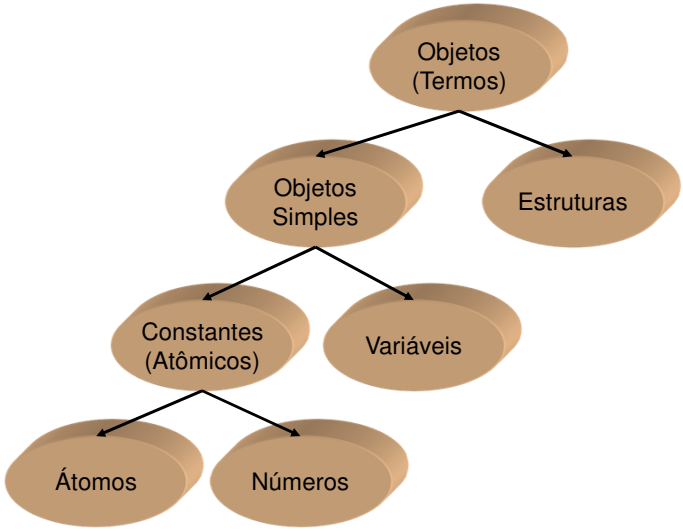
Sintaxe e Semântica de Programas Prolog

Thiago A. S. Pardo
Solange Rezende

Inteligência Artificial

1

Objetos em Prolog



2

Átomos

- São cadeias compostas pelos seguintes caracteres:
  - letras maiúsculas: A, B, ..., Z
  - letras minúsculas: a, b, ..., z
  - dígitos: 0, 1, ..., 9
  - caracteres especiais: + - * / < > = : . & _ ~
- Podem ser construídos de três maneiras:
  - **cadeias de letras, dígitos e o caractere** '_', começando com uma letra minúscula: ana, nil, x25, x_25, x_25AB, x_, x__y, tem_filhos, tem_um_filho
  - cadeias de **caracteres especiais**: <--->, =====>, ..., .:., ::=
  - cadeias de **caracteres entre aspas simples**: 'Abraão', 'América_do_Sul', 'América_Latina'

3

Números

- Números usados em Prolog incluem números inteiros e números reais

Operadores Aritméticos Predicado pré-definido **is**	
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

4

Números

- Números usados em Prolog incluem números inteiros e números reais

Operadores Aritméticos Predicado pré-definido **is**	
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

Há variações entre prologs, por exemplo: =\= e \==

Números

- O **operador =** apenas **tenta unificar**
  ?- 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**

6

Comparação: Aritmética e Prolog

Aritimética	Prolog
2 + 3 = 5 3 x 4 = 12 5 – 3 = 2 3 – 5 = -2 4 div 2 = 2 1 é o resto quando 7 é dividido por 2	?- 5 is 2+3. ?- 12 is 3*4. ?- 2 is 5-3. ?- -2 is 3-5. ?- 2 is 4/2. ?- 1 is mod(7,2).

7

Exemplos

```
?- 10 is 5+5.
yes

?- 4 is 2+3.
no

?- X is 3 * 4.
X=12
yes

?- R is mod(7,2).
R=1
yes
```

8

Usando o operador =

```
?- X = 3 + 2.
```

9

Usando o operador =

```
?- X = 3 + 2.
X = 3+2
yes

?-
```

10

Usando o operador =

```
?- X = 3 + 2.
X = 3+2
yes

?- 3 + 2 = X.
```

11

Usando o operador =

```
?- X = 3 + 2.
X = 3+2
yes

?- 3 + 2 = X.
X = 3+2
yes

?-
```

12

Usando o predicado is/2

```
?- X is 3 + 2.
```

13

Usando o predicado is/2

```
?- X is 3 + 2.
X = 5
yes

?-
```

14

Usando o predicado is/2

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
```

15

Usando o predicado is/2

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Argumentos não estão instanciados
    corretamente

?-
```

16

Usando o predicado is/2

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: Arguments are not sufficiently instantiated
    (Argumentos não estão instanciados corretamente)

?- Resultado is 2+2+2+2+2.
```

17

Usando o predicado is/2

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: Arguments are not sufficiently instantiated
    (Argumentos não estão instanciados corretamente)

?- Resultado is 2+2+2+2+2.
Resultado = 10
yes

?-
```

18

Definição de Predicados Aritméticos

```
soma_Tres_e_Dobra(X, Y):-
   Y is (X+3) * 2.
```

19

Definição de Predicados Aritméticos

```
soma_Tres_e_Dobra(X, Y):-
   Y is (X+3) * 2.
```

```
?- soma_Tres_e_Dobra (1,W).
W=8
yes

?- soma_Tres_e_Dobra (2,W).
W=10
yes
```

20

Exercício

- Dadas as distâncias do sol aos planetas do Sistema solar em milhões de milhas:

Mercúrio	**36**
Venus	**67**
Terra	**93**
Marte	**141**
Jupiter	**484**
Saturno	**886**
Urano	**1790**
Netuno	**2800**

- Escreva o programa: distancia_planetas(P1, P2, D) que encontra a distância entre dois planetas quaisquer.

