

# Protótipo de Software de Apoio ao Aprendizado da Linguagem de Programação Lógica Prolog

ORIENTANDO : WAGNER M. STAHNKE

ORIENTADOR : ROBERTO HEINZLE

FURB - Universidade Regional de Blumenau  
Bacharelado em Ciências da Computação

# Roteiro da Apresentação

- Introdução
- Trabalhos Correlatos
- Aspectos Matemáticos
- Programação Lógica
- Ambiente Prolog
- Implementação do Protótipo
- Conclusão
- Extensões

# Introdução

---

- História
- Área
- Motivação
- Objetivos

# Introdução - História

---

- Prolog, "PROgramming in LOGic".  
Edimburgo - Escócia no início da década de 70
- Sua popularidade mundial aconteceu devido ao projeto Japonês de Computadores de Quinta Geração

# Introdução - Área

---

- A principal área de Aplicação para o PROLOG é a Inteligência Artificial (IA)
- O presente trabalho se enquadra na Área da Educação

# Introdução - Motivação

---

- Existem diversas razões para se fazer um estudo sobre programação lógica
- A programação lógica oferece uma maneira diferente de pensar sobre a resolução de problemas
- Sua Linguagem se dá de forma mais Natural do que outras linguagens

# Introdução - Objetivos

---

- Desenvolvimento de um protótipo de software capaz de visualizar o processo de execução de programas escritos na linguagem de programação Prolog

# Trabalhos Correlatos

```
File Edit Buffers Info Debug Switch Help
                               M A I N
->
yes
?- avo_de(X,Y).

(0) CALL: avo_de(_18,_20) ? >
(1) CALL: pai_de(_18,_734) ? >
(1) EXIT(N): pai_de('Ari Fontoura','Jonas Fontoura')
(2) CALL: pai_de('Jonas Fontoura',_20) ? >
(2) EXIT(D): pai_de('Jonas Fontoura','Mario Fontoura')
(0) EXIT(N): avo_de('Ari Fontoura','Mario Fontoura')
Type any key to continue.
```



# Programação Lógica (PL)

---

- Relações
- Lógica
- Programação em Lógica
  - Fatos e Regras
- Áreas de Aplicação

# Aspectos Matemáticos: Lógica das Proposições

## ■ Fórmulas Atômicas:

- A, B, PedroTemCarro

## ■ Símbolos:

- Conectivos:  $\wedge$  (e),  $\vee$  (ou),  $\Rightarrow$  (se... somente se),  $\neg$  (negação).

## ■ Regras de Sintaxe:

- Uma fórmula atômica é uma fbf (**fórmula bem formada**)
- Se  $\alpha$  é uma fbf então  $(\neg \alpha)$  é uma fbf
- Se  $\alpha$  e  $\beta$  são fbfs, então  $(\alpha \wedge \beta)$ ,  $(\alpha \vee \beta)$ ,  $(\alpha \Rightarrow \beta)$  e  $(\alpha \equiv \beta)$  são fbfs

### Exemplos de fbfs:

$$(A \wedge B) \Rightarrow (\neg C)$$

$$A \Rightarrow (\neg B)$$

$$(A \vee B) \Rightarrow C$$

$$(A \Rightarrow B) \Rightarrow (\neg C \Rightarrow \neg D)$$

$$\neg \neg A$$

# Aspectos Matemáticos: Lógica dos Predicados

## ■ Fórmulas Atômicas

- $\text{pai}(X, Y)$ ,  $\text{homem}(X)$

## ■ Símbolos: (além de)

- Quantificadores:  $\forall$  (para todo),  $\exists$  (existe)

## ■ Regras de Sintaxe: (além das já citadas)

- se  $t_1, \dots, t_n$  são termos e  $p$  é um predicado que tem  $n$  argumentos, então  $p(t_1, \dots, t_n)$  é uma fbf
- se  $V$  é uma variável e  $\alpha$  é uma fbf, então  $(\forall V \alpha)$  e  $(\exists V \alpha)$  são fbfs

Exemplos de fbfs:

$\text{pai}(\text{joão}, \text{vitor})$

$\forall X \forall Y (\text{pai}(Z, X) \wedge \text{pai}(Z, Y) \Rightarrow \text{irmao}(X, Y))$

