Programação Lógica
Parte 1

Profa. Heloisa de Arruda Camargo
1°. Semestre 2019

Programação Lógica

- Objetos de dados
- Fatos, regras e consultas
- Consultas compostas
- Unificação
- Regras recursivas

Programação Lógica

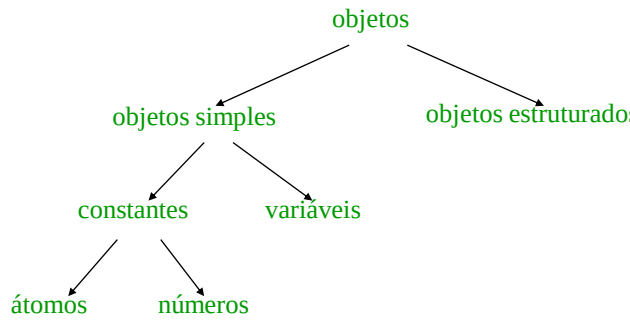
- Linguagem PROLOG - PROgramming in LOGic
- Linguagem de programação baseada no Cálculo de Predicado de Primeira Ordem
- Apropriada à:
  - Processamento simbólico, não numérico
  - Resolução de problemas que envolvam objetos e relações entre objetos
- Mecanismos básicos:
  - Casamento de padrão
  - Estrutura de listas
  - Retrocesso automático (Backtracking)

Programação Lógica

- Informação típica representada no programa:
- João é o pai de Pedro
- pai_de('João', 'Pedro').
- A lista [1, 3, 2, 7,4], quando ordenada em ordem crescente se torna a lista [1, 2, 3, 4, 7]
- ordena([1, 3, 2, 7, 4], [1, 2, 3, 4, 7]).
- A pessoa Y é o avô de uma pessoa X se a pessoa Y é o pai de X e Z é o pai de Y
- avo(X,Y) :- pai_de(X,Z), pai_de(Z,Y).

Programação Lógica

- Objetos de dados do PROLOG
  - Simples
  - Estruturados



Programação Lógica

- Objetos de dados do PROLOG
  - Átomos - cadeias de letras maiúsculas, letras minúsculas, dígitos e caracteres especiais construídas como:
    - cadeias de letras, dígitos e o caracter"_" (underscore), começando com letra minúscula
    - cadeia de caracteres especiais: ::=, <-->, etc.
    - cadeia de caracteres entre apóstrofes

Ex: joao, pedro, maria, a, x, elemento, a1, cubo_a, ponto_1

Programação Lógica

- Objetos de dados do PROLOG
  - Números – seguem a sintaxe usual da maioria das linguagens de programação
  - Exemplos:
    - inteiros: 1, -25, 4851, -9556
    - ponto flutuante: 1.55, -0.55, 84.756, 4.1
  - Variáveis – cadeias de letras, dígitos e caracter "_", começando com letra maiúscula ou com o caracter "_"
  - Exemplos:
    - X, X1, Lista1, _abc, YZW, A123

Programação Lógica

- Objetos de dados do PROLOG
  - Estruturas ou Objetos estruturados – são objetos de dados que tem vários componentes, podendo cada um deles, por sua vez ser uma estrutura.
  - A combinação dos componentes é feita através do **funtor**, que dá um nome para a estrutura:
    - data(13, marco, 2018)
    - autor('Russel & Norvig')
    - par(primeiro,segundo)
    - semana(seg,ter,qua,qui,sex,sab,dom)
    - semana(util(seg,ter,qua,qui,sex), fim-de(sab,dom))
    - livro(titulo('Inteligencia Artificial'), autor('G. Bittencourt'))

Programação Lógica

- Relações ou Predicados
  - Componente principal das construções Prolog
  - Descrevem algum tipo de ligação entre os objetos
    - **(Quais objetos? Simples ou estruturados)**

    - pai_de(joao, pedro)
    - bonita(maria)
    - gosta_de(ana, vinho)
    - maior(5,2)
    - sobre(cubo_a, cubo_b)
    - gato(pequeno)
    - menor(X,Y)
    - joga(carlos, futebol)
    - joga(joana, X)
    - entrega(trabalho1, data(13, marco, 2018)
    - pai_de(joao, filhos(pedro, maria))

Programação Lógica

- Relações ou Predicados
  - Qual a diferença entre predicados e estruturas?

