Aula prática I

Alexsandro Santos Soares
prof.asoares@gmail.com

Programação Lógica
Faculdade de Computação
Universidade Federal de Uberlândia

14 de março de 2011

1 Objetivos

Esta aula tem por objetivos:

- Apresentar o ambiente de programação SWI-Prolog.
- Montar bases de conhecimento e realizar consultas a respeito das mesmas.

2 Comandos do Prolog

O texto de um programa em Prolog é normalmente criado num arquivo, ou conjunto de arquivos, usando um dos editores de texto standard , como por exemplo, o bloco de notas (notepad) do Windows. Embora em alguns momentos deste texto ênfase seja dada para o ambiente Windows nada impede que o mesmo procedimento seja adotado para o SWI-Prolog executado em Linux.

O interpretador de Prolog pode depois ser instruído a ler os programas ou bases de conhecimento destes arquivos - a este processo chamamos consultar. Embora esses arquivos possam ter uma extensão arbitrária, optamos por usar a extensão .pl, que é a extensão por defeito usada pelo SICStus (uma outra implementação de Prolog) e pelo SWI-Prolog.

2.1 Iniciar o interpretador de Prolog

Existe um ícone para o interpretador SWI-Prolog no Windows. Depois de clicar duas vezes nesse ícone , o interpretador de Prolog fica à espera de "ordens" , aparecendo o prompt

```
?-
```

2.2 Consultar bases de conhecimento

Por exemplo, para consultar uma base de conhecimento que se encontra no arquivo teste.pl, no diretório `c:\prolog\` (admitindo que haja este caminho no seu computador. Pode ser que não exista) digitamos o seguinte:

```
?-['c://prolog/teste.pl'].
```

No ambiente Windows você dispõe de um menu com a opção "consult" para executar o procedimento descrito anteriormente.

2.3 Mais Comandos

Alguns comandos uteis:

halt Sair do interpretador de prolog.

listing Mostrar a informação carregada em memória. Experimentar após o consult de um arquivo.

3 Relacionamentos familiares

Prolog é uma linguagem adequada para processamento simbólico. É indicada para resolver problemas que envolvam objetos e seus relacionamentos, por exemplo, o relacionamento familiar.

Em Prolog para especificar que Carlos é pai de Ana escreve-se:

```
pai(carlos,ana).
```

Continuando com o exemplo da família deve-se especificar que Carlos tem mais uma filha: Juliana. Assim temos que:

```
pai(carlos,ana).
pai(carlos,juliana).
```

Após comunicar este programa ao sistema Prolog podemos fazer com que ele nos responda a perguntas. Isto se faz por colocar o sistema num modo em que isto é possível.

Em geral o símbolo `?-`, que é um prompt, representa este estado. Por exemplo, podemos perguntar: Carlos é pai de Ana? Em Prolog isto se faz por escrever:

```
?- pai(carlos,ana).
```

Perguntas mais interessantes podem ser formuladas. Exemplo: Quem é filho de Carlos?

```
?- pai(carlos,X).
```

Desta vez a resposta não será `true` ou `false`. Prolog nos dirá qual é o valor de `X` que tornará a cláusula verdadeira.

```
X=ana
```

Conforme especificado anteriormente, temos que Carlos tem mais de uma filha. Para dizer a Prolog que queremos outras soluções digitamos `;` na frente da resposta dada:

```
?- pai(carlos,X).
X=ana;
X=juliana;
false
```

Ex. 1 Estender o programa Prolog para incluir os relacionamentos *mae*, *homem* e *mulher*.

Ex. 2 Escrever em Prolog as seguintes consultas:

- Quem é pai de Ana?
- Quem é pai de Quem?

Uma pergunta mais complicada: Quem é o avô de Ana?

Não foi dito ao Prolog nenhum relacionamento avô. A consulta tem que ser decomposta em dois passos:

- Quem é o pai de Ana? Assuma que é algum `Y`.
- Quem é o pai de `Y`? Assuma que é algum `X`.

Tal consulta composta é escrita em prolog da seguinte forma:

```
?- pai(Y,ana),pai(X,Y).
```

Esta consulta pode ser lida da seguinte forma: Encontre `X` e `Y` de tal forma que `pai(Y,ana)` **e** `pai(X,Y)` sejam satisfeitos.

Ex. 3 Escrever em Prolog as seguintes consultas:

- Quem são os netos de João?
- Juliana e Ana têm o mesmo pai?

Se ao invés de uma consulta quisermos algo mais definitivo, devemos criar uma regra. Definindo uma regra para a relação `avo`:

```
avo(X,Y) :-
   pai(X,Z),
   pai(Z,Y).
```

Esta regra pode ser lida da seguinte forma: para todo `X`, `Y` e `Z`, `X` é avô de `Y` se `X` é pai de `Z` e `Z` é pai de `Y`.

Ex. 4 Escreva regras para o relacionamentos `filho`, `filha`, `irmao`, `irmã`, `irmaos`, `tio`, `tia`, `primo`, `prima` e `avó`.

4 Mundo do Harry Potter

Observe a seguinte base de conhecimento:

```
elfo_domestico(dobby).

bruxo(hermione).
bruxo('McGonagall').
bruxo(rita_skeeter).

magico(X):- elfo_domestico(X).
magico(X):- feiticeiro(X).
magico(X):- bruxo(X).
```

Quais das seguintes consultas são satisfeitas? Onde relevante, dê todas as instanciações de variáveis que levam ao sucesso.

Ex. 5 `magico(elfo_domestico).`

Ex. 6 `feiticeiro(harry).`

Ex. 7 `magico(feiticeiro).`

Ex. 8 `magico('McGonagall').`

Ex. 9 `magico(Hermione).`

Desenhe a árvore de busca para a quinta consulta.

