Laboratório de PROLOG #2/3

Disciplina de Modelos de Linguagens de Programação

Modelo lógico

Tópicos da aula

- Prolog: recapitulação
- Manipulação de listas

Prolog: recapitulando...

- Prolog não tem tipos de dados!
- Ela é constituída de "termos"

Os principais "tipos" de *termos* são:

- Átomos:
  - constantes alfanuméricas: leonardo, 'Porto Alegre'
  - constantes numéricas: 1, 12.12
- Variáveis: X, Cidades, _ruas, _123abc
- Termos funcionais (cláusulas):
  - fatos/consultas: pessoa(pedro, 22, masculino).
  - regras: capital_pais(X,Y) :- pais(X) , cidade(Y) , capital(X, Y)

Prolog: recapitulando...

- **Mecanismos de computação lógica:**
  - Base de fatos e regras
  - Processo de inferência: identificação de uma cadeia de regras e/ou fatos que conecte o objetivo (consulta) a um ou mais fatos na base, usando a unificação
  - Processo de unificação: técnica de casamento de padrões que combina dois elementos

Unificação: exercícios

- No interpretador, teste as expressões:

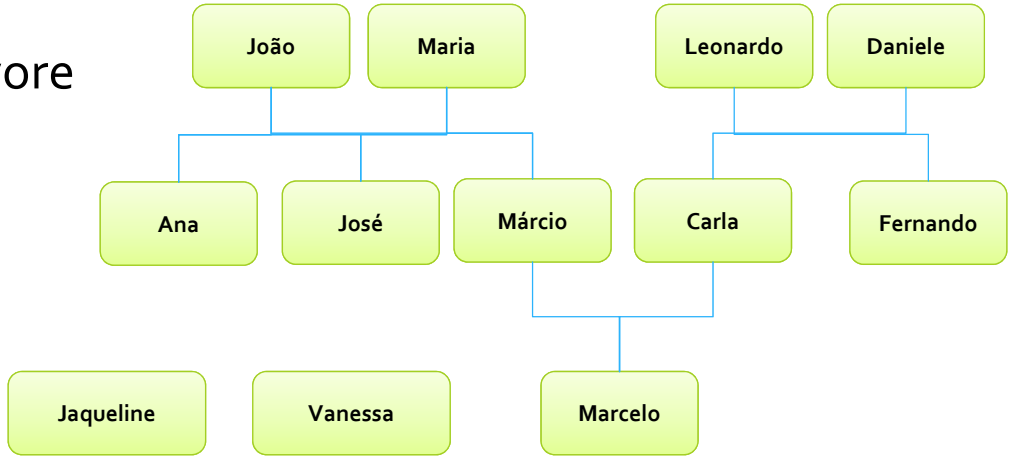
```
?- X = 1.
?- X = X.
?- 1 = 1.
?- p(1) = p(1).
?- p(X, 1) = p(2, Y).
```

Prolog: recapitulando...

- O processo de unificação combina dois elementos, levando em conta as seguintes regras:
  - constantes: se forem exatamente iguais
    e.g., leandro = leandro
  - variáveis: se um lado for constante e o outro variável
    e.g., Nome = leandro
  - cláusulas: se possuírem o mesmo predicado
    e.g., cidade(X) = cidade('Porto Alegre')

Exercício de recapitulação

- Usar arquivo com dados da árvore genealógica, fornecido pelo professor
- Consultar:
  - O avô de Marcelo.
  - Os pais de José.
  - Os tios de Márcio.
  - Fernando tem irmã?
  - Quem gosta de boliche e de cinema?
  - Há algo que Maria e Marcelo gostem? O que?
  - Quais pessoas gostam de outras pessoas?
  - Quais são os antepassados do Marcelo?
  - Quais são os parentes da Carla?



Prolog: recapitulando...

- **Mecanismos de computação lógica:**
  - Backtracking:
    - recuo que ocorre no processo de inferência quando uma subconsulta falha, buscando por combinações alternativas
  - Reversibilidade:
    - toda transformação é reversível (pode ser desfeita)
    - os parâmetros de uma cláusula podem se comportar como entradas ou saídas
    - computação bidirecional

Mecanismo de resolução: exemplo

OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Mecanismo de resolução: exemplo

OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos

capital_pais(X,Y)

pais(X), cidade(Y), capital(X, Y).

FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Sempre casa com a primeira regra que encontra!

