Programação em Lógica

Profa. Juliana Vizzotto

juvizzotto@inf.ufsm.br

Disciplina de Paradigmas de Programação

Roteiro

- Introdução
- Programando em Prolog

Introdução

Prolog

- Paradigma de Programação baseado no cálculo da Lógica de Predicados
- Programação declarativa: especifica "o que" deve ser solucionado!
- Visão de *dedução* em lógica como processo *computacional*
- Idéia básica: um algoritmo é constituído por dois elementos disjuntos: a **lógica** e o **controle**
- O componente lógico corresponde à definição do que deve ser solucionado.
- O componente de controle estabelece como a solução pode ser obtida.
- O programador é somente responsável pela definição do componente lógico

O Papel do Programador

Deve-se definir uma base de conhecimento, através da definição de fatos e regras, de onde pode-se extrair conhecimento implícito.

O Papel do Programador

Prolog tem algumas similaridades com linguagens funcionais:

- Um programa funcional consiste de uma série de definições de funções
- Um programa em lógica consiste de uma série de definições de **relações**
- Ambos trabalham fortemente com definições recursivas
- A principal diferença está na *máquina de execução*
- A engenharia de execução de uma linguagem funcional avalia uma expressão convertendo-a para um grafo acíclico e então reduzindo o grafo a uma forma normal que representa o valor computado.
- O ambiente de execução Prolog *deduz* uma resposta a partir das definições de relações

O Papel do Programador

Resumindo

- Um programa Prolog é uma coleção de **fatos** e **regras** para provar coisas
- Você não executa o programa!
- Você formula perguntas e o sistema de execução da linguagem usa o conjunto de fatos e regras para tentar responder as perguntas!

Programando em Prolog

Interpretador SWI-Prolog

- Define-se tudo em Prolog utilizando-se **fatos** e **regras**
- Um **fato** é uma sentença consistindo de um indentificador seguido por uma $n-$*tupla* de constantes
- Exemplo de fatos para trabalhar com um Grafo em Prolog:

```
edge(a,b).
edge(a,e).
edge(b,d).
edge(b,c).
edge(c,a).
edge(e,b).
```

Programando em Prolog

Fatos

- O identificador `edge` é o nome de uma relação ou *predicado*
- Diz-se, por exemplo, que a dupla `(c,a)` satisfaz o predicado `edge`
- Os identificadores `a, b, c, d,` e são valores de constantes abstratas representando nomes para os nodos do grafo.
- Desenhe este grafo.

Programando em Prolog

Regras

- Um **fato** diz que uma tupla satisfaz um predicado *incondicionalmente*
- Uma **regra** é uma sentença Prolog a qual representa sob quais condições uma tupla satisfaz um predicado.
- Um **fato** pode ser visto como um caso especial de regra, onde a condição é sempre satisfeita.

Programando em Prolog

Regras

- Um **fato** diz que uma tupla satisfaz um predicado *incondicionalmente*
- Uma **regra** é uma sentença Prolog a qual representa sob quais condições uma tupla satisfaz um predicado.
- Um **fato** pode ser visto como um caso especial de regra, onde a condição é sempre satisfeita.

Programando em Prolog

Regras

- Especificando propriedades mais interessantes para os grafos
- Caminho de tamanho dois entre dois nós:

```
tedge(Node1,Node2) :-
        edge(Node1,SomeNode),
        edge(SomeNode,Node2).
```

- Novo predicado `tedge`
- Lado esquerdo: como um fato, entretanto os parâmetros são *variáveis* ao invés de constantes.
- Lado direito: dois termos
- Vírgula significa um `E` lógico

Programando em Prolog

O par `(Node1,Node2)` satisfaz o predicado `tedge` se existe algum nodo `SomeNode`, tal que os pares `(Node1,SomeNode)` e `(SomeNode,Node2)` ambos satisfazem o predicado

Ambiente SWI-Prolog

- Sistema interativo
- Desenvolvido pelo Swedish Institute of Computer Science
- Pode ser executado pela linha de comando no shell ou pelo emacs.

Ambiente SWI-Prolog

- Digitar o comando `swipl`

```
juliana@amitaba:~/Teaching/Paradigmas/Aulas$ swipl
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Versior
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is f
and you are welcome to redistribute it under certain cc
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
```

- Para sair do ambiente:

```
?- halt.
```

Ambiente SWI-Prolog

- Todo o programa deve ser salvo com a extensão `.pl`
- Para carregar o programa:

```
?- ['p1.pl'].
% p1.pl compiled 0.01 sec, 1,260 bytes
true.
```

- Para listar o conteúdo da base:

```
?- listing.
```

Ambiente SWI-Prolog

- O interpretador SWI-Prolog é similar ao Ghci ou Hugs, pois um programa consiste de uma série de definições lidas a partir de um arquivo.
- O interpretador Haskell possibilita que o usuário compute uma **expressão** baseado nas definições de funções do programa.
- Da mesma maneira, o interpretador SWI-Prolog lê um programa e então possibilita que o usuário compute uma **pergunta** (*query*) baseado nas definições (fatos e regras).

Ambiente SWI-Prolog

- Uma *query* pergunta ao sistema se existem valores, os quais satisfazem um predicado particular.
- Por exemplo:

```
?- edge(a,b).
true .
```

- Ou

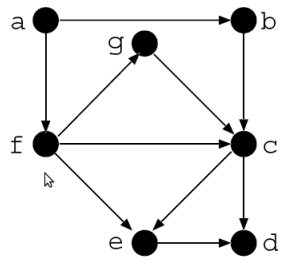
```
?- edge(a,b),edge(a,c).
false.
```

- Ou ainda

```
?- tedge(a,c).
true .
```

Exercício

- Escreva um programa Prolog que descreva um programa Prolog com a estrutura abaixo. Também inclua no programa um predicado que defina a relação *caminho de tamanho 3*. Defina ele de duas maneiras: primeiro somente referenciando o predicado edge e então posteriormente fazendo uso do predicado `tedge`.



Ambiente SWI-Prolog

- Uma *query* pergunta ao sistema se existem valores, os quais satisfazem um predicado particular.
- Por exemplo:

  ```
  ?- tredge(a,d).
  ```

- Ou

  ```
  ?- tredge(f,X).
  ```

- Ou ainda

  ```
  ?- tredge(X,Y).
  ```