UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC511 – Lógica de Predicados (2021.2)
Prof. Msc. Thais Oliveira Almeida

AULA 10:

PROLOG

Aspectos Gerais

- O nome Prolog é um acrônimo para *programmation en logique*;
- Linguagem enquadrada nos paradigmas lógico e declarativo da programação;
- Projetada, em 1972, por Alain Colmerauer e Philippe Roussel na França;
- Sua semântica foi formalizada por Robert Kowalski baseada no conceito das *cláusulas de Horn*;
- Desenvolvida para criar programas de tradução de linguagens naturais como português, francês e inglês.

Aspectos Gerais

- Tem na lógica um formalismo conveniente para representar e processar o conhecimento de maneira natural;
- Linguagem não-numérica, orientada a símbolos;
- É adequada à solução de problemas envolvendo objetos e suas relações;
- Tem sido aplicada em inteligência artificial, redes de computadores, educação, base de dados e sistemas paralelos.

Porque estudar Prolog?

- Há uma preferência por serviços de especificação no mercado de trabalho;
- Pode ser vista como uma linguagem de programação e de especificação;
- Eficaz na elaboração de ambientes e interfaces computacionais para seres humanos;
- Permite definir e estender sistemas reflexivos, utilizados em robótica;
- É adequado para a descrição do mundo real com todos os seus aspectos e sutilezas.

Características

- A programação limita-se a fornecer uma descrição do problema que se pretende computar;
- A execução de um programa em Prolog é efetivamente a prova de um dado teorema;
- Obtém respostas alternativas através de *backtracking*;
- Suporta código recursivo e iterativo, dispensando o uso de mecanismos como while, for e repeat;
- Apesar do longo tempo de desenvolvimento, ainda não é uma linguagem portável.

Cláusula de Horn

- É uma implicação cujo antecedente é uma conjunção de fórmulas atômicas e cujo conseqüente consiste em, no máximo, uma fórmula atômica.

Lógica:	Prolog:
$(\alpha_1 \wedge \ldots \wedge \alpha_n) \longrightarrow \beta$	β :- $(\alpha_1, \ldots, \alpha_n)$

