



Linguagem de Programação I

Programação Lógica

Equipe:

Yuri

Joyce

Carlos Éderson

Paulo Maurício



Súmario

- Introdução
- Origem
- Características
- Conceitos básicos
- Vantagens e desvantagens
- Aplicações
- Linguagem Prolog

Introdução

- Criador: John McCarthy.
- A programação em lógica se desenvolveu no início dos anos 70 a partir de alguns trabalhos sobre prova de teoremas;
- Desde então ela tem demonstrado ser um formalismo simples, mas muito poderoso, que é bastante adequado tanto para a representação do conhecimento quanto como ferramenta de programação;
- Primeira Linguagem Lógica: Planner.



Paradigmas

- Paradigma baseado na Lógica Matemática e no Cálculo de Predicados
- Programas são **declarativos**:
 - Especificam **resultados desejados em vez de** procedimentos para produzi-los
- Possui semelhanças com o paradigma funcional
- Separação de lógica e controle:
 - **Lógica**: definição do que deve ser solucionado
 - **Controle**: como a solução pode ser obtida
- Programador escreve definições que permitam a **dedução** da solução

Vantagens e Desvantagens

- Vantagens:
 - Existem mecanismos de inferência eficientes que deduzem o valor verdade de uma fórmula em tempo (e espaço) linear em função do número de constantes da fórmula
 - Existem também mecanismos de inferência eficientes para abdução (diagnóstico) e indução (aprendizagem).
- Desvantagens:
 - Em domínios complexos, explosão combinatória da base de conhecimento não permite:
 - Representar relações genéricas entre conceitos representados em intenção
 - Estruturação modular do conhecimento
 - Representação de conhecimento em ambientes dinâmicos ou não deterministas ou contínuos.

Diferenças entre programas convencionais e programas em lógica

PROGRAMAS CONVENCIONAIS	PROGRAMAS EM LÓGICA
Processamento Numérico	Processamento Simbólico
Soluções Algorítmicas	Soluções Heurísticas
Estruturas de Controle e Conhecimento Integradas	Estruturas de Controle e Conhecimento Separadas
Difícil Modificação	Fácil Modificação
Somente Respostas Totalmente Corretas	Incluem Respostas Parcialmente Corretas
Somente a Melhor Solução Possível	Incluem Todas as Soluções Possíveis

Aplicações

- Sistemas Baseados em Conhecimento (SBCs)
- Sistemas de Bases de Dados (BDs)
- Sistemas Especialistas (SEs):
- Processamento da Linguagem Natural (PLN)
- Educação
- Arquiteturas Não-Convencionais

Prolog

- Foi criada em meados de 1972 por Alain Colmerauer e Philippe Roussel, na Universidade de Marselha.
- O nome PROLOG foi escolhido por Philippe Roussel como uma abreviação de “PROgrammation en LOGique”.
- Propósito da criação: criar programas para tradução de linguagens faladas, como português ou inglês.

Prolog

- Todos os objetos no Prolog são denominados termos
- Alguns termos:
 - átomo
 - Numero
 - Variável
 - Fato
 - Regra

As Interpretações

- Declarativa.
 - As Cláusulas* são vistas como descrição do problema
- Procedimental
 - As Cláusulas são vistas como entrada para um método
- Operacional
 - As Cláusulas são vistas como comandos para um procedimento de prova por refutação

Fatos

- São fatos!!!
 - Representam um sentença verdadeira no PROLOG para o universo do problema em questão.
- Alguns fatos: Dunha está vivo, João conhece o Mário
- De forma logica porderia ser interpretado como:
 - Vd (onde: V=vivo e d=dunha)
 - Cjm (onde: C=conhece, j=joão e m=mário)
- e no prolog (tem que ser minúsculo mesmo*):
 - vivo(dunha).*
 - conhece(joão,mário).