- Predicados X Estruturas

  - Estruturas são formalmente idênticas aos predicados
  - Todos predicado é uma estrutura
  - Nem toda estrutura é um predicado
  - Um predicado é uma estrutura que declara coisas que podem ser verdadeiras ou falsas
  - Estruturas que nomeiam objetos não fazem declarações e não podem ser verdadeiras ou falsas

Programação Lógica

- Programas
- Um programa Prolog consiste de:
  - Fatos
  - Regras
  - Consultas

Programação Lógica

- Fatos
  - Um fato é uma declaração de que uma determinada relação existe entre certos objetos.
  - pai_de(joao, pedro).
  - bonita(maria).
  - gosta_de(ana, vinho).
  - maior(5, 4, 3, 2, 1).
  - gosta(X, vinho). (fatos universais)
  - vezes(0, X, 0).
- Base de dados: conjunto de fatos em um programa Prolog.

Programação Lógica

- Exemplo 1 – Família real

```
% base de dados
pai_de (henrique_pai, henrique).
pai_de (henrique_pai, maria).
pai_de (henrique, elizabeth2).
pai_de(henrique, eduardo).
```


Exemplo 1: FAMÍLIA REAL

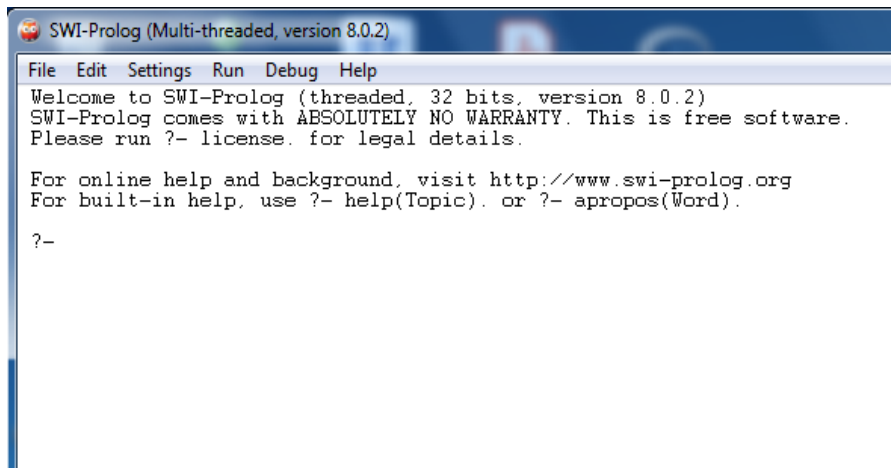
```
% base de dados
pai_de(henrique_pai, henrique).          %1
pai_de(henrique_pai, maria).             %2
pai_de(henrique, elizabeth2).            %3
pai_de(henrique, eduardo).               %4
homem(henrique_pai).                     %5
homem(henrique).                         %6
homem(eduardo).                          %7
mulher(catarina).                        %8
mulher(elizabeth1).                      %9
mulher(maria).                           %10
mulher(elizabeth2).                      %11
mulher(ana).                             %12
mulher(jane).                            %13
mae_de(catarina, maria).                 %14
mae_de(ana, elizabeth2).                 %15
mae_de(jane, eduardo).                   %16
mae_de(elizabeth1, henrique).            %17
```