$\forall X (\text{humano}(X) \Rightarrow \text{mortal}(X))$

$\exists X \exists Y (\text{humano}(X) \wedge \text{humano}(Y) \wedge \text{ama}(X, Y))$

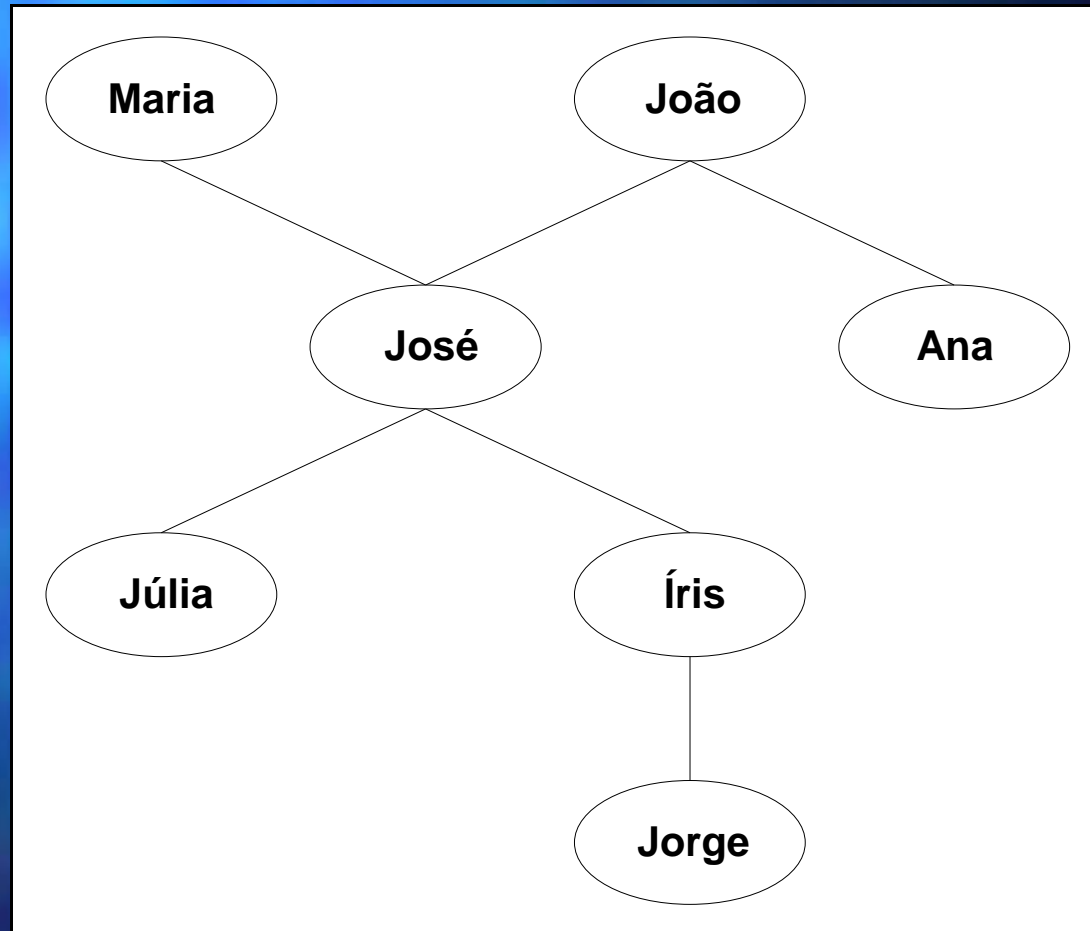
# Ambiente Prolog

---

- Exemplo
- Fatos
- Regras
- Construções Recursivas

# Prolog: Exemplo

- Família



# Prolog: Fatos

```
progenitor(maria, josé).  
progenitor(joão, josé).  
progenitor(joão, ana).  
progenitor(josé, júlia).  
progenitor(josé, íris).  
progenitor(íris, jorge).
```

```
?- progenitor(josé, íris).  
sim  
?- progenitor(ana, jorge).  
não  
?- progenitor(luís, íris).  
não
```

```
?- progenitor(X, íris).  
X=josé  
?- progenitor(josé, X).  
X=júlia;  
X=íris;  
não
```

```
?- progenitor(X, Y).  
X=maria Y=josé;  
X=joão Y=josé;  
X=joão Y=ana;  
X=josé Y=júlia.
```

```
?- progenitor(X, Y), progenitor(Y, jorge).  
X=josé Y=íris.
```