Regras (P \rightarrow Q)

- Relação causa-conseqüência, sendo que o conseqüente é
- informado antes do antecedente.
- considerando os fatos sobre paternidade**
criar uma
- regra para a relação "avô"

Regras

Interpretação:

"Uma pessoa é avô de outra, se a primeira for pai de um terceira pessoa, e essa terceira pessoa for pai da segunda"

Formalizando:

para todo X e Y
 X é avô de Y se
 X for pai de Z e
 Z for pai de Y .

Regras

E o código em Prolog:

```
avo(X,Y) :- pai(X,Z), pai(Z,Y).
```

operadores logico:

not() -> negação

"," -> conjunção

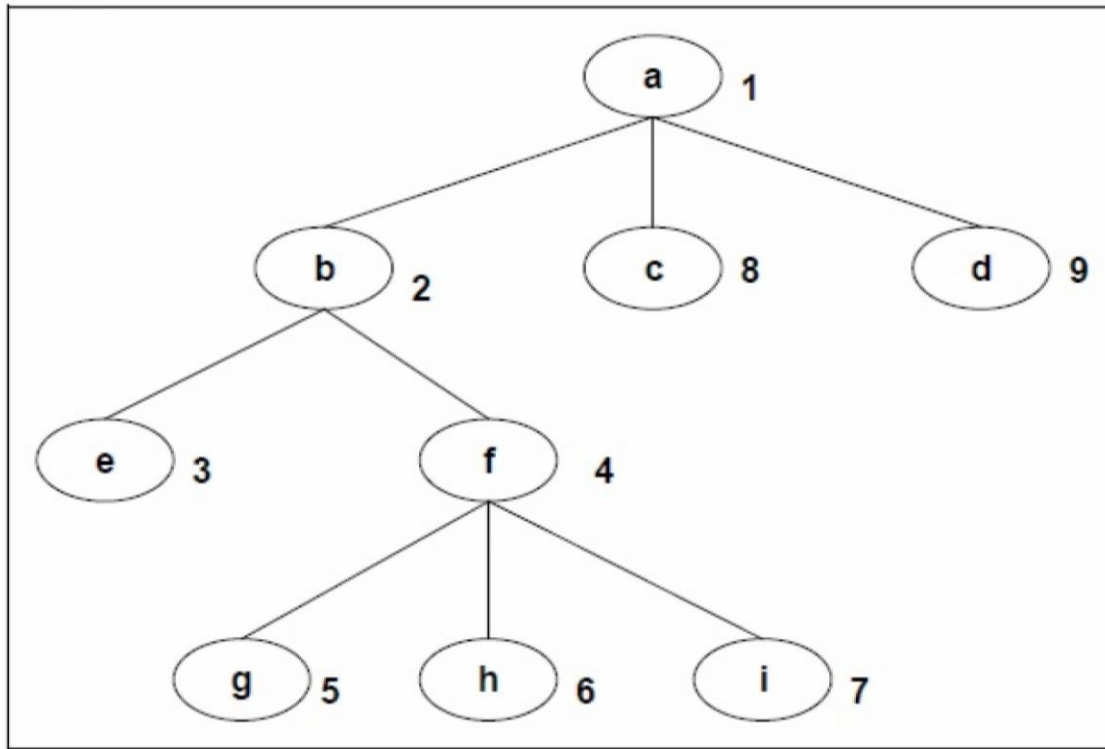
