Lógica Matemática – Aula 11
Programação Lógica

Prof. Juliano Henrique Foleiss, M. Sc.

1 PROLOG

Ao contrário de linguagens de programação procedurais, como `C` e `C++`, que consistem em instruções de como executar determinada tarefa, as **linguagens de programação lógicas** pertencem a um paradigma de programação denominado **programação declarativa**. Neste paradigma, os programas são descritos em forma de declarações que são processadas por um motor de execução capaz de realizar deduções. No caso das lingaugens de programação lógica, o motor de execução é uma implementação de um subconjunto das regras da lógica de predicados, baseado em regras de inferência como *modus ponens* e resolução. Um exemplo de linguagem de programação lógica é a linguagem `PROLOG`, abreviação de *PROgramming in LOGic*. O conjunto de declarações que formam um programa PROLOG é denominado **banco de dados PROLOG**. Este banco de dados é formado por **fatos** e **regras**.

1.1 Fatos

Os fatos `PROLOG` permitem a definição de predicados por meio da declaração de quais itens satisfazem os predicados. Por exemplo, vamos criar um programa que descreva as cadeias alimentares de uma região ecológica. Podemos começar com um predicado de duas variáveis, `come`, que indica quais animais se alimentam dos outros. Assim, para descrever o predicado basta fornecer os pares dos elementos do domínio que tornam o predicado verdadeiro. Portanto, alguns fatos poderiam ser:

```
come(urso, peixe).
come(urso, raposa).
come(veado, grama).
```

Neste exemplo, `urso, peixe, raposa, veado e grama` são constantes e representam elementos específicos do conjunto universo. Veja que estes elementos são iniciados por letra minúscula, indicando que são constantes, e não variáveis. Note também que não há outra forma de enunciar os elementos do conjunto universo, exceto pela declaração de predicados. Além disto, é de responsabilidade do usuário manter a semântica correta dos predicados. Por exemplo, `come(urso, peixe)` pode ser usado para representar que ursos comem peixe, e não o contrário!

Para incrementar nossa base, podemos inserir outros fatos, como, por exemplo, os predicados `animal` e `planta`:

```
animal(urso).
animal(peixe).
animal(raposa).
animal(veado).
planta(grama).
```

Além de inserir fatos no banco de dados podemos também consultar o banco. Por exemplo a pergunta

```
?- animal(urso).
```

questiona o motor de inferência do PROLOG se urso satisfaz o predicado animal. Como animal(urso) está na base, o programa responde yes. Outros exemplos de consultas: come(veado,grama) responderia yes e come(urso,coelho) responderia no. Perguntas também podem conter variáveis, por exemplo:

```
?- come(urso,X).
```

questiona o PROLOG sobre quais valores para X fazem com que o predicado come seja verdadeiro. O PROLOG responde com uma lista de todos os valores de X que satisfazem o predicado. Neste caso, peixe e raposa. Note que, as variáveis são iniciadas por letras maiúsculas. Podemos também utilizar os conectivos lógicos conjunção (,), disjunção (;) e negação (**not()**) para criar consultas mais elaboradas. Por exemplo, a consulta

```
?- come(X,Y), planta(Y).
```

retorna X que come Y tal que Y é uma planta. Neste caso, X = veado e Y = grama.

1.2 Regras

Outra forma de declarar predicados em PROLOG é definindo **regras**. Uma regra é uma declaração de um predicado por meio de uma expressão condicional. Por exemplo, podemos utilizar um predicado para definir um predicado para presa:

```
presa(X) :- come(Y,X), animal(X).
```

Neste caso, estamos definindo que X é presa se X é um animal que é comido. Neste caso, a consulta presa(X) retornaria peixe e raposa.

1.3 Cláusulas de Horn e Resolução

É interessante notar que os fatos e regras do PROLOG possuem uma relação direta com a lógica de predicados que temos estudado até agora. Por exemplo, os fatos apresentados nas seções anteriores podem ser representados pelas afirmações:

$$C(u,p)$$
$$C(u,r)$$
$$C(v,g)$$
$$A(u)$$
$$A(p)$$
$$A(r)$$
$$A(v)$$
$$P(g)$$

e a regra por $C(y,x) \wedge A(x) \rightarrow Pr(x)$. Apesar dos quantificadores universais não serem apresentados explicitamente na regra, a linguagem trata a regra como se estivesse universalmente quantificada, ou seja $(\forall x)(\forall y)(C(y,x) \wedge A(x) \rightarrow Pr(x))$ e usa, repetidamente, a **instanciação universal** (que estudaremos mais adiante) para retirar os quantificadores e permitir às variaveis que assumam todos os valores do conjunto universo.

Cláusulas de Horn e o Motor de Inferência do PROLOG

Tanto os fatos quanto as regras são exemplos de **cláusulas de Horn**. Uma cláusula de Horn é uma exppressão lógica composta de predicados ou da negação de predicados, com variáveis ou constantes como argumentos, combinados por meio de disjunções, de tal forma que no máximo um predicado não seja negado. Portanto, o fato $C(v,g)$ é uma cláusula de Horn, uma vez que consiste em pelo menos um único predicado não-negado. Outra cláusula de Horn é a expressão

$\neg(C(y,x))\vee\neg(A(x))\vee Pr(x)$, que consiste em três predicados conectados por disjunções com apenas $Pr(x)$ não-negado. Pelas leis de DeMorgan, ela é equivalente a $\neg(C(y,x) \wedge A(x)) \vee Pr(x)$ que é equivalente a $C(y,x) \wedge A(x) \rightarrow Pr(x)$, que é exatamente a regra apresentada anteriormente.

