Ficha Prática 1[1]

1.1 Objectivos

1. Aprender a trabalhar com o interpretador.
2. Fazer interrogações à informação existente.

1.2 Conceitos

1.2.1 Prolog

Prolog (*PROgramming in LOGic*) é uma linguagem declarativa para computação simbólica:

- Um programa em Prolog não descreve como *calcular* a solução de um dado problema.
- Um programa em Prolog consiste numa base de dados de factos e relações lógicas (regras) que descrevem o problema (também chamada de Base de Conhecimento).

Em vez de *correr* o programa para obter a solução, o utilizador faz uma pergunta. Quando uma pergunta é colocada, o sistema efectua uma procura na base de dados de factos e regras até determinar (por dedução lógica) a resposta.

1.2.2 Átomos, variáveis e números

Átomos podem ser construídos de três formas:

1. *strings* de letras, dígitos e *underscores* começando por uma letra minúscula:

```
rui
mario
tcp4
t_3
t___
parad_prog_3
```

2. *strings* de caracteres especiais:

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

3. *strings* de carateres entre plicas:

[1] Actualizada em 07/10/2004.

```
'daytona'
'kafka'
```

Variáveis são sequências de letras começadas por letra maiúscula ou por um *underscore*.

```
X
Y
_X
NomeVar
```

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 uma linguagem adequada à computação simbólica.

1.2.3 Factos e regras

O facto de que o Mário é pai do 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. Note que `mario` e `manuel` foram escritos com letra minúscula. Porquê? Consegue pensar numa alternativa?

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

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

Este conhecimento pode ser descrito pela expressão:

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

a que se chama uma regra.

A principal diferença entre factos e regras é que os factos expressão relações que são sempre verdadeiras, enquanto as regras definem que uma relação 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 factos e regras com o mesmo nome definem um predicado (ou procedimento). Neste caso definimos os predicados pai/2 e filho/2 (os números após a barra indicam o número de parâmetros 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 se podem extrair 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 existência dos seguintes ***factos***:

```
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 factos no ficheiro `bd.swi`[2].

[2]Tradicionalmente a extensão de ficheiros 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 ".swi".

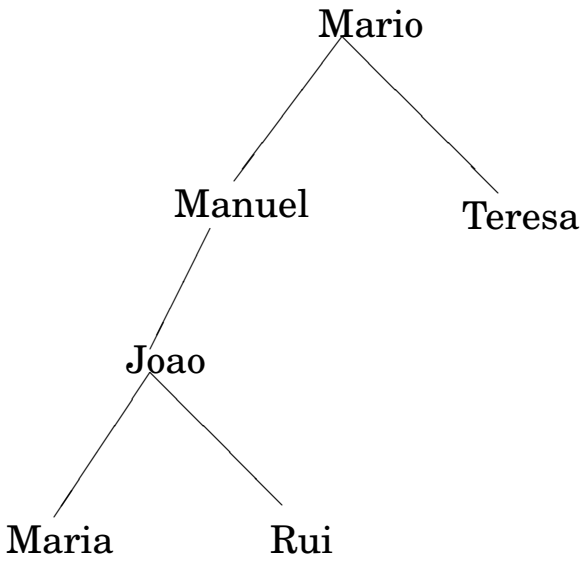

Figura 1.1: Árvore Genealógica

1.3.1 SWI-Prolog

O interpretador de Prolog que iremos utilizar é o SWI-Prolog (encontra-se disponível em `http://www.swi-prolog.org`). Ao instalar o SWI-Prolog em Windows preste atenção à directoria de trabalho que define, será essa a directoria em que os ficheiros serão procurados por omissão (é possível mudar de directoria de trabalho com o predicado `cd/1`).

- Arrancar com o interpretador:
  - Windows: procure a entrada apropriada no menu Start/Iniciar (nos laboratórios pode também procurar o ficheiro `pl.exe` em `t:\bin\pl\bin\plwin`).
  - Linux: execute o comando `pl`.

  O *prompt* "`| ?-`" significa que o interpretador está à espera de instruções.
- Carregar um ficheiro

  ```
  | ?- consult('bd.swi').
  ```

  ou,

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

  ou,

  ```
  | ?- [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
```

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:

- 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`.

