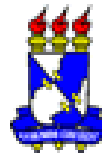


# Linguagem Lógica Prolog

Prof. Alberto Costa Neto  
alberto@ufs.br

Linguagens de Programação



Departamento de Computação  
Universidade Federal de Sergipe

# Conteúdo

- O que é diferente na Programação Lógica
- Cláusulas, Fatos, Regras e Predicado
- Objetos e suas Relações
- Consultas
- Tuplas, Átomos, Variáveis e Aridade
- Regras
- Comparação de Termos
- Recursividade
- Disjunção e Conjunção



# O que é diferente na Prog. Lógica

- Num algoritmo qualquer identificamos dois principais componentes:
  - lógica - o que solucionar (problema)
  - controle - como solucionar (solução).
- Na programação em lógica o programador deve descrever somente o componente lógico.
- O controle é exercido pelo sistema usado.



# O que é diferente na Prog. Lógica

- Um programa não é a descrição de um procedimento para se obter a solução do problema.
- Somente descreve o que solucionar.
- O responsável pelo procedimento a ser adotado na execução da solução é o sistema usado no processamento dos programas.



# O que é diferente na Prog. Lógica

- Um programa em lógica (programação declarativa) é a representação de determinado problema através de um conjunto finito de sentenças lógicas denominadas **cláusulas**.
- As cláusulas componentes dos programas em PROLOG podem ser **fatos** ou **regras**.



# O que é diferente na Prog. Lógica

- A **programação declarativa** engloba também a **programação funcional** (LISP e Haskell, por exemplo).
- Programar em uma linguagem funcional consiste em construir funções, expressas obedecendo a princípios matemáticos, para resolver um problema dado.



# Cláusulas, Fatos, Regras e Predicado

- **Cláusulas** são as sentenças lógicas componentes dos programas.
  - Um **fato** denota uma verdade incondicional.
  - As **regras** definem as condições a serem satisfeitas para que uma certa declaração seja considerada verdadeira.
- O conjunto completo de cláusulas (fatos e regras) empregados para descrever uma relação, é denominado **predicado**.



# Objetos e suas Relações

- A programação em lógica reforça a tese de que a lógica é um formalismo conveniente para representar e processar conhecimento.
- Para tanto faz-se necessário identificar nos problemas
  - objetos (entidades)
  - relação entre estes, compondo os fatos (verdades incondicionais)





# Objetos e suas Relações

É possível definir, entre os **objetos** (indivíduos) uma **relação** chamada progenitor.

```
progenitor(jo, ze) .  
progenitor(iris, gal) .
```



# Objetos e suas Relações

Cláusula:

progenitor(jo, ze).

Sendo:

**progenitor(jo, ze).**



**RELACÃO ARGUMENTOS**



# Objetos e suas Relações

Cláusulas:

`progenitor(jo, ze) .`

`progenitor(jo, ana) .`

`progenitor(ze, beth) .`

Estas constituem três cláusulas que denotam três fatos acerca da relação progenitor.



# Objetos e suas Relações

Cláusulas:

`progenitor(jo, ze) .`

`progenitor(jo, ana) .`

`progenitor(ze, beth) .`

**Exercício 1:** Usando o compilador SWI-ProLog, implementar a árvore ao lado.



# Consultas

Ao submeter cláusulas a um sistema PROLOG, ele é capaz de responder questões **consultas** sobre essas relações

Essas são denotadas por ?-.

**Exemplo:** Jô é progenitor de Zé?

?- progenitor(jo, ze).



# Consultas

Considerando que há no programa um fato declarando que Jô é progenitor de Zé, o sistema responde **yes**.

**Exemplo:** Jô é progenitor de Zé?

?- progenitor(jo, ze).  
yes



# Consultas

Jô é progenitor de Tel?

```
?-progenitor(jo, tel).
```

no

Importante observar que Jô e Tel, por tratarem-se de constantes, são escritos, em ProLog, com letras **minúsculas**.

**Exercício 2:** Elaborar 3

consultas relativas à árvore dada, cujas respostas sejam negativas.



# Consultas

É possível também formular consultas usando variáveis  $V$  (iniciadas com letras maiúsculas) dentre os argumentos.

Neste caso o sistema busca valores para  $V$  que tornem a assertiva dada verdadeira.

**Exemplo:** Quem é o progenitor de Tel?

?- progenitor(Pai, tel).  
Pai = lipe





# Consultas

**Exemplo:** Quem são os filhos de Jô?

?- progenitor(jo, Filho).

Filho = ze;

Filho = ana;

no.

Respondendo à consulta, o sistema fornece o 1º valor e aguarda (;) para continuar a pesquisa.

Importante atentar para a necessidade de usar nomes significativos para as variáveis.



# Consultas

**Exercício 3:** Sobre a cláusula dada a seguir responda:  
progenitor(X, X).

- a) a que consulta corresponde?
- b) em que situação tem resposta?

**Exercício 4:** Elabore cláusula cuja resposta corresponda a todas as relações existentes na árvore genealógica trabalhada.



# Consultas

É possível elaborar consultas por múltiplas chamadas à(s) regra(s) trabalhadas.

**Exemplo:** Quem é a avó de Gil?

?- progenitor(Avo, Pai),  
progenitor(Pai, gil).

Avo = ana

Pai = lipe



# Consultas

A resolução desta é  
efetuada em dois passos:  
(1) quem é o progenitor Y  
de Gil? e (2) quem é o  
progenitor X de Y?