;" -> disjunção

ESTRUTURAS DE CONTROLE

- Backtracking
- Cut (!)
- Negação por fail

ESTRUTURAS DE CONTROLE

Backtracking

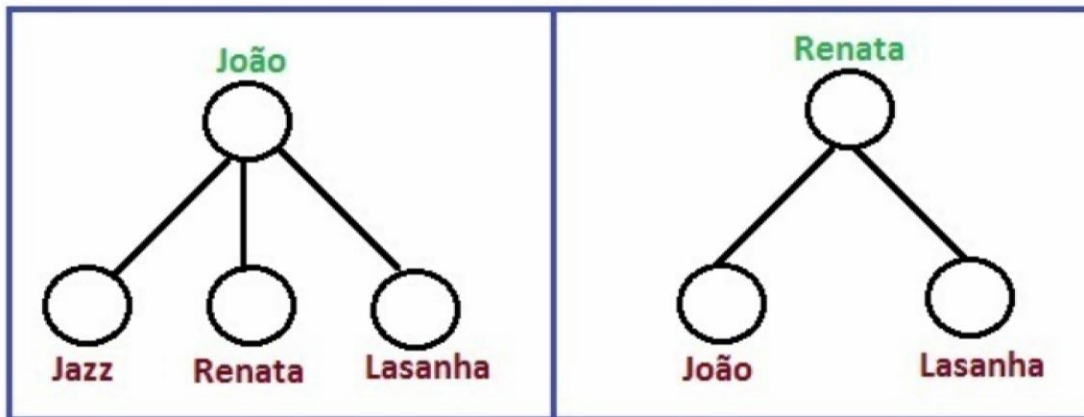


a :- b.
a :- c.
a :- d.
b :- e.
b :- f.
f :- g.
f :- h.
f :- i.
d.

ESTRUTURAS DE CONTROLE

Backtracking

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, joão).  
gosta(renata, lasanha).
```

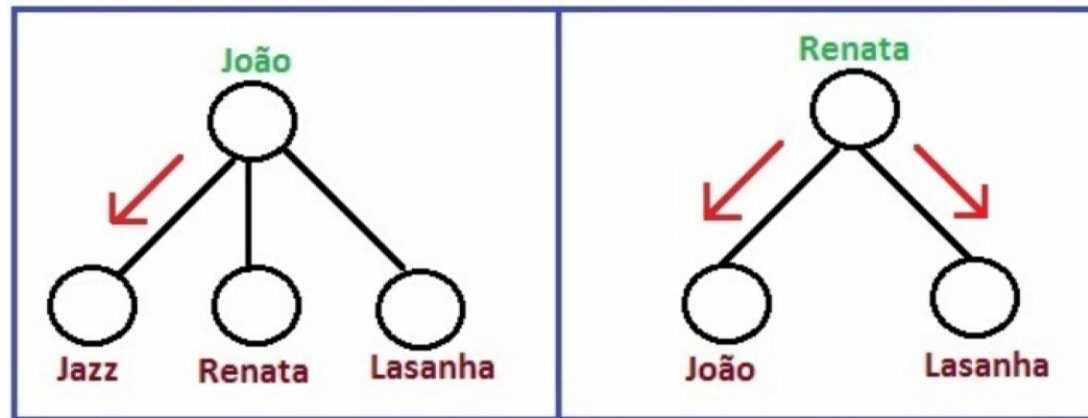


`gosta(joão, x), gosta(renata, x).`

ESTRUTURAS DE CONTROLE

Backtracking

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, joão).  
gosta(renata, lasanha).
```