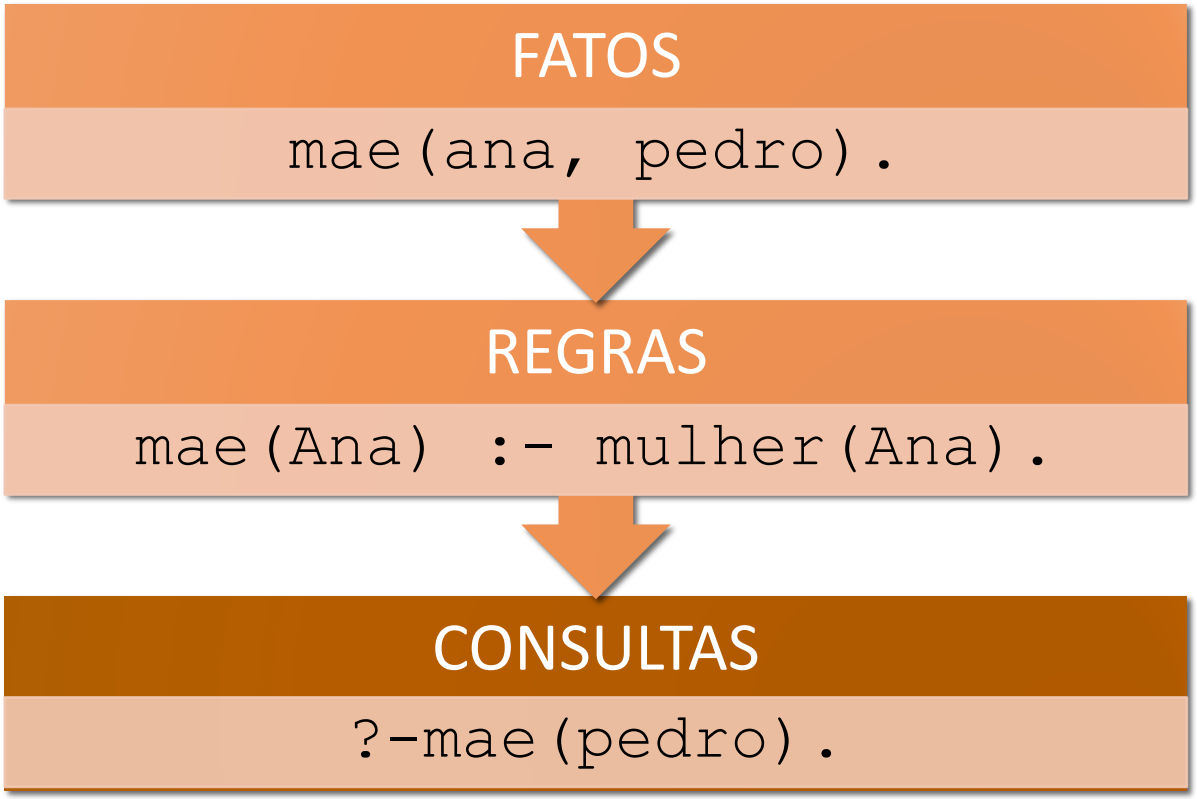
Backtracking

- A evolução da busca por soluções assume a do padrão da busca em profundidade em árvores;
- Quando a pesquisa falha ou é encontrado um nó terminal, o sistema retorna pelo mesmo caminho percorrido com a finalidade de encontrar soluções alternativas;
- O backtracking pode se tornar em uma fonte de ineficiência, uma vez que o programa pode executar passos tentando satisfazer objetivos que não contribuirão para a solução do problema.

Tipos de Dados

átomos	• `sapato, 'Sapato'`
variáveis	• `X, _X`
números inteiros	• `43, -35, 'a = 97`
números em ponto flutuante	• `43.0, .8e21`
strings	• `$ele foi ao cinema$, 'ele foi ao cinema'`
estruturas	• `nasceu(joca, natal)`
listas	• `[a, b, c]`

Conceitos Básicos



Operadores Básicos

pré-definidos	_		:-		?-		!		
não, e, ou lógicos	not		,		;				
aritmética	+	-	*	/	//	mod	^	is	
relacional	=	==	=:=	\==	=\=	<	=<	>	>=
outros	member(elemento, [ ])					\+			