# Prolog: Regras

```
filho(Y, X) :-  
    progenitor(X, Y).  
avo(X, Z) :-  
    progenitor(X, Y),  
    progenitor(Y, Z).  
irmao(X, Y) :-  
    progenitor(Z, X),  
    progenitor(Z, Y).
```

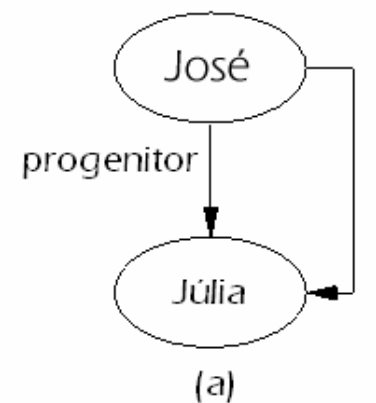
```
?- filho(X, josé).  
X=júlia;  
X=íris.  
?- avo(X, jorge).  
X=josé;  
?- irmao(X, íris).  
X=júlia;  
X=íris.
```

```
?- filho(jorge, X), irmao(X, Y), avo(joão, Y).  
X=íris Y=júlia.
```

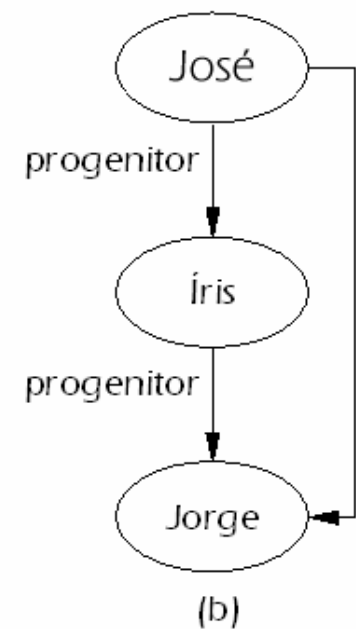
# Prolog: Construções Recursivas

```
antepassado(X, Z) :-  
    progenitor(X, Z).  
antepassado(X, Z) :-  
    progenitor(X, Y),  
    antepassado(Y, Z).
```

```
?- antepassado(maria, julia).  
sim  
?- antepassado(X, jorge).  
X=íris;  
X=joão;  
X=maria.
```