Mecanismo de resolução: exemplo

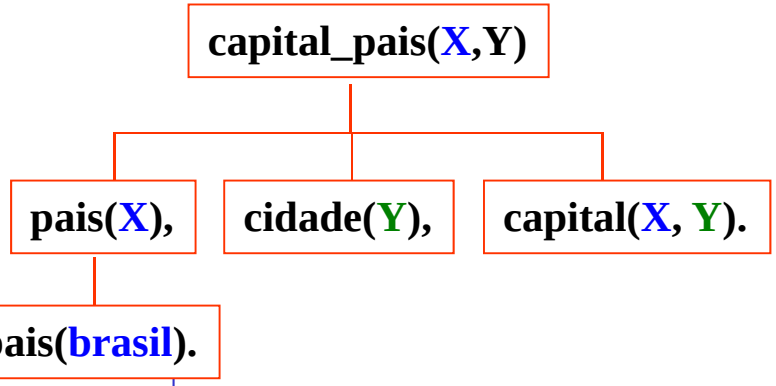
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Mecanismo de resolução: exemplo

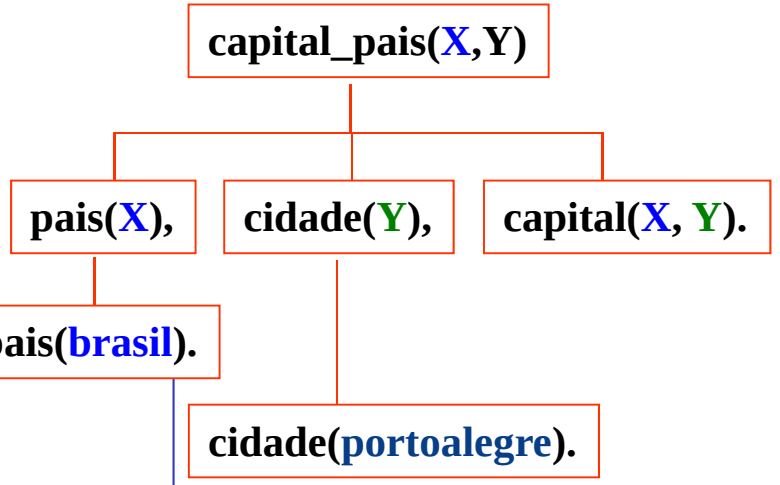
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Mecanismo de resolução: exemplo

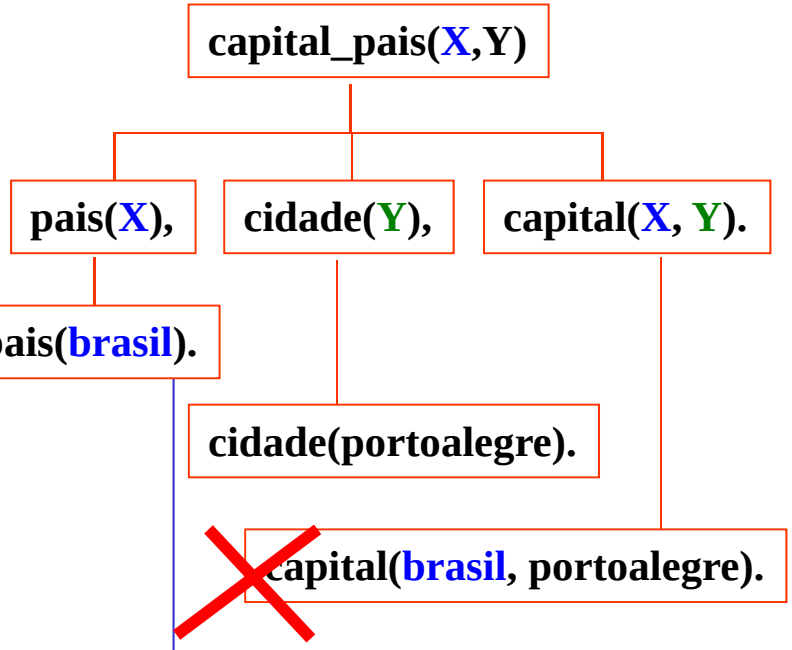
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Mecanismo de resolução: exemplo

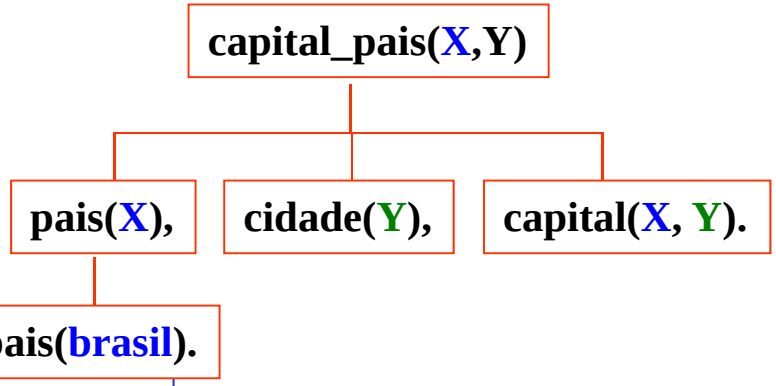
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :- pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :- estado(X), cidade(Y), capital(X, Y).

