

Linguagens de Programação Lógicas

Prof. Lucas Ismaily

Sumário

- Introdução
- Prolog
 - Fatos
 - Regras
 - Consultas
 - Exemplos

Introdução

- LP lógicas também são chamadas de **declarativas**
- Programas são expressos através de lógica simbólica
- Os resultados são produzidos através de um processo de inferência lógica
- **Declarativa** ao invés de **procedural**:
 - Especificamos apenas o que caracteriza um resultado, não o procedimento para obtê-lo

Introdução

- LP lógica utiliza uma forma de lógica simbólica como uma linguagem de programação
- A única linguagem lógica amplamente utilizada é **Prolog**

Proposições

- Objetos nas proposições de programação lógica são representados por termos simples: constantes ou variáveis
- **Constantes**: um símbolo que representa um objeto
- **Variável**: um símbolo que pode representar diferentes objetos em diferentes momentos
 - Diferente das variáveis em linguagens imperativas
 - Semelhante ao significado da matemática

Prolog

Breve história

- Primeira versão:
 - Início dos anos 70
 - Alain Colmerauer, Universid. de Marseilles, França
 - Como uma ferramenta para programação em logica

Áreas de aplicação

- Amplamente utilizada em
 - Aplicações de IA
 - Sistemas especialistas
- Versões mais modernas têm sido utilizadas para
 - Processamento de linguagem natural
 - Representação de conhecimento

Linguagem declarativa

- Significa que
 - O programador
 - Declara **fatos**
 - Define **regras** para raciocinar sobre os fatos
 - Prolog usa dedução para
 - Determinar novos fatos com base nos fatos atuais
 - Decidir se um fato proposto (**objetivo**) pode ser derivado dos fatos conhecidos
(esta decisão é chamada **conclusão**)

Predicado

- Na lógica clássica, uma sentença declarativa (**fato**) é chamado de **predicado**
- Um predicado é decidível (verdadeiro ou falso)
- Exemplos:
 - O céu é vermelho
 - Maria gosta de sorvete

Primitivo X Composto

- Um predicado é uma sentença primitiva
 - Não pode ser dividido em sentenças menores
 - Pode ser representado por um único símbolo

P: o céu é vermelho

Q: maria gosta de sorvete
- Predicados podem ser combinados em sentenças compostas através de conectores lógicos

Conectores lógicos

- Negação
 $\neg(P)$
- Conjunção
 P, Q
- Disjunção
 $P; Q$

Implicação (**regra**)

- Uma relação “se.. então” entre predicados
 $p :- q$ significa
 - Se q é verdadeiro, então p é verdadeiro
 - q é a hipótese, e p é a conclusão

Programa Prolog

- O conhecimento do Prolog é um conjunto finito de fatos e regras que são fornecidas pelo programa
- Assim, um programa Prolog não é uma sequência de ações, mas uma lista de fatos e regras
- Depois de ler estes fatos e regras, o Prolog pode responder as consultas do usuário
 - Através da inferência de novos fatos

Exemplo

Linguagem natural:

Carro é divertido.

Lógica de predicados:

`divertido(carro) .`

A rosa é vermelha.

`vermelho(rosa) .`

Se carro é divertido,

Então João gosta de

carro.

`gosta(joao,carro) :- divertido(carro) .`

Fatos

- Um fato é uma relação entre objetos

`gosta(joao, maria) .`

`gosta(maria, joao) .`

`gosta(pedro, caes) .`

- Um fato também pode expressar uma propriedade do objeto

`verde(alfaca) .`


`mulher(silvia) .`

- Fatos e regras devem terminar com ‘.’
- Fatos de um mesmo predicado devem ser listados juntos

Exemplos de regras

```
gosta(cindy, Algo) :-  
    gosta(joao, Algo) .  
gosta(alice, Algo) :-  
    verde(Algo) .
```

Variáveis
começam com
maiúscula.



- Leia estas regras como:
 - “Para provar que cindy gosta de alguma coisa X, mostre que joao gosta de X.” Ou seja, cindy gosta de tudo que joao gosta.
 - “Para provar que alice gosta de alguma coisa, mostre que esta coisa é verde”. Ou seja, alice gosta de tudo que é verde.

Consultas

- Em linguagem natural, perguntamos:
`Joao gosta de Cindy?`
- Na sintaxe Prolog, escrevemos:
`gosta(joao, cindy) .`

Consultas com variáveis

- Podemos perguntar em linguagem natural:

O que joao gosta?

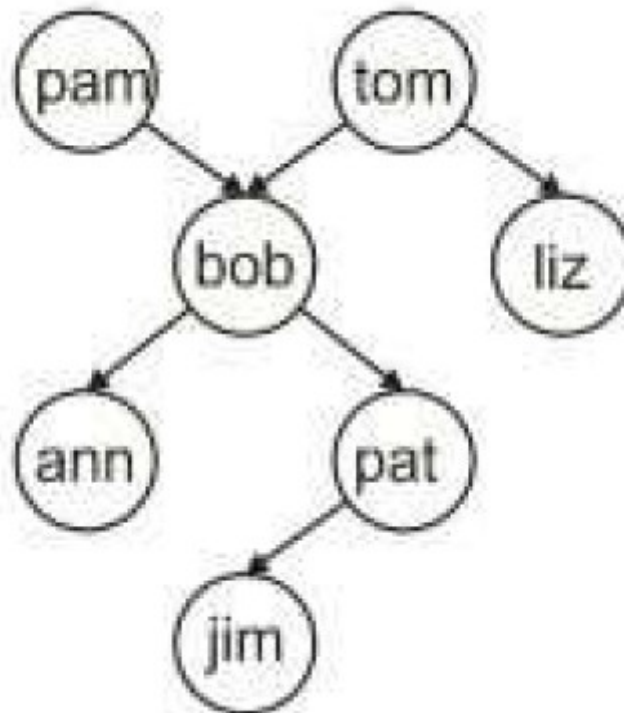
- Em sintaxe Prolog escrevemos:

`gosta(joao, X) .`

- Responde o 1º símbolo que satisfaz a regra
 - O usuário pode indicar que quer outro símbolo, digitando “;”
 - Para indicar que não quer outro, basta <ENTER>

Recursão

Você deseja representar uma hierarquia, como fazer?