Programação Lógica

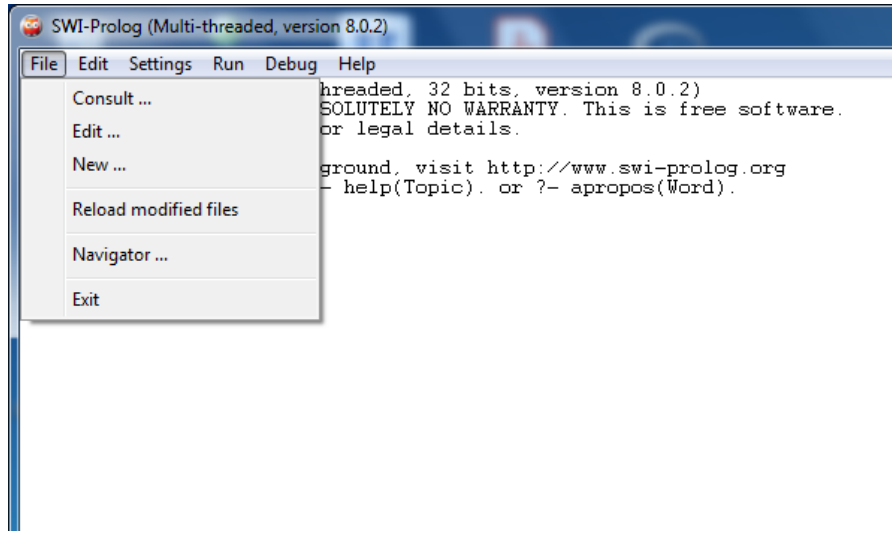
- Os exemplos deste material seguem a sintaxe do SWI-Prolog, que pode ser encontrado em:

http://www.swi-prolog.org/

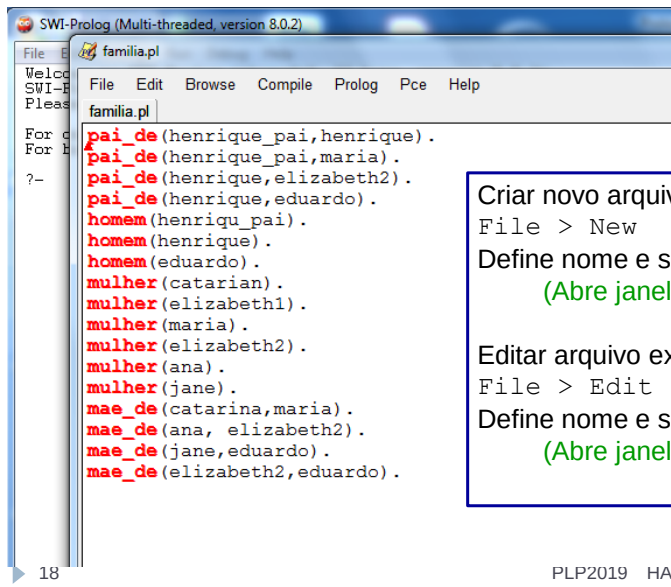
Ambiente de desenvolvimento SWI-Prolog



Ambiente de desenvolvimento SWI-Prolog



Ambiente de desenvolvimento SWI-Prolog



Criar novo arquivo:
`File > New`
Define nome e salva
(Abre janela do editor)

Editar arquivo existente:
`File > Edit`
Define nome e seleciona abrir
(Abre janela do editor)

Programação Lógica

- Consultas

- São o meio de recuperar informação em um programa lógico. Podem ser de dois tipos:
  - Confirmação
  - Recuperação

Programação Lógica

- Consultas de confirmação
  - a busca é realizada até encontrar uma resposta, confirmando ou negando o que foi perguntado.
  - Exemplo – após iniciar uma sessão do interpretador e carregar a base de dados Família real:

```
| ?- pai_de(henrique,eduardo).
true.

| ?- pai_de(henrique,maria).
false.
```

Programação Lógica

- Consultas de recuperação
  - - todos os valores que satisfazem a consulta são recuperados

```
| ?- pai_de(X,maria).

X = henrique_pai .
```

```
| ?- pai_de(henrique,X).

X = elizabeth2 ;

X = eduardo
```

```
| ?- pai_de(X,eduardo).

X = henrique.
```

```
| ?- mae_de(X,maria).

X = catarina.
```