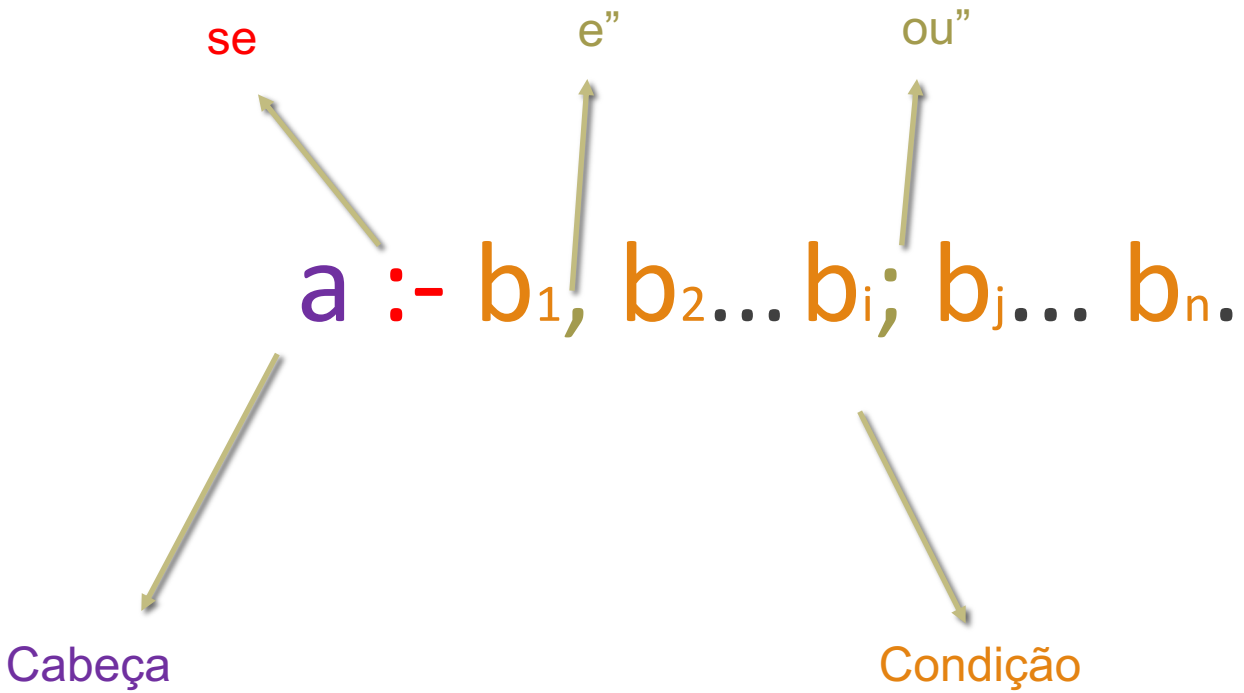
Aplicações

Escrita de compiladores	Prova de Teoremas	Redes de Computadores
Sistemas Baseados em Conhecimento	Sistemas de Bases de Dados	Sistemas Especialistas
Processamento de Linguagem Natural	Educação	Arquiteturas Não-Convencionais

Prolog

- Para resolução de problemas lógicos, podemos utilizar o Prolog como uma forma de facilitar o encontro de uma solução;
- Pelo fato de ser uma linguagem interpretada, pode ser facilmente incorporada a uma linguagem de programação que faça chamadas de execução de programas;
- Prolog é uma linguagem muito utilizada na área da pesquisa, desenvolvimento de projetos científico, vem tentando ser utilizada no mercado.

