FUNDAMENTOS MATEMÁTICOS PARA COMPUTAÇÃO

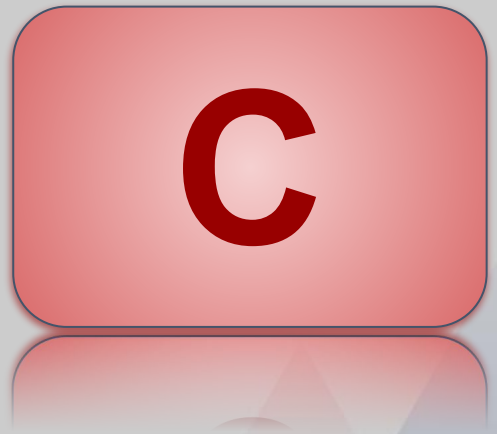
Programação Lógica

SUMÁRIO

- Linguagens Procedimentais e Declarativas
- Prolog
- Cláusulas de Horn
- Recorrência

Linguagens procedimentais e declarativas

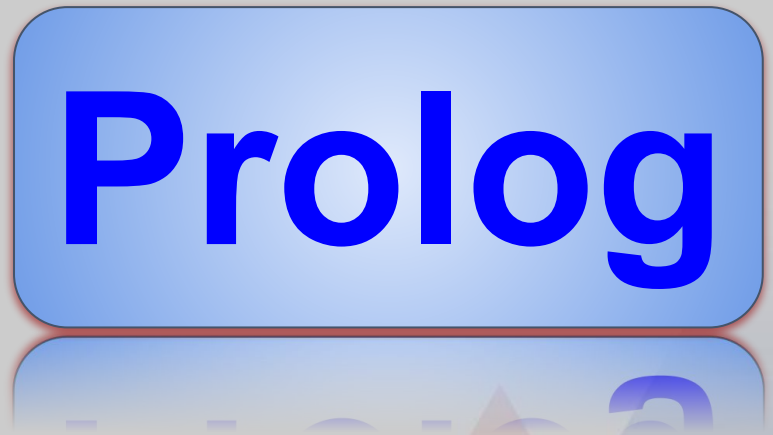
- Linguagem procedimental
  - Instruções para executar o algoritmo que resolverá o problema.
  - O computador é instruído a resolver o problema passo a passo.



Linguagens procedimentais e declarativas

- Linguagem declarativa
  - Baseia-se na lógica de predicados.
  - Utiliza regras de inferência, permitindo ao usuário obter conclusões a partir das hipóteses.

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



Prolog

PROgramming in LOGic - Programando em Lógica

Banco de dados Prolog: declarações que constituem um programa em Prolog.

- **Fatos**
- **Regras**

Prolog

Fatos: definem predicados para itens em algum conjunto universo.

Exemplo:

- Planta(x): x é planta
- Animal(x): x é animal
- seAlimenta(x,y): x se alimenta de y

Prolog

animal(urso)	animal(veado)	seAlimenta(urso, peixe)
animal(peixe)	planta(grama)	seAlimenta(urso, raposa)
animal(raposa)	planta(flores)	seAlimenta(veado, grama)

Consultas:

?seAlimenta(veado, grama)

sim

?seAlimenta(urso, coelho)

não

Consultas:

?seAlimenta(urso, X)

peixe

raposa

Prolog

animal(urso)	animal(veado)	seAlimenta(urso, peixe)
animal(peixe)	planta(grama)	seAlimenta(urso, raposa)
animal(raposa)	planta(flores)	seAlimenta(veado, grama)

Consultas:

?seAlimenta(X, Y) e planta(Y)

?seAlimenta(urso, peixe) e planta(peixe)

V **F**

?seAlimenta(veado, grama) e planta(grama)

V **V**

Prolog

Regras: descrição de um predicado por meio de um condicional.

Exemplo:

presa(X) <= seAlimenta(Y, X) e animal(X)
presa(X): x é uma presa.