Programação Lógica

- Consultas de recuperação

```
| ?- mae_de(catarina,X).
X = maria

| ?- mae_de(X,catarina).
false.
```

```
| ?-mae_de(X,Y).
X = catarina ,
Y = maria ;
X = ana ,
Y = elizabeth2 ;
X = jane ,
Y = eduardo ;
X = elizabeth1 ,
Y = henrique
```

Programação Lógica

- Consultas compostas
- São interpretadas como conjunção

| ?- pai_de(X,elizabeth2) ,pai_de(X,eduardo).

(Existe um valor para X que torne as duas partes da consulta verdadeiras ao mesmo tempo?)

X = henrique .

Programação Lógica

- Consultas compostas

| ?- pai_de(X,eduardo),pai_de(Y,X).
(Quem é o avô de eduardo?)
X = henrique ,
Y = henrique_pai .

| ?- pai_de(henrique_pai,X),pai_de(X,Y).
(Quem são os netos de henrique_pai?)
X = henrique ,
Y = elizabeth2 ;
X = henrique ,
Y = eduardo ;
false.

Programação Lógica

- Unificação
  - Dois termos unificam se:
  1. São idênticos
  2. As variáveis em ambos os termos podem ser instanciadas em objetos de tal forma que após as substituições das variáveis por esses objetos os termos se tornam idênticos
- Exemplo

Termo 1 : data(25, maio, Ano)
Termo 2 : data(D, maio,1983)

Termo 1 e Termo2 unificam
Resultado : D = 25 Ano = 1983

Programação Lógica

- Unificação – outros exemplos

Termo 1 : data(D1,abril, A)
Termo 2 : data(D2,M,1900)
Resultado : D1 = D2
M = abril
A = 1900

Termo 1: pai_de(henrique, filhos(eduardo,elizabeth2))
Termo 2: pai_de(henrique,X)
Resultado: X = filhos(eduardo,elizabeth2)

Programação Lógica

- Unificação
- Regras para decidir se dois termos unificam

1. Se S e T são constantes então S e T unificam se e só se são o mesmo objeto;
2. se S é uma variável e T é qualquer termo, então unificam e S é instanciada com T; vice-versa com a variável T instanciada com S;
3. se S e T são estruturas, elas unificam se e só se S e T tem o mesmo funtor principal e todos os elementos correspondentes unificam.

Programação Lógica

- Unificação - exemplos

Termo 1	**Termo 2**	**Resultado da Unificação**
henrique	henrique	unificam
eduardo	henrique	não unificam
X	par(a,b)	X = par(a,b)
2.35	Y	Y = 2.35
data(25,maio,Ano)	data(D,maio,1983)	D = 25 Ano = 1983
data(D1,abril,A)	data(D2,M,1900)	D1 = D2 M = abril A = 1900
data(17,marco,2000)	date(17,M,2000)	não unificam
pai_de(X,eduardo)	pai_de(henrique,Y)	X = henrique Y = eduardo

Programação Lógica

- Regras
- Componente do programa Prolog que permite definir novas relações a partir das já existentes
- Definir relações de filho e filha:

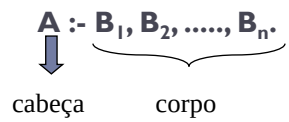
```
filho_de(Y,X) :- pai_de(X,Y), homem(Y).
filha_de(Y,X) :- pai_de(X,Y), mulher(Y).
```

- Definir a relação de avô:

```
avo_de(X,Z) :- pai_de(X,Y), pai_de(Y,Z).
```


Programação Lógica

- Regras - Forma geral



- A: objetivo, meta
- B_i: subobjetivos, condições
- Para provar A, provamos B_1, B_2, ..., B_n
- O conjunto de regras de um programa é chamado de base de conhecimento

