AULA 12 – PARADIGMAS DE PROGRAMAÇÃO

22/04/2013

PROFESSOR GLAUBER FERREIRA CINTRA
SEMESTRE 2012.2 – ENGENHARIA DE COMPUTAÇÃO IFCE

PROLOG

A linguagem Prolog foi desenvolvida no início dos anos 70 por Russel, Colmarauer e outros pesquisadores da Universidade de Narselha (França) e da Universidade de Edinburgo (Escócia).

Essa linguagem é baseada no paradigma lógico e é mais utilizada pela comunidade de inteligência artificial. Tal linguagem utiliza o método de Robinson (1965) para verificar a **satisfabilidade** das cláusulas de Horn.

Uma cláusula de Horn é uma fórmula da lógica de predicados que expressa uma implicação na qual a premissa é uma conjunção de predicados e o consequente é *um único predicado*. Além disso, todas as variáveis estão quantificadas com o *quantificador universal*.

$$\forall x_1 \forall x_1 \forall x_1 \ldots [\, P_1 \wedge P_2 \wedge P_3 \ldots \wedge P_k \rightarrow Q]$$

Um programa em Prolog é uma coleção de cláusulas de Horn. Em Prolog todas as variáveis começam com **letra maiúscula** e o nome dos predicados em **letras minúsculas**. O conseqüente é colocado a esquerda da premissa. A quantificação das variáveis é omitida e toda cláusula termina com **.** (ponto final). A conjunção é denotada por **:-** (dois pontos e hífen). Os argumentos de cada predicado são colocados entre parênteses e são separados por **;** (ponto e vírgula) e pode ser uma variável ou um literal, uma constante como número, caractere, lista ou string (entre aspas simples) etc.

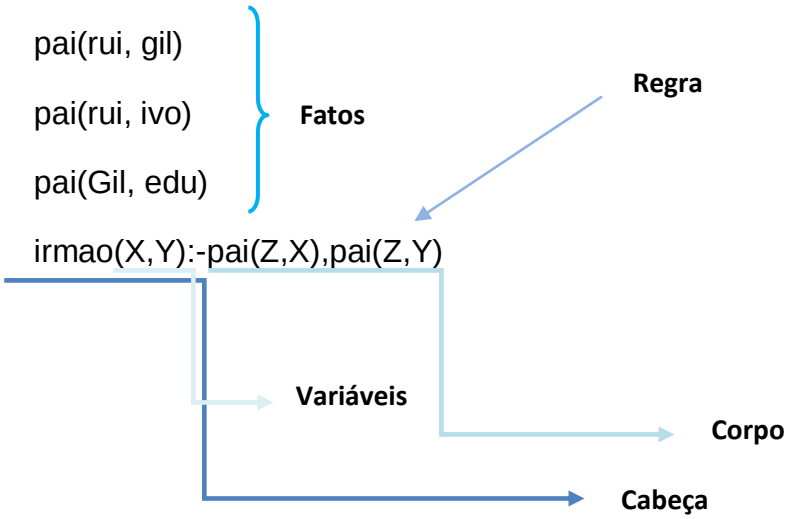
Exemplo:

q(...):- p1(...),p2(...),p3(...),...,pk(...).

Em Prolog todas as cláusulas do programa são consideradas verdadeiras, aquilo que pode ser provado a partir das cláusulas do programa também são consideradas verdadeiras. Tudo o mais é considerado falso (*Hipótese de mundo fechado*).

O consequente de uma cláusula é chamado de **cabeça** e premissa é chamada de **corpo** da cláusula. Uma cláusula *sem corpo* é chamada de **fato**. As demais cláusulas são chamadas de *regras*.

Exemplo:



Em Prolog a computação tem início quando submetemos uma *meta* ao interpretador. Uma meta é um predicado (*meta simples*), uma conjunção ou uma disjunção de predicados (*meta composta*). Ao receber uma meta, o interpretador verifica se ela sucede (*é verdadeira*) ou falha (*é falsa*). Veremos agora como é feita a verificação de uma *meta simples*.

Ao receber a meta o interpretador verifica se ela casa com alguma cláusula do programa. Uma meta casa com uma cláusula do programa. Uma meta *casa* com uma cláusula se o predicado da meta é **igual** ao predicado da cabeça da cláusula, e os argumentos da meta *casam* com os argumentos da cabeça da cláusula.

Um argumento da mesma meta casa com o argumento correspondente da cabeça da cláusula se:

1. Ambas são variáveis. Nesse caso, as duas variáveis são unificadas, ou seja, passam a compartilhar a mesma região de memória;

   ou

2. Um argumento é variável e outro não. Nesse caso, a variável é instanciada (particularizada) com o valor do outro argumento;

   ou

3. Ambas não são variáveis, mas são iguais.

Se a meta não casar com nenhuma cláusula, ela **falha**. Se a meta casa com um fato, ela **sucede**.

Se a meta casar com uma regra será preciso verificar o corpo da regra. Se o corpo for verdadeiro, a meta **sucede**. Se um predicado do corpo for verdadeiro, a meta **sucede**. Se um predicado do corpo da regra **falha**, o interpretador *volta* para a *última meta* que sucedeu e tenta sucedê-la de outra maneira.

Se não for mais possível retroceder a meta **falha**.

Execute no interpretador Prolog o seguinte código:

```
pai(rui, gil)

pai(rui, ivo)

pai(Gil, edu)

irmao(X,Y):-pai(Z,X),pai(Z,Y)
```