BACKTRACKING!

Mecanismo de resolução: exemplo

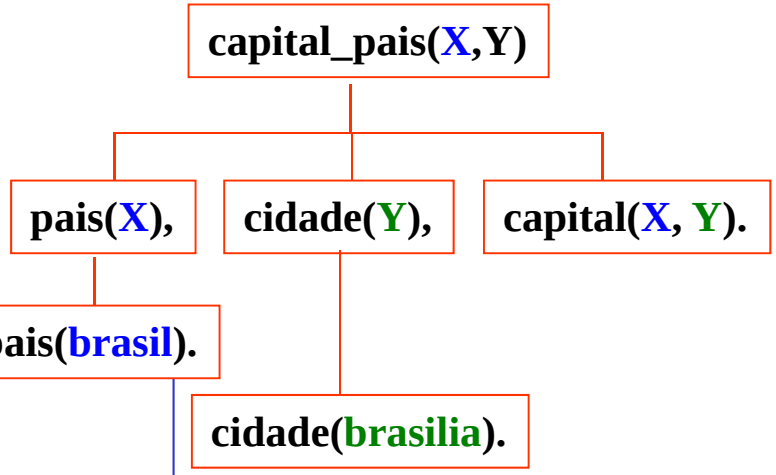
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

```
cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :-  pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :-  estado(X), cidade(Y),
capital(X, Y).
```

Mecanismo de resolução: exemplo

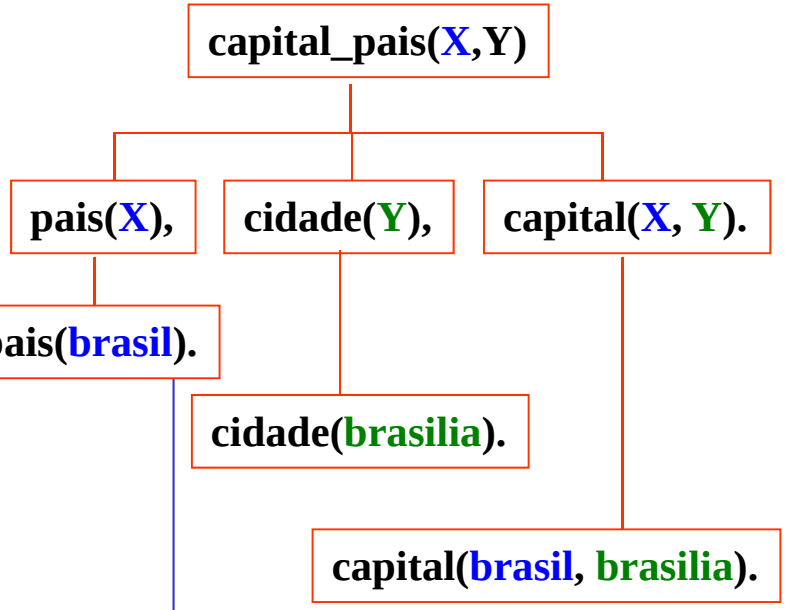
OBJETIVO

Seja a consulta: ? capital_pais (brasil, K).

Resposta esperada: K=brasilia

Busca: UNIFICAÇÃO do objetivo e todos os sub-objetivos

sub-objetivos



FATOS E REGRAS

cidade(portoalegre).
cidade(brasilia).
pais(brasil).
capital(brasil, brasilia).
estado(rgs).
capital(rgs, portoalegre).
capital_pais(X,Y) :- pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :- estado(X), cidade(Y), capital(X, Y).