Recursão

Você deseja representar uma hierarquia, como fazer?

```
genitor(pam, bob).  
genitor(tom, bob).  
genitor(tom, liz).  
genitor(bob, ann).  
genitor(bob, pat).  
genitor(pat, jim).
```

Recursão

Certo, mas como definir mãe e pai?

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

Recursão

Certo, mas como definir mãe e pai?

```
mae(X,Y) :- genitor(X,Y), mulher(X).  
avos(X,Z) :- genitor(X,Y), genitor(Y,Z).
```

Recursão

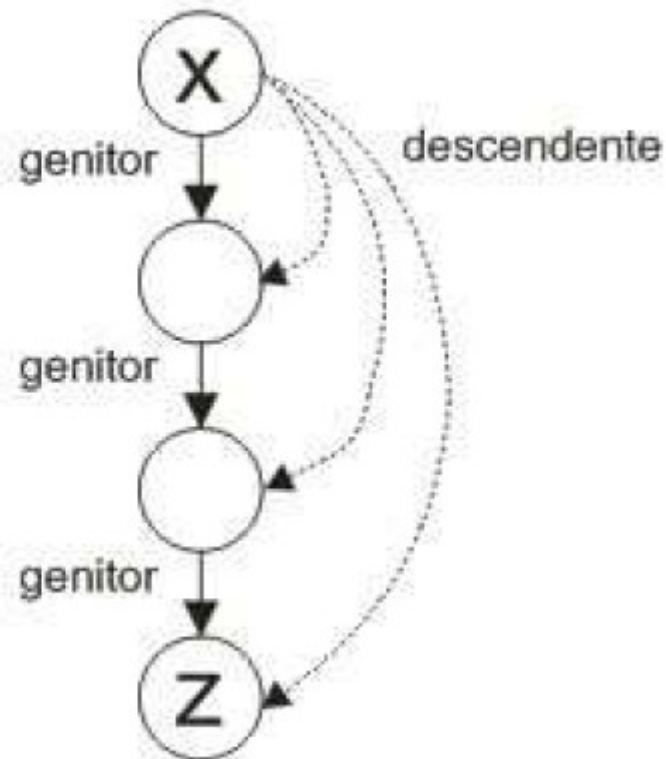
Certo, e descendente?

```
descendente(X,Z) :- genitor(X,Z).
```

Funciona?

Recursão

Essa é a estrutura



Recursão

E agora?

```
descendente(X,Z) :- genitor(X,Y), genitor(Y,Z).  
descendente(X,Z) :- genitor(X,Y), genitor(Y,W), genitor(W,Z).
```

Recursão

Agora sim.

```
descendente(X,Z) :- genitor(X,Z).  
descendente(X,Z) :- genitor(X,Y), descendente(Y,Z).
```

Se testar para 'pam', temos

```
13 ?- descendente(pam,X).  
X = bob ;  
X = ann ;  
X = pat ;  
X = jim ;
```

```
genitor(pam, bob).  
genitor(tom, bob).  
genitor(tom, liz).  
genitor(bob, ann).  
genitor(bob, pat).  
genitor(pat, jim).
```

Listas

- Existem as listas vazias e as não vazias
- As listas não vazias podem ser divididas em duas partes
 - Cabeça - primeiro elemento da lista
 - Cauda - o resto da lista
- Exemplo:
L = [1, 2, 3, 4, 5], nesse caso 1 é cabeça e [2, 3, 4, 5] é a cauda

Listas

- Podemos acessar também dessa forma
 - $L = [\text{cabeça} \mid \text{cauda}]$, por exemplo
 - $[a \mid b, c] = [a, b, c]$

Exemplo de problema: determine se um elemento pertence a uma lista, ou seja, implemente a função (predicado) pertinencia? Ex. $\text{pertence}(x,y)$ que diz que x pertence a y .

Listas

- Exemplo de problema: determine se um elemento pertence a uma lista, ou seja, implemente a função (predicado) pertinencia?

```
pertence(X,[X|L]).  
pertence(X,[Y|L]):- pertence(X,L).
```

Exercícios

- 1- Inserir elemento numa lista. Ex.: `insere(X,L,L2)` adiciona X em L2.
- 2- Concatenar duas listas L1,L2. Ex.: `conc(L1,L2,L3)` concatena L1 com L2 e coloca o resultado em L3.
- 3- Implementar o velho ditado: todo homem é mortal, Sócrates é homem, logo é mortal.
- 4- Implementar um banco de fatos sobre família, onde possamos consultar pai, mãe, irmão e descendente.
- 5- Implementar as quatro operações matemáticas, onde possamos saber se um número é uma (soma, multiplicação, divisão, ou subtração) de dois números. Ex.: `operacao(X,Y,Z)` pode gerar "Z é uma soma de X+Y".