Programação Lógica

- Cláusulas

- Regras,
- Fatos e
- Consultas

  - são chamadas de cláusulas de Horn,
  - ou somente cláusulas

Programação Lógica

- Exemplo

- O programa Família Real pode ser expandido acrescentando as regras, que formam a base de conhecimento do programa:

```
%base de conhecimento

filho_de(Y,X) :- pai_de(X,Y), homem(Y).      %18
filha_de(Y,X) :- pai_de(X,Y), mulher(Y).     %19
avo_de(X,Z) :- pai_de(X,Y), pai_de(Y,Z).     %20
```

Programação Lógica

- Como o programa Prolog usa as regras?

- Colocada a consulta:

  | ?- filho_de(eduardo,henrique).

- Não há no programa fatos sobre a relação filho_de.
- É necessário usar as regras
  - A consulta é comparada com a cabeça das regras que definem a relação filho_de, na sequencia.
  - Ocorre uma unificação entre a consulta e a cabeça da regras %18, com as instanciações:

    Y = eduardo

    X = henrique

Programação Lógica

- Como o programa Prolog usa as regras?

A regra fica:

filho_de(eduardo,henrique):-
pai_de(henrique,eduardo), homem(eduardo)

- O objetivo é substituído pelos sub-objetivos:
  pai_de(henrique,eduardo), homem(eduardo)

  que devem ser verdadeiros ao mesmo tempo.

- Os sub-objetivos são provados, pois são fatos no programa.

Logo, o objetivo também é verdadeiro e a resposta é:
true.

Programação Lógica

- Como o programa Prolog usa as regras?

```
?- filho_de(eduardo,X).
X = henrique.

?- avo_de(henrique_pai,X).
X = elizabeth2 ;
X = eduardo ;
false.
?- avo_de(X,maria).
false.

?- avo_de(X,eduardo).
X = henrique_pai ;
false.

?- filha_de(X,henrique_pai).
X = maria.

?- filha_de(X,henrique).
X = elizabeth2 ;
false.

?- avo_de(X,Y).
X = henrique_pai,
Y = elizabeth2 ;
X = henrique_pai,
Y = eduardo ;
false.
```


Programação Lógica

- Definindo uma nova relação com mais de uma regra
  - Duas ou mais regras que definem a mesma relação indicam formas alternativas de provar um objetivo
  - Logo, correspondem ao operador "ou"
- Pelo mecanismo de backtracking, as regras são utilizadas na sequência em que aparecem

Programação Lógica

- Considere as regras:

```
pai_ou_mae(X,Y) :- pai_de(X,Y).        %21
pai_ou_mae(X,Y) :- mae_de(X,Y).        %22
```

- E a consulta:
- | ?- pai_ou_mae(X,elizabeth2).

- O processamento inicia com a regra %21 e depois passa para a regra %22

```
X = henrique ;
X = ana.
```


Programação Lógica

- Outros exemplos:

```
| ?- pai_ou_mae(X,eduardo).
   X = henrique ;
   X = jane.

| ?- pai_ou_mae(jane,X).
   X = eduardo.

| ?- pai_ou_mae(henrique,Y).
   Y = elizabeth2 ;
   Y = eduardo ;
   false
```

Programação Lógica

```
%O programa FAMILIA REAL pode ser expandido com as regras
% que são acrescentadas, formando a base de conhecimento

%base de conhecimento

filho_de(Y,X) :- pai_de(X,Y), homem(Y).        %18
filha_de(Y,X) :- pai_de(X,Y), mulher(Y).       %19
avo_de(X,Z) :- pai_de(X,Y), pai_de(Y,Z).       %20
pai_ou_mae(X,Y) :- pai_de(X,Y).                %21
pai_ou_mae(X,Y) :- mae_de(X,Y).                %22
```


Programação Lógica

- Regras recursivas
  - Recursão: operação em que um objeto é usado em sua própria definição
  - Regras recursivas: definidas em termos delas mesmas
- Exemplo:
  - Definir a relação de predecessor
  - Considerar a base de dados Família Real
  - predecessor(X,Y) :- pai_de(X,Y) %23
  - predecessor(X,Y) :- pai_de(X,Z), predecessor (Z,Y) %24