[animal(X)]∧[seAlimenta(Y,X)] →[presa(X)]

?presa(X)

peixe
raposa

Prolog

animal(urso)	animal(veado)	seAlimenta(urso, **peixe**)
animal(**peixe**)	planta(grama)	seAlimenta(urso, **raposa**)
animal(**raposa**)	planta(flores)	seAlimenta(veado, grama)

Exemplo:

presa(X) <= seAlimenta(Y, X) e animal(X)

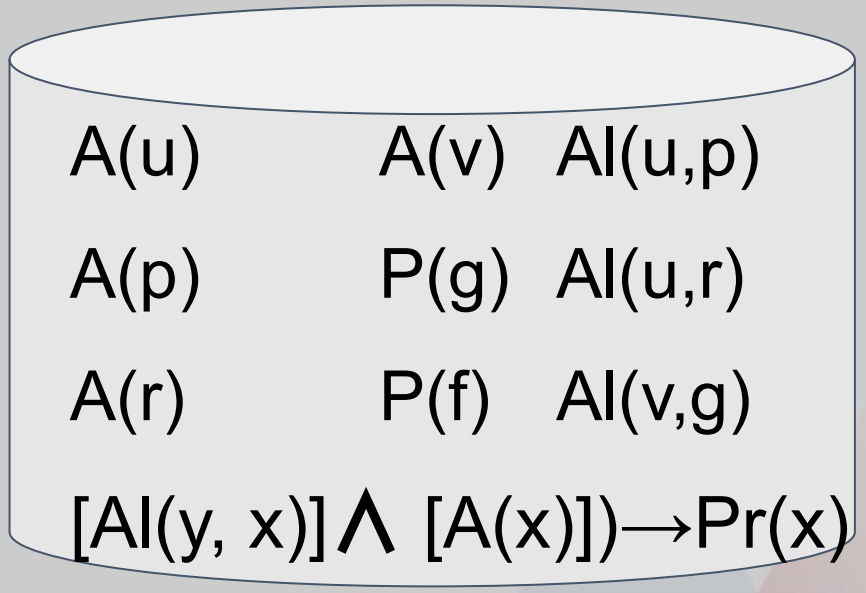
?presa(X)

peixe

raposa

Cláusulas de Horn

Trata-se de uma fbf formada por predicados ou negação de predicados conectada por disjunções, onde no máximo um predicado não esteja negado.



- **Al(v, g)**

- **[Al(y, x)]′∨ [A(x)]′∨Pr(x)**
  ⇔([Al(y, x)] ∧ [A(x)])′∨Pr(x) **De Morgan**
  ⇔([Al(y, x)] ∧ [A(x)])→Pr(x) **Condicional**

Cláusulas de Horn

- A regra de resolução do Prolog busca um termo e sua negação para inferir uma cláusula de Horn de duas outras.

- A única regra de inferência usada pelo Prolog é chamada de **resolução**.

Cláusulas de Horn

Resolução

1. $A(a)$	Cláusula de Horn
2. $[A(a)]' \vee B(b)$	Cláusula de Horn
3. $B(b)$	Resolução

O mesmo que

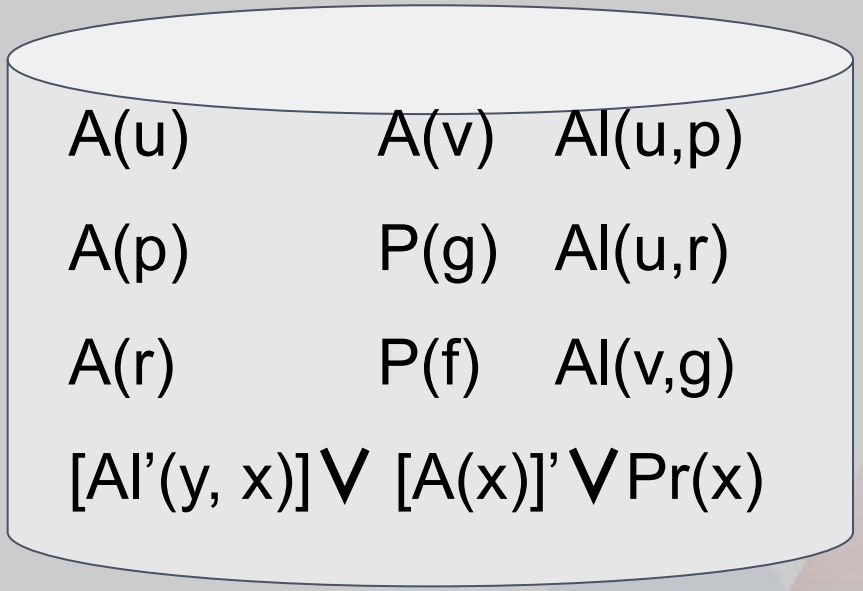
1. $A(a)$	
2. $A(a) \rightarrow B(b)$	$[A(a)]' \vee B(b) \Leftrightarrow A(a) \rightarrow B(b)$
3. $B(b)$	1,2 modus ponens

