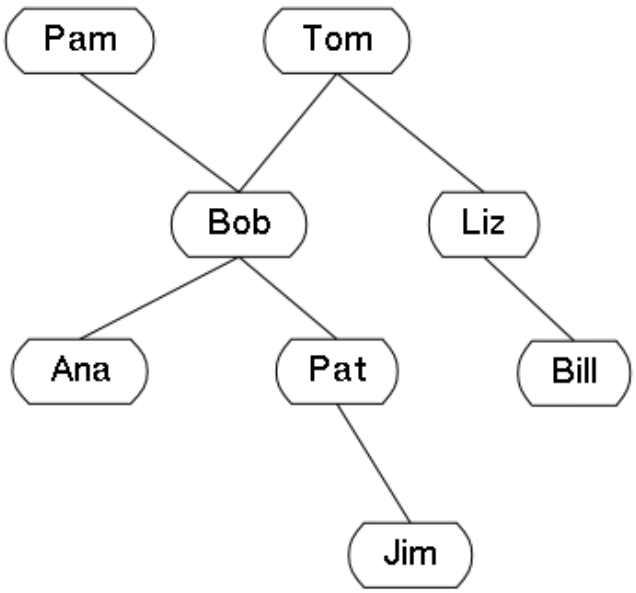
Universidade Federal de Santa Catarina **DAS / PPGEAS**
Fundamentos de Matemática Discreta para Controle e Automação
Professor: Jomi F. Hübner

Exercício de Prolog: **Genealogia**

Sumário

1 Objetivo

Fazer um primeiro experimento em Prolog para realizar provas sobre relações de parentesco sobre as pessoas da figura abaixo (as linhas representam as relações de pai–filho).



2 Parte I: Fatos e Perguntas

2.1 Definição das relações de parentesco

Assim como na lógica de predicados, um programa prolog permite estabelecer *relações entre objetos*. Neste exercício, estamos interessados em estabelecer relações de parentesco entre pessoas.[1]

A primeira tarefa é escrever um programa Prolog onde estejam definidas as relações dadas na figura acima. Chamaremos está relação de "genitor" (por exemplo "Tom é genitor de Bob") e a representaremos por um predicado de aridade 2 (por exemplo `genitor`(tom, bob)). Temos assim a primeira versão do programa prolog:

```
genitor(pam,bob). % Pam é mãe de Bob
genitor(tom,bob). % Tom é pai de Bob
genitor(tom,liz).

genitor(bob,ana).
genitor(bob,pat).

genitor(liz,bill).

genitor(pat,jim).
```

Edite o programa Prolog acima em um editor de textos (notepad, vi, emacs, ...) e salve-o em um arquivo chamado `familia.pl`.

Depois de salvo, carregue o programa no SWI-Prolog com os seguintes comandos (comandos que iniciam com $ devem ser digitados em um terminal de comandos; comandos que iniciam com `?-` devem ser digitados no Prolog):

```
$ swipl
?- consult('familia').
```

2.2 Perguntas no Prolog

O programa acima é formado por *fatos* sobre as relações de parentesco (fatos são premissas representadas por átomos). O prolog permite fazer vários tipos de *perguntas* sobre estes fatos. Para cada pergunta, é mostrado "yes" na tela caso se consiga uma prova para a pergunta ou "no" caso contrário. Por exemplo, para saber se existe uma prova de que Pam é genitora de Bob, basta digitar a pergunta

```
?- genitor(pam,bob).

Yes.
```

A resposta será "Yes", pois há uma prova trivial de que Pam é genitora de Bob (é uma das premissas).