A regra de inferência utilizada pelo `PROLOG` é denominada **resolução**, que já estudamos anteriormente. Duas cláusulas de Horn em um banco de dados `PROLOG` podem ser resolvidas em uma nova cláusula de Horn se uma delas tiver um predicado não-negado que corresponda a um predicado negado na outra. Esta nova cláusula elimina o termo correspondente e fica disponível para o uso em resposta à consultas realizadas. Por exemplo,

$$A(a)$$
$$\neg(A(a)) \vee B(b)$$

por resolução nos dá $B(b)$. Ou seja, a partir de $A(a), \neg(A(a)) \vee B(b)$, que é equivalente a $A(a), A(a) \rightarrow B(b)$ o motor de inferência do `PROLOG` infere $B(b)$. Isto é equivalente à aplicação de *modus ponens*, que é um caso especial da resolução. Ao aplicar a resolução, as variáveis são consideradas correspondentes, ou seja, são unificadas, com qualquer símbolo constante, que é uma aplicação da instanciação universal. Quando uma nova cláusula é obtida, as variáveis são substituídas pelas constantes previamente associadas à mesma variável. Desta forma, quando a pergunta "quais `X` são presas" é feita, o motor de inferência do `PROLOG` busca no banco uma regra que tenha o predicado desejado como consequente, neste caso $Pr(x)$. Encontra $\neg(C(y,x)) \vee \neg(A(x)) \vee Pr(x)$ e procura no banco cláusulas alguma cláusula que pode ser resolvida com esta. A primeira delas é o fato $C(u,p)$. Assim, aplicando a resolução entre $\neg(C(y,x)) \vee \neg(A(x)) \vee Pr(x)$ e $C(u,p)$ resulta em $\neg(A(p)) \vee Pr(p)$ (veja que p substituiu x). Usando esta nova cláusula, ela pode ser resolvida com o fato $A(p)$ para obter $Pr(p)$. Após obter todas as resoluções possíveis do fato $C(u,p)$, o `PROLOG` volta atrás, procurando outra cláusula que possa ser resolvida com a regra da consulta, encontrando $C(u,r)$. O `PROLOG` continua a utilizar este algoritmo até que não sejam possíveis mais resoluções.

2 Exercícios

1. Suponha os predicados `mae(M,Y)` e `pai(P,X)`, que indicam que `M` é mãe de `Y` e `P` é pai de `X`, respectivamente. Escreva uma regra `PROLOG` que defina o predicado `irmao(X,Y)`, que indica que `X` é irmão de `Y`, ou seja, `X` e `Y` têm o mesmo pai e mãe.

2. Suponha os predicados `mae(M,Y)` e `pai(P,X)`, que indicam que `M` é mãe de `Y` e `P` é pai de `X`, respectivamente. Escreva uma regra `PROLOG` que defina o predicado `avô(A,N)`, que indica que `A` é avô de `N`.

3. Um banco de dados Prolog contém os dados a seguir, onde `patrao(X,Y)` significa que "`X` é patrão de `Y`" e `supervisor(X,Y)` indica que "`X` é supervisor de `Y`".

```
patrao(miguel,joana).
patrao(judite,miguel).
patrao(anita,judite).
patrao(judite,kim).
patrao(kim,henrique).
patrao(anita,samuel).
patrao(henrique,jeferson).
patrao(miguel,hamal).

supervisor(x,y) :- patrao(x,y).
supervisor(x,y) :- patrao(x,z), supervisor(z,y).
```

Encontre os resultados das seguintes consultas:

a. `patrao(X,samuel)`
b. `patrao(judite,X)`
c. `supervisor(anita,x)`
d. `supervisor(miguel,x)`
e. `supervisor(X,henrique)`

4. Construa um banco de dados `PROLOG` que fornece informações sobre estados e suas capitais. Algumas cidades são grandes, outras pequenas. Os estados são da região sul, norte, nordeste, centro-oeste e sudeste.

a. Escreva uma consulta para encontrar todas as capitais pequenas.
b. Escreva uma consulta para encontrar todos os estados que tem capitais pequenas.
c. Escreva uma consulta para encontrar todos os estados na região nordeste com capitais grandes.
d. Formule uma regra para definir cidades cosmopolitas como sendo capitais grandes dos estados nas regiões sul e sudeste.
e. Escreva uma consulta para encontrar todas as cidades cosmopolitas.

5. Um banco de dados contém os dados a seguir, onde `instrutor(P,D)` indica que o professor `P` é instrutor de da disciplina `D` e `matriculado(A,D)` indica que o aluno `A` está matriculado na disciplina `D`.

```
instrutor(chan,math273).
instrutor(patel,ee222).
instrutor(grossman,cs301).
matriculado(kevin,math273).
matriculado(joana,ee222).
matriculado(joana,cs301).
matriculado(kiko,math273).
matriculado(kiko,cs301).
```

a. Formule a regra `eProfessorDe(P,A)` que indica que o professor `P` é professor do aluno `A`.
b. Realize a consulta `instrutor(patel,cs301)`.
C. Realize a consulta `instrutor(grossman,X)`.
d. Realize a consulta `eProfessorDe(patel,X)`.
e. Realize a consulta `eProfessorDe(X,kiko)`.
f. Realize a consulta `eProfessorDe(joana,X)`.

3 Bibliografia

`Gersting, Judith`. Fundamentos Matemáticos para Ciência da Computação – Um Tratamento Moderno da Matemática Discreta. Ed. LTC. Quinta edição. Rio de Janeiro, 2012.

`Rosen, Kenneth`. Discrete Mathematics and its Applications. Ed. McGraw Hill. Sétima Edição. Nova Iorque, 2012.