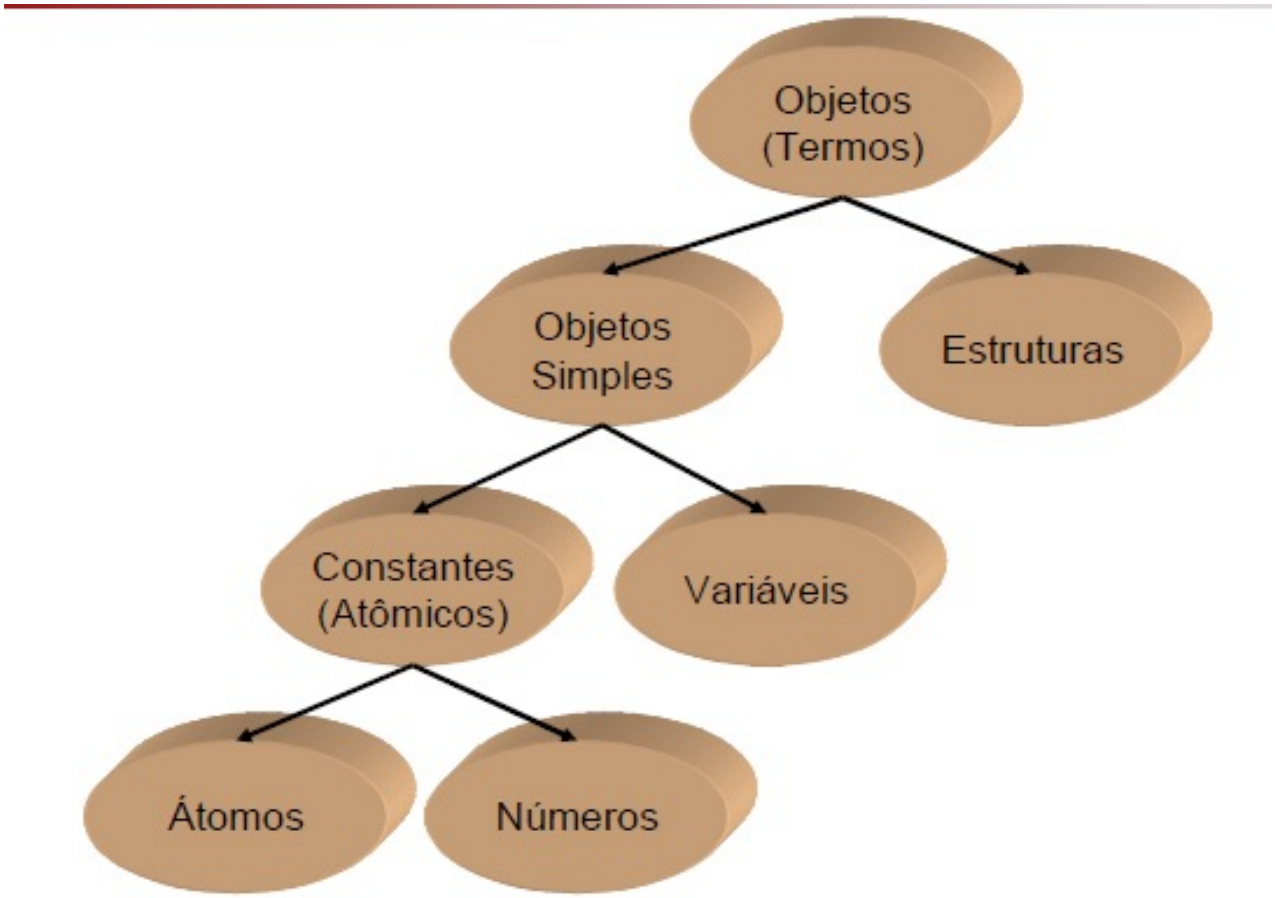
A Cláusula Prolog



Operadores Prolog

Linguagem Natural	Cálculo de Predicados	Programas Prolog
E	$\wedge$	,
OU	$\vee$	;
SE	$\leftarrow$	:-
NÃO	$\neg$	**not**

Objetos em Prolog



Fatos e Consultas

- *Fatos* servem para estabelecer um relacionamento existente entre objetos de um determinado contexto de discurso. Por exemplo:
  - pai(adão,cain).
- Para recuperar informações de um programa lógico, usamos *consultas*. Uma consulta pergunta se uma determinado relacionamento existe entre objetos. Por exemplo, a consulta:
  - **?-** pai(adão,cain).
  - Deverá retornar um valor *true*

Fatos e Consultas

- Sintaticamente, fatos e consultas são muito similares. A diferença é que fatos são agrupados no arquivo que constitui o programa, enquanto consultas são sentenças digitadas no prompt (**?-**) do interpretador Prolog.

Números

- Números usados em Prolog incluem números inteiros e reais.

Operadores Aritméticos	
adição	+
subtração	-
multiplicação	*
divisão	/
divisão inteira	//
resto divisão inteira	mod
potência	**
atribuição	is

Operadores Relacionais	
X > Y	X é maior do que Y
X < Y	X é menor do que Y
X >= Y	X é maior ou igual a Y
X =< Y	X é menor ou igual a Y
X =:= Y	X é igual a Y
X = Y	X unifica com Y
X =\= Y	X é diferente de Y

Números

- O operador **=** tenta **unificar** apenas.
  - **?-** X = 1 + 2.
  - **X = 1 + 2**
- O operador **is** **força a avaliação aritmética**.
  - **?-** X is 1 + 2.
  - **X = 3**
- Se a variável à esquerda do operador **is** já estiver instanciada, Prolog apenas compara o valor da variável com o resultado da expressão à direita de is
  - **?-** X = 3, X is 1 + 2.
  - **X = 3**
  - **?-** X = 5, X is 1 + 2.
  - **no**

Variável Anônima

- Quando uma variável aparece em uma única cláusula, não é necessário utilizar um nome para ela;
- Utiliza-se a variável **anônima**, que é escrita com um simples caracter '_'. Por exemplo:
  - temfilho(X) :- progenitor(X,Y).
- Para definir **temfilho**, não é necessário o nome do filho(a);
- Assim, é o lugar ideal para a variável anônima:
  - temfilho(X) :- progenitor(X,_).

Variável Anônima

- Cada vez que um *underscore* '_' aparece em uma cláusula, ele representa uma nova variável anônima.
- Por exemplo:
  - alguém_tem_filho :- progenitor(_,_).

Equivale à:

  - alguém_tem_filho :- progenitor(X,Y).

Que é bem diferente de:

  - alguém_tem_filho :- progenitor(X,X).

Variável Anônima

- Quando utilizada em uma pergunta, seu valor não é mostrado. Por exemplo, se queremos saber quem tem filhos mas sem mostrar os nomes dos filhos, podemos perguntar:
  - **?-** progenitor(X,_).

Estruturas

- Objetos estruturados (ou simplesmente estruturas) são objetos de dados que têm vários componentes;
- Cada componente, por sua vez, pode ser uma estrutura;
- Por exemplo, uma data pode ser vista como uma estrutura com três componentes: dia, mês, ano;
- Mesmo possuindo vários componentes, estruturas são tratadas como simples objetos.