Um recurso importante do Prolog é fazer perguntas com variáveis, neste caso é encontrado um, ou vários, valor(es) para a variável que satisfazem a pergunta (as variáveis sempre iniciam com letra maiúscula no Prolog). Por exemplo, para saber se Tom tem pelo menos um filho ($\exists_x$ `genitor`(tom, x)):

```
?- genitor(tom,X).

X = bob<Tecle Enter aqui>

Yes
```

[1] A linguagem Prolog se distingue de outras linguagem justamente por definir as relações entre os objetos e não os procedimentos ou rotinas de processamento de dados como ocorre, por exemplo, nas linguagem C e Pascal.

e para saber o nome de todos os seus filhos:

```
?- genitor(tom,X).

X = bob<Tecle ; aqui>

X = liz<Tecle ; aqui>

No
```

Também é possível utilizar *conjunções* nas perguntas (a vírgula representa o $\wedge$ da lógica), por exemplo para encontrar dois genitores do Bob:

```
?- genitor(P, bob), genitor(M, bob).

P = pam
M = pam ;

P = pam
M = tom ;

P = tom
M = pam ;

P = tom
M = tom ;

No
```

O problema desta consulta é que P e M podem ter o mesmo valor. Poderíamos então melhorar a consulta para

```
?- genitor(P, bob), genitor(M, bob), P \= M.

P = pam
M = tom ;

P = tom
M = pam ;

No
```

(o `\=` significa não unifica).

2.3 Definição do sexo das pessoas

Para saber os pais do Bob (o nome da mãe e do pai), precisamos saber o sexo dos genitores. Temos assim a segunda versão do programa:

```
genitor(pam,bob).
genitor(tom,bob).
genitor(tom,liz).

genitor(bob,ana).
genitor(bob,pat).
```

```
genitor(liz,bill).

genitor(pat,jim).

mulher(pam).
mulher(liz).
mulher(pat).
mulher(ana).
homem(tom).
homem(bob).
homem(jim).
homem(bill).
```

Altere o programa anterior incluindo os fatos sobre o sexo das pessoas, salve e carregue novamente no Prolog. Agora podemos fazer a consulta:

```
?- consult('familia').
?- genitor(P,bob), genitor(M,bob), homem(P), mulher(M).

P = tom
M = pam

Yes
```

2.4 Exercícios

- Qual a resposta para as seguintes consultas
  1. genitor(X,jim).
  2. genitor(jim,X).
  3. genitor(pam,X), genitor(X,pat).
  4. genitor(pam,X), genitor(X,Y), genitor(Y, jim).
- Faça uma pergunta para saber
  1. a mãe de Jim,
  2. o avô materno de Jim,
  3. o bisavô materno de Jim,
  4. a bisavó materna de Jim,
  5. o pai de Ana e Pat,
  6. o irmão de Bob,
  7. a irmã de Pat.

3 Parte II: Regras

3.1 Definição de inferências

Para estabelecer a relação de "é filho" no nosso programa, poderíamos incluir novos fatos como filho(bob,tom), filha(liz,tom), etc. Contudo esses fatos são decorrentes (conseqüência) dos fatos já conhecidos e seria redundante enumerá-los no programa. Para resolver este problema, podemos incluir uma *regra* no programa Prolog equivalente à seguinte fórmula

$$\forall_x \forall_y \texttt{genitor}(x, y) \wedge \texttt{homem}(y) \rightarrow \texttt{filho}(y, x)$$

em Prolog:

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

Como se pode perceber, o conectivo $\rightarrow$ transformou-se em `:-`, a direção da implicação foi invertida e os quantificadores omitidos.
Agora podemos fazer as seguintes perguntas:

```
?- filho(bob, tom).

Yes

?- filho(bob, X).

X = tom

Yes
```

3.2 Exercícios

Defina regras Prolog que representam as seguintes relações de parentesco:

1. avô/avó
2. irmão/irmã
3. tio/tia
4. primo/prima

3.3 Resumo

Um programa Prolog é uma seqüência de fatos e regras em notação clausal com no máximo um literal positivo (cláusulas de Horn). Estas cláusulas são consideradas como premissas e, para cada pergunta feita, o Prolog realiza uma prova (por resolução) para verificar se a pergunta é conseqüência das premissas.

4 Bibliografia

- BRATKO, Ivan. **Prolog programming for artificial intelligence**. 2.ed. Wokingham : Addison-Wesley, 1990.
- STERLING, Leon; SHAPIRO, Ehud. **The art of Prolog**: advanced programming techniques. 2.ed. Cambridge : MIT, 1994. (p.411-478)
- CASANOVA, Marco Antonio, et al. **Programação em lógica e a linguagem PROLOG**. São Paulo : E. Blucher, c1987.