Cláusulas de Horn

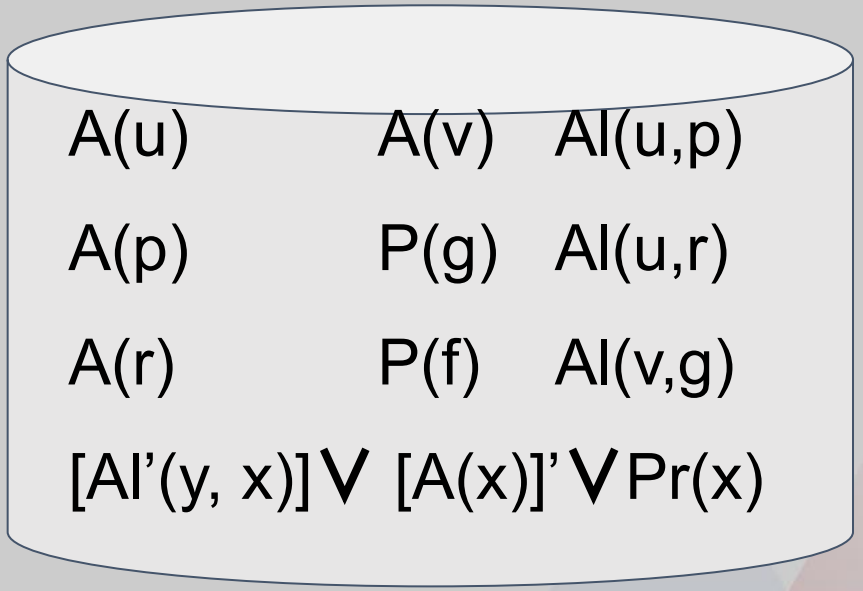
?presa(X)

Prolog busca uma regra que tenha o predicado Pr(x) como consequente.

[Al’(y, x)]∨ [A(x)]’∨Pr(x)



Cláusulas de Horn



?presa(X)

[Al'(y, x)]∨ [A(x)]'∨Pr(x)

Procuramos no banco por cláusulas que permitam a resolução.

Al(u,p)

Cláusulas de Horn

?presa(X)

[Al'(u,p)] ∨ [A(p)]' ∨ Pr(p)

Al(u,p)

Resolução

[A(p)]' ∨ Pr(p)

A(u)	A(v)	Al(u,p)
A(p)	P(g)	Al(u,r)
A(r)	P(f)	Al(v,g)
[Al'(y, x)] ∨	[A(x)]' ∨	Pr(x)

Al(u,p)

[Al(u,p)] → [A(p)]' ∨ Pr(p)

[A(p)]' ∨ Pr(p)

Cláusulas de Horn

?presa(X)

[A(p)]’ ∨ Pr(p)

A(p)

Resolução

Pr(p)

A(u)	A(v)	Al(u,p)
A(p)	P(g)	Al(u,r)
A(r)	P(f)	Al(v,g)
[Al’(y, x)] ∨	[A(x)]’ ∨	Pr(x)

A(p)

[A(p)]→ Pr(p)

Pr(p)

Cláusulas de Horn

caçado(X) <= presa(X)

Pr(x)→Ca(x)⇔[Pr(x)]′ ∨ Ca(x)

[Pr(x)]′ ∨ Ca(x)

[Al’(y, x)]∨ [A(x)]’∨Pr(x)

[Al(y, x)]′ ∨ [A(x)]′ ∨ Ca(x)

A(u)	A(v)	Al(u,p)
A(p)	P(g)	Al(u,r)
A(r)	P(f)	Al(v,g)

[Al’(y, x)]∨ [A(x)]’∨Pr(x)

[Pr(x)]’∨Ca(x)