Programação Lógica

- Regras recursivas
  - Consultas:

```
|?- predecessor(henrique_pai,X).
X = henrique ;
X = maria ;
X = elizabeth2 ;
X = eduardo ;
false.

| ?- predecessor(X,henrique).
X = henrique_pai.

?- predecessor(X,eduardo).
X = henrique ;
X = henrique_pai ;
false.
```

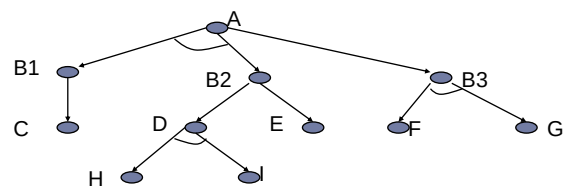
```
pai_de(henrique_pai, henrique).        %1
pai_de(henrique_pai, maria).           %2
pai_de(henrique, elizabeth2).          %3
pai_de(henrique, eduardo).             %4
    .
    .
    .
```

```
predecessor(X,Y) :- pai_de(X,Y)        %23
predecessor(X,Y) :- pai_de(X,Z),
                    predecessor (Z,Y)  %24
```


Programação Lógica

- Representação de um programa Prolog na forma de
  - Árvore AND/OR
    - Nós AND - representam conjunções
    - Nós OR - representam disjunções (caminhos alternativos)
  - A árvore AND/OR de um programa Prolog representa todos os caminhos que levam a uma solução, isto é, permitem provar o objetivo

```
A :- B1, B2.
A :- B3.
B1 :- C.
B2 :- D.
B2 :- E.
B3 :- F, G.
D :- H, I.
```



Programação Lógica

- Árvore de execução
  - Árvore que representa o processo de execução de uma consulta
  - Nós – representam objetivos (simples ou compostos)
  - Arcos de um nó A para um nó B – representam substituições de um objetivo por subobjetivos pela unificação do objetivo do nó A com fatos e regras
- Representação usada para acompanhar a execução de uma consulta passo a passo
- Não é a árvore AND/OR

Programação Lógica

- Observação 1:
  - Variáveis DIFERENTES em um mesma cláusula PODEM assumir valores iguais
  - Nas relações lógicas, ignorar esse detalhe pode levar a resultados corretos computacionalmente mas sem sentido lógico
- Exemplo: Definir a relação de irmã

```
pai_ou_mae(tom,bob).
pai_ou_mae(tom,liz).
pai_ou_mae(bob,ana).
pai_ou_mae(bob,pat).
pai_ou_mae(pat,jim).
mulher(ana).
mulher(pat).
irma(X,Y) :- pai_ou_mae(Z,X),
             pai_ou_mae(Z,Y),
              mulher(Y).
```

Programação Lógica

- Observação: I
  - Variáveis DIFERENTES em um mesma cláusula PODEM assumir valores iguais

- Consulta:

```
| ?- irma(pat,X).
X = ana ;
X = pat ;
false
```

  - A segunda resposta equivale a dizer que pat é irmã dela mesma

Programação Lógica

- Observação 2:
  - Escopo de variáveis: O escopo de todas as variáveis é limitado a uma cláusula
  - Não existem variáveis globais

```
pai_ou_mae(tom,bob).
pai_ou_mae(tom,liz).
pai_ou_mae(bob,ana).
pai_ou_mae(bob,pat).
pai_ou_mae(bob,joe).
pai_ou_mae(pat,jim).
mulher(ana).
mulher(pat).
homem(joe).
homem(bob).
irma(X,Y) :- pai_ou_mae(Z,X),
             pai_ou_mae(Z,Y),
             mulher(Y).
```

Consulta:

```
?- irma(pat,X).
```

A variável X da consulta não é a mesma variável X da regra