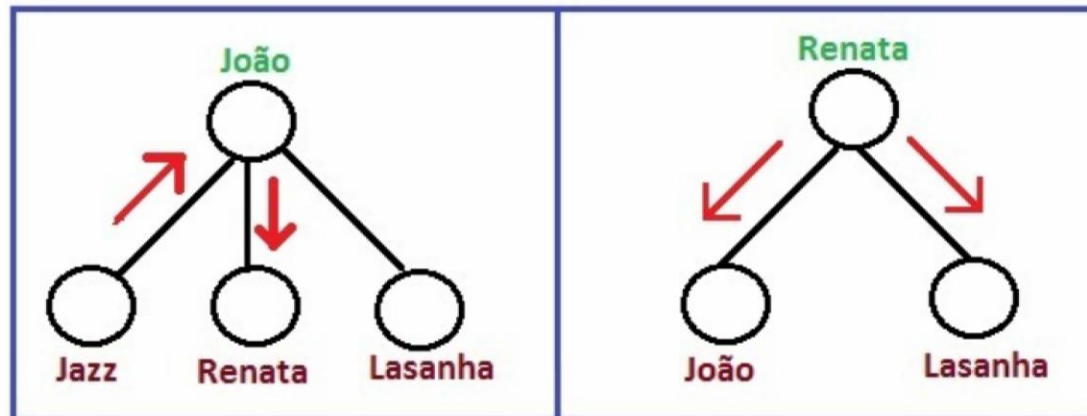
`gosta(joão, x), gosta(renata, x).`

X = Jazz

ESTRUTURAS DE CONTROLE

Backtracking

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, joão).  
gosta(renata, lasanha).
```



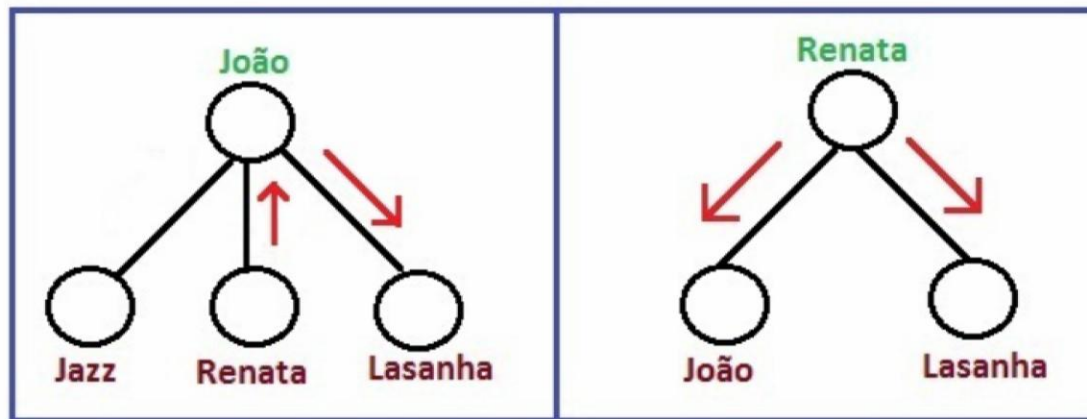
gosta(joão, x), gosta(renata, x).

~~X = Jazz~~
X = Renata

ESTRUTURAS DE CONTROLE

Backtracking

gosta(joão, jazz).
gosta(joão, renata).
gosta(joão, lasanha).
gosta(renata, joão).
gosta(renata, lasanha).



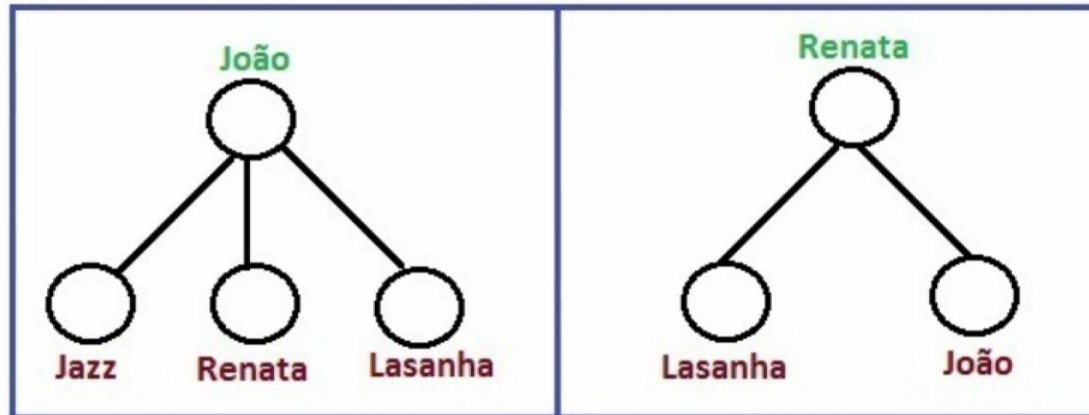
gosta(joão, x), gosta(renata, x).

~~X = Jazz~~
~~X = Renata~~ X = Lasanha

ESTRUTURAS DE CONTROLE

Backtracking para o Cut (!)

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, lasanha).  
gosta(renata, joão).
```