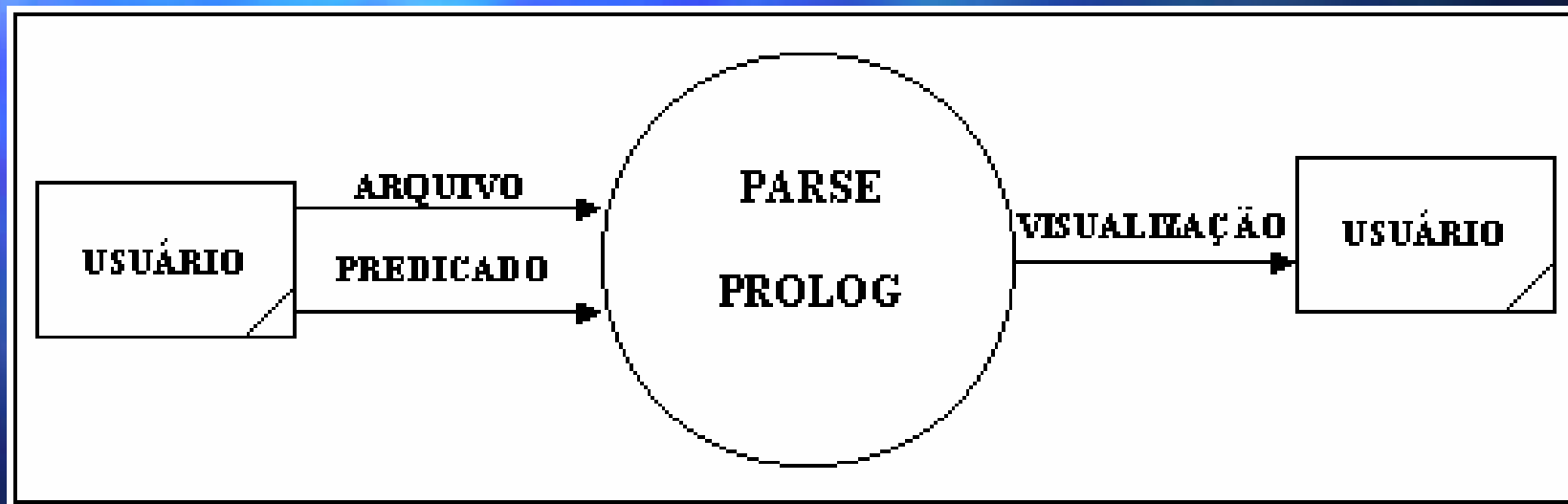
antepassado direto



antepassado indireto



# Implementação do Protótipo: Especificação



# Implementação do Protótipo: Especificação

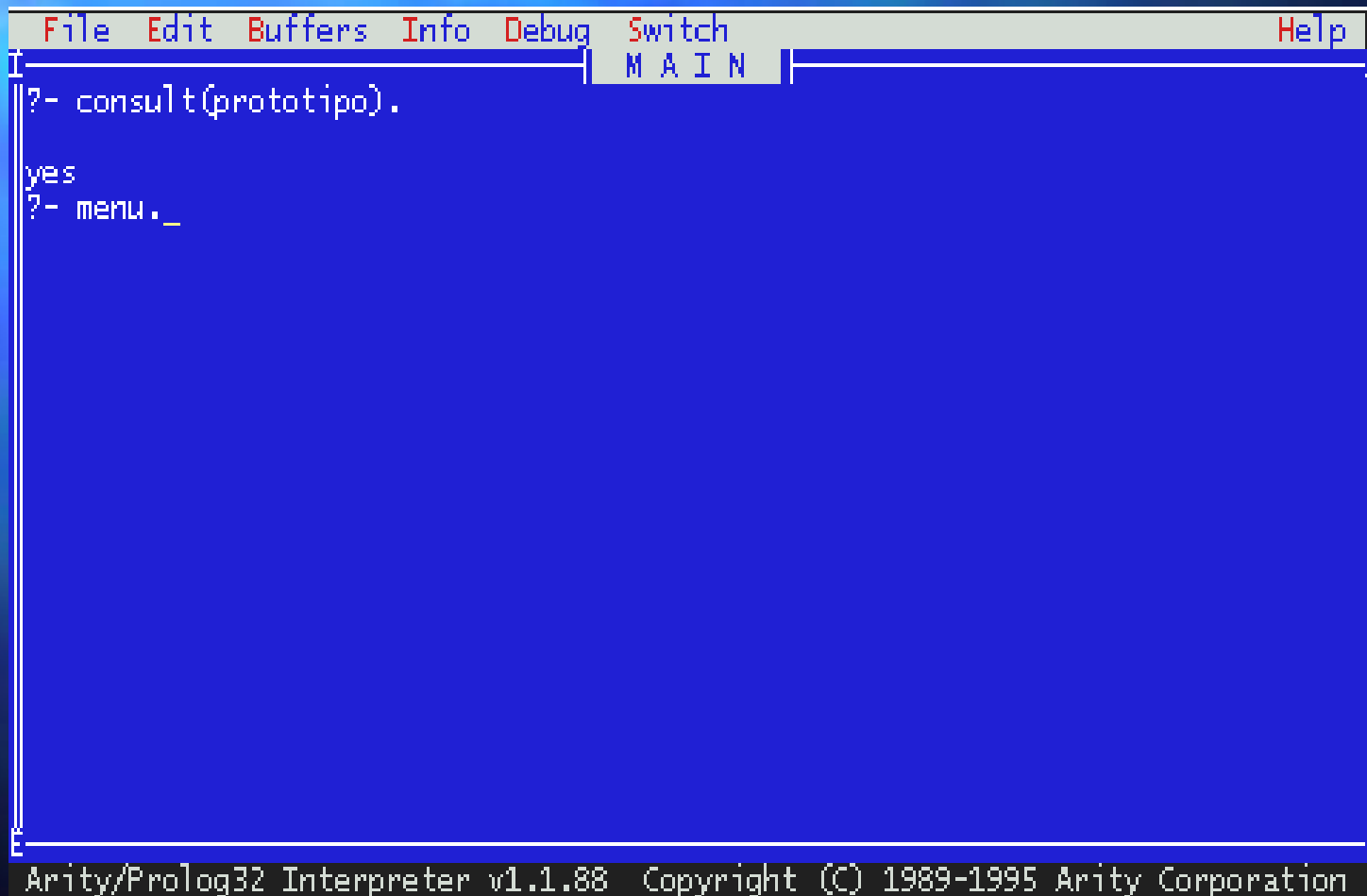
- $\langle \text{regra} \rangle ::= \langle \text{condição} \rangle$
- $\langle \text{condição} \rangle ::= \langle \text{cláusula1} \rangle \mid \langle \text{cláusula2} \rangle \mid \langle \text{cláusula3} \rangle \mid \langle \text{cláusula4} \rangle$
- $\langle \text{cláusula1} \rangle ::= \langle \text{predicado} \rangle \text{ e } \langle \text{predicado} \rangle \text{ e } \langle \text{valor} \rangle$
- $\langle \text{cláusula2} \rangle ::= \langle \text{sistema} \rangle \text{ e } \langle \text{valor} \rangle$
- $\langle \text{cláusula3} \rangle ::= \langle \text{predicado} \rangle \text{ e } \langle \text{valor} \rangle$
- $\langle \text{cláusula4} \rangle ::= \langle \text{predicado} \rangle \text{ e } \langle \text{valor} \rangle$
- $\langle \text{predicado} \rangle ::= = \mid \neq$
- $\langle \text{valor} \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid \dots \mid 10$
- $\langle \text{sistema} \rangle ::= \langle \text{predicado\_sistema} \rangle$
- $\langle \text{predicado\_sistema} \rangle ::= \text{write} \mid \text{read} \mid \text{keyb} \mid \text{nl} \mid \dots \mid \text{open} \mid$