5 Rastreando predicados

Vamos introduzir uma das mais úteis ferramentas em Prolog: trace. Ele é um predicado já embutido no Prolog que modifica a maneira com a qual o Prolog executa: ele força o Prolog a avaliar as consultas passo a passo, indicando o que está fazendo em cada passo.

Prolog espera que você tecle `ENTER` antes que ele se mova para o próximo passo, tal que você possa ver exatamente o que está acontecendo. O predicado trace foi projetado para ser utilizado como uma ferramenta de depuração, porém, ele também é útil quando se está aprendendo Prolog: caminhar através do programas usando trace é um forma excelente de aprender como a busca de prova em Prolog funciona.

Considere a base de conhecimento abaixo:

```
f(a).
f(b).

g(a).
g(b).
```

```
h(b).

k(X) :- f(X),g(X),h(X).
```

Digite esta base de conhecimento em um arquivo de nome `prova.pl` e o consulte.

Já na linha de comando do Prolog, digite trace. e tecle enter:

```
 ?- trace.
true
```

A partir de agora, o Prolog está no modo de rastreamento e avaliará todas as consultas passo a passo. Por exemplo, se pusermos a consulta `k(X)` e teclarmos `ENTER` a cada vez que o Prolog vem com um `?`, obteremos algo como o seguinte:

```
[trace]  ?- k(X).
   Call: (6) k(_G360) ?
   Call: (7) f(_G360) ?
   Exit: (7) f(a) ?
   Call: (7) g(a) ?
   Exit: (7) g(a) ?
   Call: (7) h(a) ?
   Fail: (7) h(a) ?
   Redo: (7) f(_G360) ?
   Exit: (7) f(b) ?
   Call: (7) g(b) ?
   Exit: (7) g(b) ?
   Call: (7) h(b) ?
   Exit: (7) h(b) ?
   Exit: (6) k(b) ?
X = b.
```

Estude isto cuidadosamente. Isto é, tente fazer o mesmo e relacione isto à construção de uma árvore de prova (vista na última aula).

Para auxiliá-lo, note que a terceira linha é onde a variável na consulta é (erroneamente) instanciada para `a` e a linha marcada `Redo` é quando o Prolog nota que tomou o caminho errado e retrocede para instanciar a variável para `b`.

Enquanto estiver aprendendo Prolog, use o trace. Quando desejar sair do modo de rastreamento, digite notrace. e tecle `ENTER`. Isto o levará para o modo de depuração, sobre o qual falaremos em outra aula. Para sair deste modo e voltar ao modo normal, digite nodebug.:

```
[trace]  ?- notrace.
true.

[debug]  ?- nodebug.
true.

?-
```

6 Minigramática

Aqui está um pequeno dicionário e uma minigramática com somente uma regra que define a sentença como consistindo de cinco palavras: um artigo, um nome, um verbo e, novamente, um artigo e um nome.

```
palavra(artigo,um).
palavra(artigo,qualquer).
palavra(nome,criminoso).
palavra(nome,'mac lanche feliz').
palavra(verbo,come).
palavra(verbo,adora).

sentenca(Palavra1,Palavra2,Palavra3,Palavra4,Palavra5) :-
   palavra(artigo,Palavra1),
   palavra(nome,Palavra2),
   palavra(verbo,Palavra3),
   palavra(artigo,Palavra4),
   palavra(nome,Palavra5).
```

Ex. 10 Qual consulta deve-se colocar a fim de encontrar quais sentenças a gramática pode gerar?

Ex. 11 Liste todas as sentenças que esta gramática pode gerar na mesma ordem em que o Prolog as geraria.

Assegure-se que você compreende o porquê do Prolog gerá-las nesta ordem.

7 Alunos

Escreva os seguintes fatos (especificados num arquivo de nome aluno.pl):

```
aluno(joao,poo).
aluno(pedro,poo).
aluno(maria,pl).
aluno(rui,pl).
aluno(manuel,pl).
aluno(pedro,pl).
aluno(rui,ed1).
```

Ex. 12 Verifique que os fatos estão presentes na Base de Conhecimento (utilize o predicado `listing`).

Ex. 13 Escreva uma consulta que verifique se joao é aluno de `pl`.

Ex. 14 Escreva uma consulta que verifique se rui é aluno de `poo`.

Ex. 15 Escreva uma consulta que verifique se joao e maria são ambos alunos de `ed1`. joao e maria são ambos alunos de `ed1` se joao for aluno de `ed1` e maria for aluna de `ed1`.

Ex. 16 Escreva uma consulta que permita saber quem é aluno de `pl`.

Ex. 17 Escreva uma consulta que permita saber as disciplinas em que rui é aluno.

Adicione os seguintes fatos à Base de Conhecimento anterior:

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

Ex. 18 Sabendo que a aluno A faz a disciplina D se A é aluno de D e A estuda, escreva uma consulta que lhe permita saber se maria faz `pl`.

Ex. 19 Experimente agora a segunte consulta:

```
?- aluno(X,pl), estuda(X).
```

O que lhe permite esta consulta saber?

Ex. 20 Utilizando a consulta anterior, acrescente à Base de Conhecimento o predicado `fazpl/1` e escreva uma consulta que lhe permita saber quem faz `pl`.

Não esquecer que:

1. Tudo o que se segue a um `%` é considerado pelo interpretador de prolog como um comentário.
2. As variáveis em prolog começam por maiúsculas.
3. No decurso de uma computação, uma variável pode ser substituída por um objeto concreto. Dizemos então que a variável foi instanciada.