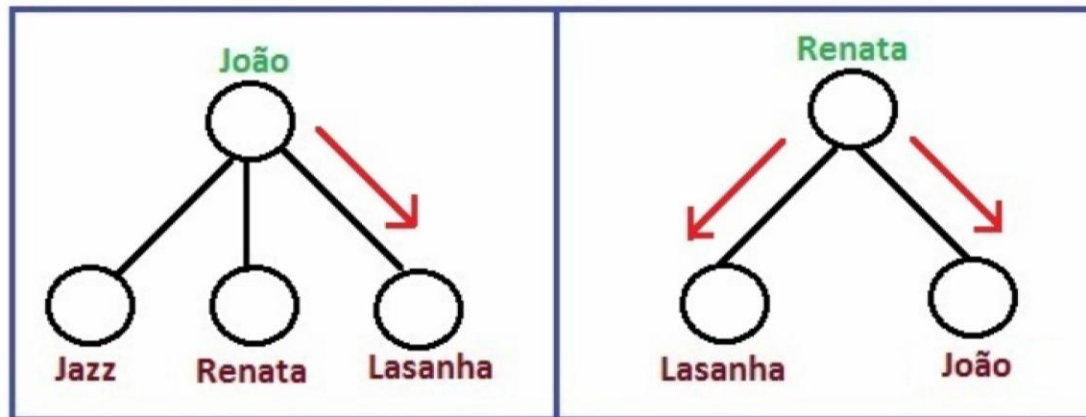


```
gosta (joão, x) , gosta (renata, x).
```

ESTRUTURAS DE CONTROLE

Cut (!)

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, lasanha).  
gosta(renata, joão).
```



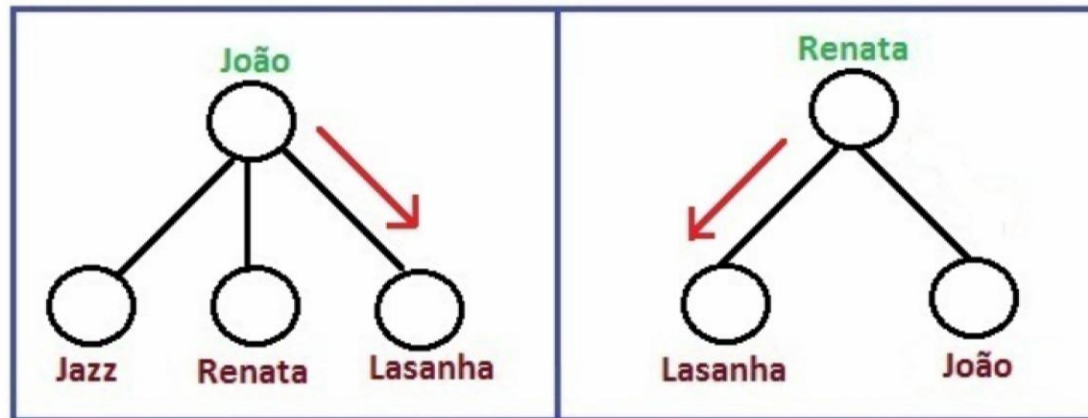
$gosta(joão, x), gosta(renata, x).$

$X = Lasanha$

ESTRUTURAS DE CONTROLE

Cut (!)

```
gosta(joão, jazz).  
gosta(joão, renata).  
gosta(joão, lasanha).  
gosta(renata, lasanha).  
gosta(renata, joão).
```



`gosta(joão, x), gosta(renata, x), !.` **X = Lasanha**



Obrigado!