  ```
  | ?- 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 secções do manual relacionados com o assunto indicado como parâmetro. (Se estiver a utilizar a versão 4.0 ou superior do SWI Prolog, essa lista aparecerá numa nova janela — ver figura 1.2.)

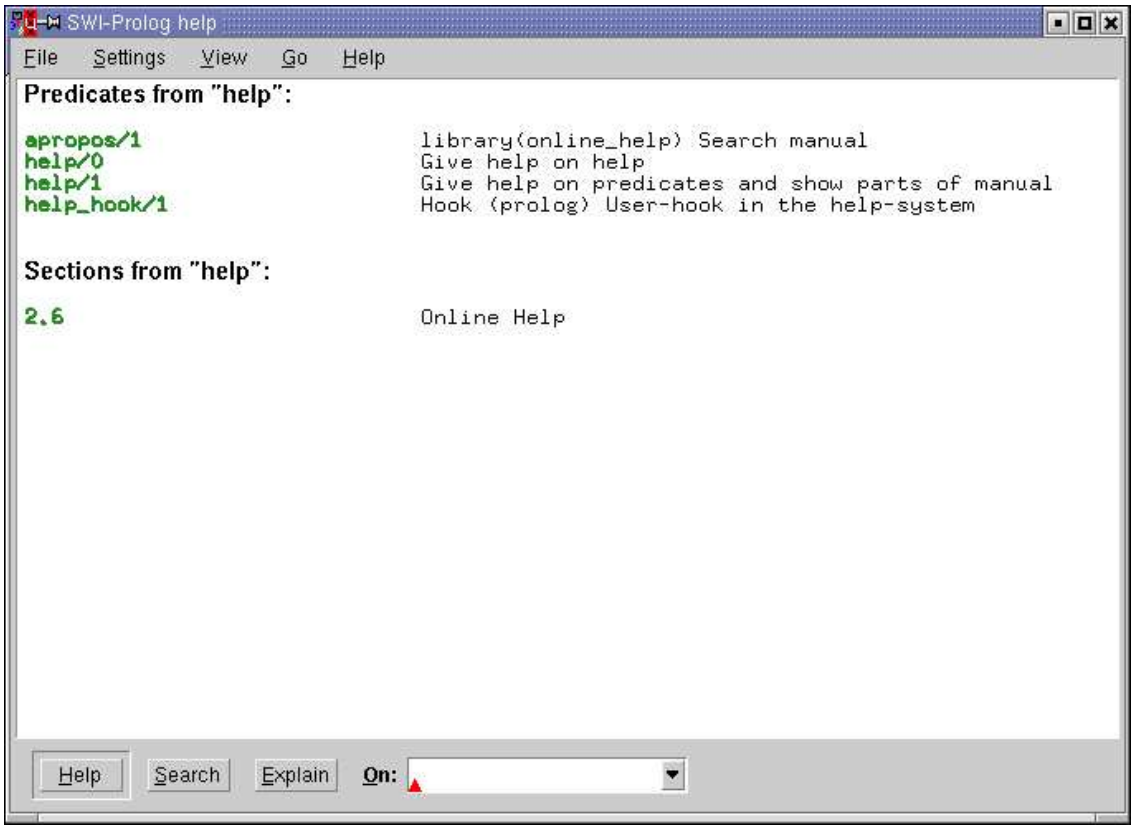

Figura 1.2: Janela de *Help* do SWI-Prolog

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 secção 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 Factos, *queries* e regras

1. Escreva os seguintes factos (pode utilizar "`[user].`" ou então escrever os factos num ficheiro 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 factos 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 factos à Base de Conhecimento (se anteriormente utilizou "`[user].`" é agora uma boa altura para começar a utilizar ficheiros):

```
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 ficheiro).