1)`:

```
aluno(joao,ppi).
aluno(pedro,ppi).
```

```
aluno(maria,ppiii).  
aluno(rui,ppiii).  
aluno(manuel,ppiii).  
aluno(pedro,ppiii).  
aluno(rui,ppiv).
```

- (a) Verifique que os fatos estão presentes na Base de Conhecimento (utilize o predicado `listing`).
 - (b) Escreva uma query que verifique se joao é aluno de ppiii.
 - (c) Escreva uma query que verifique se rui é aluno de ppi — note o efeito do Princípio do Mundo Fechado.
 - (d) Escreva uma query que verifique se joao e maria são ambos alunos de ppiv — joao e maria são ambos alunos de ppiv se joao for aluno de ppiv e maria for aluna de ppiv.
 - (e) Escreva uma query que permita saber quem é aluno de ppiii.
 - (f) Escreva uma query que permita saber de que disciplinas é rui aluno.
2. Adicione os seguintes fatos à base de conhecimento (se anteriormente utilizou `[user]` . é agora um bom momento para começar a utilizar arquivos):

```
estuda(joao).  
estuda(maria).  
estuda(manuel).
```

- (a) Sabendo que a aluno *A* faz a disciplina *P* se *A* é aluno de *P* e *A* estuda, escreva uma query que lhe permita saber se maria vai fazer ppiii.
- (b) Experimente agora a query `aluno(X,ppiii),estuda(X) ..` O que lhe permite esta query saber?
- (c) Utilizando a query da alínea anterior, acrescente à Base de Conhecimento o predicado `fazppiii/1` e escreva uma query que lhe permita saber quem faz ppiii (não se esqueça de fazer `consult` do arquivo).