# Implementação do Protótipo

---

- Metodologia Utilizada:
  - Prototipação
- Ferramenta Utilizada:
  - Arity Prolog32 V1.1.88

# Protótipo: Operacionalidade



```
File Edit Buffers Info Debug Switch Help
MAIN
?- consult(prototipo).
yes
?- menu._

Arity/Prolog32 Interpreter v1.1.88 Copyright (C) 1989-1995 Arity Corporation
```

# Protótipo: Operacionalidade

= Prototipo de Apoio ao Aprendizado do Arity Prolog - APLOG =

FURB - Fundação Universidade Regional de Blumenau  
Centro de Ciências Exatas e Naturais  
Depto de Sistemas e Computação  
Curso: Bacharelado em Ciências da Computação  
Orientador: Roberto Heinzle

Protótipo de Software de Apoio ao Aprendizado da Linguagem  
de Programação Lógica - PROLOG

Acadêmico: Wagner Moreira Stahnke

<<pressione qualquer tecla>>\_

# Protótipo: Operacionalidade

Prototipo de Apoio ao Aprendizado do Arity Prolog - APLOG

Diga o nome do arquivo sem extensão: teste.

Diga seu predicado principal: avo\_de(X,Y).

# Protótipo: Operacionalidade

```
Prototipo de Apoio ao Aprendizizado do Arity Prolog - APLOG  
CALL  
  
CALL:  
pai_de(Jonas Fontoura,_7C)  
EXIT  
  
EXIT:  
avo_de(Ari Fontoura,Mario Fontoura)  
-
```

# Conclusões

---

- O presente trabalho permitiu um estudo mais aprofundado da linguagem de programação Prolog, bem como alguns de seus componentes e ferramentas para construção deste protótipo aqui demonstrado
- O protótipo atendeu aos requisitos propostos inicialmente



# Extensões

---

## ■ Protótipo

- Implementar: Análise Sintática, de modo a verificar se um arquivo está com o formato correto das instruções
- Utilizar de recursos de outra linguagem de programação, de preferência visual
- Uma melhor geração de telas