A busca prossegue até esgotar todas as possibilidades de unificação

Prolog: operadores

Operador	Significado
,	And
;	Or
=	Unificação
\=	Negação da unificação
==	Teste de identidade
\==	Negação da identidade
=:=	Igualdade aritmética
> >= < =<	Operadores relacionais
:=	Condicional

Atenção:

- Operador "=" representa unificação:
  - `+(2, D) = +(E, 2).`
    `D = 2`
    `E = 2`
    `yes`
  - `2+2=1+3.`
    `no`
- Operador "=:=" representa comparação:
  - `1+2=:=2+1.`
    `yes`

Manipulação de listas

MLP - PROLOG

Listas

- Análise do arquivo "genealogia3.pl"
  - Verifique as regras de geração de listas (antepassados e gostos) que utilizam "setof" e "findall", realizando consultas com elas.
- Elabore regras para gerar:
  - Lista de descendentes de uma pessoa
  - Lista das pessoas que gostam de determinada coisa ou pessoa

Listas: representação

- Listas são estruturas compostas por constantes, variáveis ou quaisquer outros termos, incluindo outras listas.
- Exemplos:

  Lista vazia (átomo): [ ]

  Lista simples: [canario, bentevi, periquito]

  Lista de listas: [[azul,preto,branco], [vermelho,branco]]

Listas, variáveis e E/S

- Pode-se associar uma lista a uma variável (unificação):
  - Lista1 = [1, 2, 3].
- Listas podem ser lidas e impressas:
  - Lista2 = [1, 2, 3], write(Lista2).
  - read(Lista), write(Lista).

OBS: na verdade, read e write são comandos que leem qualquer coisa!

Listas e cláusulas

- Cláusulas de listas: o argumento é uma lista

  lista([10, 20, 30]).

- Lista de cláusulas: os argumentos são cláusulas

  lst_clausulas ([gosta(ana, pedro), gosta (pedro, ana)]).

Separando a cabeça da cauda...

- **[H | T] → lista com *head* H e *tail* T**

?- [H|T]=[1,2,3].

H = 1

T = [2,3]

?- [Parte1|Parte2]=[brasil,uruguai,argentina,paraguai].

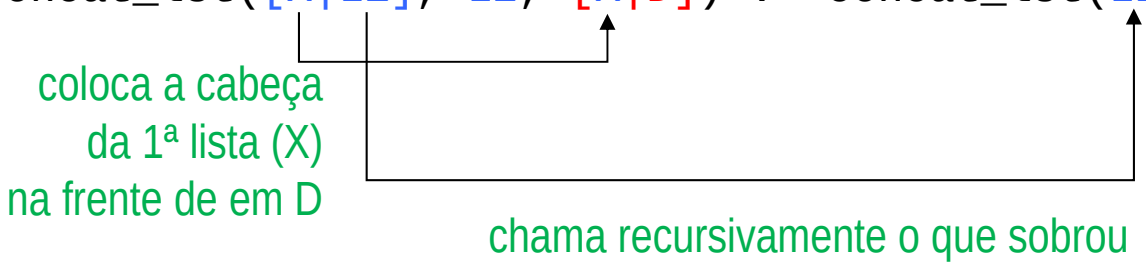
Parte1 = brasil

Parte2 = [uruguai,argentina,paraguai]

Listas: concatenação

- Um predicado para simular a concatenação de listas:
  - Sintaxe: `concat_lst(lst1, lst2, lst_destino)`
  - Objetivo: `concatenar lst1 com lst2 e colocar em lst_destino`
  - Definição:

```
concat_lst([], L2, L2):-!. // critério de parada: retorna L2 se 1ª lista está vazia
concat_lst([X|L1], L2, [X|D]) :- concat_lst(L1, L2, D).
```



Listas: operações básicas

- Para o tratamento de listas, algumas operações básicas devem ser definidas, tais como:
  - adicionar e retirar elementos de uma lista
  - determinar se um elemento pertence a uma lista
  - somar elementos de uma lista, etc.

Listas: verificando pertinência

- Algoritmo para determinar se um elemento X pertence a uma lista L:
  - **X é membro de L se:**
    1. X é cabeça de L; ou
    2. X é membro do corpo de L.
  - **Em Prolog:**

```
// se estiver na cabeça, retorna true
membro(Elem, [Elem| _ ]).
// se não, recursivamente testa se
// elemento é membro da cauda
membro(Elem, [ _ |Cauda]) :- membro(Elem, Cauda).
```

Exercícios

- Crie uma regra para imprimir na tela os elementos que fazem parte de uma lista.
- Crie uma regra para retornar o último elemento de uma lista.
- Crie uma regra que inverta os elementos de uma lista.
- Crie uma regra que insira um elemento em uma lista ordenada.
- Crie uma regra que remova um elemento de uma lista