Cláusulas de Horn

$[Pr(x)]' \vee Ca(x)$ $\quad Pr(x) \rightarrow Ca(x)$

$[Al(y, x)]' \vee [A(x)]' \vee Pr(x)$ $\quad ([Al(y, x)] \wedge [A(x)])' \rightarrow Pr(x)$

$[Al(y, x)]' \vee [A(x)]' \vee Ca(x)$ $\quad ([Al(y, x)] \wedge [A(x)])' \rightarrow Ca(x)$

$A \rightarrow B$ e $B \rightarrow C$, temos

$A \rightarrow C$

Recorrência

As regras no Prolog são condicionais com antecedentes no lado direito das regras.
As regras podem depender dos fatos:

presa(X) <= seAlimenta(Y, X) e animal(X)

Ou de outras regras:

caçado(X) <= presa(X)

Recorrência

O antecedente de uma regra pode depender da própria regra

nacadeiaalimentar(X, Y) <= seAlimenta(X, Y)

nacadeiaalimentar(X, Y) <=

seAlimenta(X, Z) e nacadeiaalimentar(Z, Y)

Uma definição na qual o item que está sendo definido faz parte da definição é chamada de uma definição recorrente ou recursiva.

Recorrência

presa(X) <= seAlimenta(Y, X) e animal(X)

seAlimenta(urso, peixe)
seAlimenta(peixe, peixinho)
seAlimenta(peixinho, alga)
seAlimenta(guaxinim, peixe)
seAlimenta(urso, guaxinim)
seAlimenta(urso, raposa)
seAlimenta(raposa, coelho)
seAlimenta(coelho, grama)
seAlimenta(urso, veado)
seAlimenta(veado, grama)
seAlimenta(lince, veado)

animal(urso)
animal(peixe)
animal(peixinho)
animal(guaxinim)ani
mal(raposa)
animal(coelho)
animal(veado)
animal(lince)
planta(grama)
planta(alga)

Recorrência

presa(X) <= seAlimenta(Y, X) e animal(X)
nacadeiaalimentar(X, Y) <= seAlimenta(X, Y)
nacadeiaalimentar(X, Y) <= seAlimenta(X, Z) e
nacadeiaalimentar(Z, Y)

nacadeiaalimentar(urso, Y)
<= seAlimenta(urso, Z) e
nacadeiaalimentar(Z, Y)

Recorrência

nacadeiaalimentar(urso, Y)
<= seAlimenta(urso, Z) e nacadeiaalimentar(Z, Y)

seAlimenta(urso, peixe)
seAlimenta(peixe, peixinho)
seAlimenta(peixinho, alga)
seAlimenta(guaxinim, peixe)
seAlimenta(urso, guaxinim)
seAlimenta(urso, raposa)
seAlimenta(raposa, coelho)
seAlimenta(coelho, grama)
seAlimenta(urso, veado)
seAlimenta(veado, grama)
seAlimenta(lince, veado)

Recorrência

nacadeiaalimentar(urso, Y)
<= seAlimenta(uso, Z) e cadeiaalimentar(Z, Y)

seAlimenta(urso, peixe) e cadeiaAlimentar(peixe,Y)

seAlimenta(urso, guaxinim) e
cadeiaAlimentar(guaxinin,Y)
seAlimenta(urso, raposa) e
cadeiaAlimentar(raposa,Y)
seAlimenta(urso, veado) e
cadeiaAlimentar(veado,Y)

Recorrência

nacadeiaalimentar(urso, Y)
<= seAlimenta(urso, Z) e cadeiaalimentar(Z, Y)

cadeiaAlimentar(urso,peixe)
<= seAlimenta(urso, peixe) e cadeiaalimentar(peixe, Y)

cadeiaAlimentar(peixe,peixinho)
<= seAlimenta(peixe, peixinho) e
cadeiaalimentar(peixinho, alga)

cadeiaalimentar(peixinho, alga)
<= seAlimenta(peixinho, alga))

Os conceitos e exemplos apresentados nesses slides são baseados no conteúdo da seção 1.5 do material-base “Fundamentos Matemáticos para a Ciência da Computação”, J.L. Gersting, 7a edição, LTC editora.

FUNDAMENTOS MATEMÁTICOS PARA COMPUTAÇÃO

Programação Lógica