**Exemplo:** Quem é a avó  
de Gil?

?- progenitor(Avo, Pai),  
progenitor(Pai, gil).

Avo = ana

Pai = lipe



# Consultas

A ordem de composição da consulta não altera o seu significado lógico.

**Exemplo:** Quem é a avó de Gil?

?- progenitor(Avo, Pai),  
progenitor(Pai, gil).

Avo = ana Pai = lipe

EQUIVALENTE A

?- progenitor(Pai, gil),  
progenitor(Avo, Pai).

Avo = ana Pai = lipe



# Consultas

Considerando a consulta:

?- progenitor(jo, X),  
progenitor(X, Y).

Como resolução obtém-se:

- (1) definição de X - filhos de Jô
- (2) definição de Y - filhos de X -  
netos de Jô.

Assim sendo, esta consulta  
corresponde a: “Quem são os  
netos de Jô?”

?-progenitor(jo, Filho),  
progenitor(Filho, Neto).



# Consultas

É possível ainda consultar, por exemplo, se dois indivíduos são irmãos.

?- progenitor(Pai, gil),  
progenitor(Pai, tel).

**Exercício 5:** Consultar se dois indivíduos são primos.



# Tuplas, Átomos, Variáveis e Aridade

- A definição de relações em ProLog é efetuada pelo estabelecimento de **tuplas** de objetos que as satisfazem.
  - Os argumentos das relações podem ser
    - objetos concretos – átomos
    - objetos genéricos - variáveis.
  - progenitor é uma relação binária, pois é definida entre dois objetos.





# Tuplas, Átomos, Variáveis e Aridade

- Um exemplo de relação **unária** é a que define o sexo dos indivíduos da árvore genealógica, ex: masculino(ze), feminino(gal).
- Outra forma de se declarar a informação relativa ao sexo dos indivíduos da árvore genealógica é: sexo(ana, feminino), sexo(ze, masculino).
- O número de argumentos que uma relação possui é denominado **aridade**. Assim, a aridade de masculino é 1 e a de sexo é 2.

**Exercício 6:** Qual o significado da consulta?  
?-masculino(Macho).



# Regras

Havendo necessidade de definição da relação filho, uma opção é (re)definir as relações progenitor como segue:

```
filho(ana, jo).  
filho(ze, jo).  
filho(gil, lipe).
```

Entretanto, é possível definir filho por meio de uma regra.



# Regras

A regra para definição da relação filho, pode ser:

filho(Filho, Pai) :-  
progenitor(Pai, Filho).

Um fato é sempre verdadeiro, enquanto regras especificam algo que pode ser verdadeiro se algumas condições forem satisfeitas.



# Regras

- ◆ Nas regras, identificam-se:

`filho(Filho, Pai) :- progenitor(Pai, Filho) .`  
↓                      ↓                      ↓  
conclusão ou cabeça   se   condição ou corpo

- ◆ Elaborada a consulta: `filho(tel, lipe)`

Dizemos que as variáveis Filho e Pai (da regra filho) foram instanciadas.

**Exercício 7:** Elabore consultas de forma a testar a aplicação da regra que define a relação filho.

`filho(Filho, Pai) :- progenitor(Pai, Filho).`



# Regras

Para definir a regra mãe, fazemos uso da **conjunção**.

mae(Mae, Filho):-  
  progenitor(Mae, Filho),  
  feminino(Mae).

Em ProLog a conjunção é denotada por **vírgula**.

**Exercício 8:** Elabore regras para definir as relações: (a) avó e (b) irmã.



# Comparação de Termos

- Para definir a regra primo, temos:  
**primo(Primo1, Primo2):-**  
    **progenitor(Pai1, Primo1),**  
    **progenitor(Pai2, Primo2),**  
    **progenitor(Pai, Pai1),**  
    **progenitor(Pai, Pai2).**

Entretanto, obtemos como resposta, por exemplo, que Lipe é primo dele mesmo, assim como Beth e Iris.

- Para evitarmos isto, podemos fazer uso de operadores de comparação de termos.
- Para dizermos que um termo é deferente de outros, usamos `\==`



# Comparação de Termos

Então, redefinindo a regra primo, temos:

```
primo(Primo1, Primo2):-  
  progenitor(Pai1, Primo1),  
  progenitor(Pai2, Primo2),  
  progenitor(Pai, Pai1),  
  progenitor(Pai, Pai2),  
  Primo1 \== Primo2,  
  Pai1 \== Pai2.
```

Ver também: ==, @<. @>,  
@=< e @=>.



# Recursividade

## **Antepassado direto:**

antepassado(Antepassado, Indivíduo):-  
    progenitor(Antepassado, Indivíduo).

## **Antepassado indireto:**

antepassado(Antepassado, Indivíduo1) :-  
    progenitor(Antepassado, Indivíduo2),  
    antepassado(Indivíduo2, Indivíduo1).

## **Antepassados:**

antepassado(Antepassado, Indivíduo1) :-  
    progenitor(Antepassado, Indivíduo1);  
    progenitor(Antepassado, Indivíduo2),  
    antepassado(Indivíduo2, Indivíduo1).





# Disjunção e Conjunção

- A **disjunção** (OU) é representada por ponto e vírgula.
- A **conjunção** (E) por uma vírgula.
- Para definição da relação antepassado, ambas foram usadas.



# Sugestões de Leitura

- The Art of Prolog (Leon Sterling / Ehud Shapiro)
- Concepts of Programming Languages (Robert Sebesta)
  - Capítulo 16
- Programming Language Design Concepts (David Watt)
  - Capítulo 15