Estruturas

- De forma a combinar componentes em um simples objeto, deve-se escolher um **functor;**
- Um functor para o exemplo da data seria **data;**
- Então a data de 4 de maio de 2017 pode ser escrita como:
  - data(4,maio,2017)

Estruturas

- Qualquer dia em maio pode ser representado pela estrutura:
  - data(Dia,maio,2017)
- Note que **Dia** é uma variável que pode ser instanciada a qualquer objeto em qualquer momento durante a execução;
- Sintaticamente, todos objetos de dados em Prolog são termos. Por exemplo, são termos:
  - maio
  - data(4,maio,2017)

Listas

- Lista é uma das estruturas mais simples em Prolog, muito comum em programação não numérica;
- Ela é uma sequência ordenada de elementos;
- Uma lista pode ter qualquer comprimento;
- Por exemplo uma lista de elementos tais como **maria, tênis, pedro** pode ser escrita em Prolog como:
  - [maria, tênis, pedro]

Listas

- O uso de colchetes é apenas uma melhoria da notação, pois internamente listas são representadas como árvores, assim como todos objetos estruturados em Prolog;
- Para entender a representação Prolog de listas, é necessário considerar dois casos:
  - A lista é vazia, escrita como **[ ]** em Prolog;
  - Uma lista (não vazia) consiste:
    - No primeiro item, chamado ***cabeça (head)*** da lista.
    - Na parte restante da lista, chamada ***cauda (tail).***

Listas

- No exemplo [maria, tênis, pedro]:
  - **maria** é a Cabeça da lista
  - **[tênis, pedro]** é a Cauda da lista
- A cabeça de uma lista pode ser qualquer objeto (inclusive uma lista); a cauda tem que ser uma lista;
- A Cabeça e a Cauda são então combinadas em uma estrutura pelo functor especial **.**
  - **.**(Cabeça, Cauda)
- Como a Cauda é uma lista, ela é vazia ou ela tem sua própria cabeça e sua cauda.

Listas

- **?-** Lista1 = [a,b,c],

  Lista2 = .(a,.(b,.(c,[]))).

  Lista1 = [a, b, c]

  Lista2 = [a, b, c]

- **?-** Hobbies1 = .(tênis, .(música,[])),

  Hobbies2 = [esqui, comida],

  L = [ana,Hobbies1,pedro,Hobbies2].

  Hobbies1 = [tênis,música]

  Hobbies2 = [esqui,comida]

  L = [ana, [tênis,música], pedro, [esqui,comida]]

Unificação em Listas

Lista1	Lista2	Lista1 = Lista2
[mesa]	[X|Y]	X=mesa Y=[ ]
[a,b,c,d]	[X,Y|Z]	X=a Y=b Z=[c,d]
[[ana,Y]|Z]	[[X,foi],[ao,cinema]]	X=ana Y=foi Z=[[ao,cinema]]
[ano,bissexto]	[X,Y|Z]	X=ano Y=bissexto Z=[ ]
[ano,bissexto]	[X,Y,Z]	Não unifica

Representação Textual

Considere o seguinte texto:

"*João nasceu em Pelotas e Jean nasceu em Paris. Paris fica na França, enquanto que Pelotas fica no Rio Grande do Sul. Mas só é gaúcho quem nasceu no Rio Grande do Sul*."

```
nasceu(joão,pelotas).
nasceu(jean,paris).

fica(paris,França).
fica(pelotas,rs).

gaúcho(X):-
        nasceu(X,Y),
        fica(Y,rs).

?- gaúcho(X).
X=joão;
```

SWI-Prolog

- Download: http://www.swi-prolog.org/Download.html
- Manual: https://www.swi-prolog.org/pldoc/doc_for?object=manual

Exercício 1

- Usando os predicados avião/1, helicóptero/1, carro/1, moto/1, veículo/1, voa/1 e aeronave/1,codifique um programa em Prolog para representar as informações a seguir:
- A420 é um avião, H390 é um helicóptero, C150 é um carro e M320 é uma moto.
- Aviões, helicópteros, carros e motos são veículos de transporte.
- Aviões e helicópteros voam.
- Aeronaves são veículos de transporte que voam.

Exercício 2

- Considere a seguinte base de conhecimento "pais.pl" abaixo, que representa a área e a população dos países.

```
% país(Nome, Área, População)
pais(brasil, 9, 130).
pais(china, 12, 1800).
pais(eua, 9, 230).
pais(india, 3, 450).
```

Exercício 2

- O programa anterior representa uma tabela que relaciona a cada país sua área em Km2 e sua população em milhões de habitantes. Note que a linha iniciando com % é um comentário e serve apenas para fins de documentação. Considerando que a densidade demográfica de um país é calculada pela divisão da população/área, escreva uma query que permita:

a) Determinar a densidade demográfica do Brasil;

b) Determinar qual a diferença entre a população da China e da Índia?