21

Variáveis

- São cadeias de letras, dígitos e caracteres '_', sempre começando com letra maiúscula ou com o caractere '_'
  - X, Resultado, Objeto3, Lista_Alunos, ListaCompras, _x25, _32
- O escopo de uma variável é dentro de uma mesma regra ou dentro de uma pergunta
- Isto significa que se a variável X ocorre em duas regras/perguntas, então são duas variáveis distintas
- Mas a ocorrência de X dentro de uma mesma regra/pergunta refere-se a mesma variável

22

Variáveis

- Uma variável pode estar:
  - Instanciada: quando a variável já referencia (está unificada a) algum objeto
  - Livre ou não-instanciada: quando a variável não referencia (não está unificada a) um objeto
  - Uma vez instanciada, somente Prolog pode torná-la não-instanciada através de seu mecanismo de inferência (não o programador)

23

Variável Anônima

- Quando não há necessidade de saber o valor da instanciação de uma variável, não é necessário utilizar um nome para ela
- Utiliza-se a variável **anônima**, que é escrita com um simples caractere '_'. Por exemplo
  - temfilho(X) :- progenitor(X,Y).
- Para definir **temfilho**, não é necessário o nome do filho(a)
- Assim, é o lugar ideal para utilizar a variável anônima:
  - temfilho(X) :- progenitor(X,_).

24

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).
- 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,_).

25

Semântica de Programas Prolog

- Considere a cláusula onde *p*, *q* e *r* são termos:
  - p :- q,r.
- Significado Declarativo:
  - *p* é verdadeiro se *q* e *r* são verdadeiros
  - *p* segue (é conseqüência) de *q* e *r*
  - De (a partir de) *q* e *r* segue *p*
- Significado Procedural:
  - Para resolver o problema *p*, resolva primeiro o sub-problema *q* e então o sub-problema *r*
  - Para satisfazer *p*, primeiro satisfaça *q* e então *r*
- A diferença entre os significados declarativo e procedural é que o último não apenas define as relações lógicas entre a cabeça da cláusula e as condições no corpo, mas também a ***ordem*** na qual as condições são executadas

26

Semântica de Programas Prolog

- O significado declarativo determina quando uma dada condição é verdadeira e, se for, para quais valores das variáveis ela é verdadeira
- O significado procedural determina como Prolog responde perguntas (interrogações)

27

Significado Declarativo

- Em geral, uma meta em Prolog é uma lista de condições separadas por vírgulas que é verdadeira se todas as condições na lista são verdadeiras para a mesma instanciação de variáveis
- A vírgula denota conjunção (**e**): todas devem ser verdadeiras:
  - x, y neste exemplo x **e** y devem ser ambos verdadeiros para *x,y* ser verdadeiro
- O ponto-e-vírgula denota disjunção (**ou**): qualquer uma das condições em uma disjunção tem que ser verdadeira
  - x;y neste exemplo basta que *x* (ou *y*) seja verdadeiro para *x;y* ser verdadeiro
- O operador **\+** denota a negação (**não**): é verdadeiro se o que está sendo negado não puder ser provado por Prolog
  - **\+x** é verdadeiro se **x** falha
- O predicado **true/0** sempre é verdadeiro
- O predicado **fail/0** sempre falha

28

Significado Declarativo

- A cláusula
  - **p :- q ; r.**

  é lida como: p é verdade se q é verdade **ou** r é verdade. É equivalente às duas cláusulas:
  - **p :- q.**
  - **p :- r.**
- A conjunção (**,**) tem precedência sobre a disjunção (**;**), assim a cláusula:
  - p :- q, r; s, t, u.

  é interpretada como:
  - p :- (q, r) ; (s, t, u).

  e tem o mesmo significado que:
  - p :- q, r.
  - p :- s, t, u.

29

Significado Procedural

```
grande(urso).          % Cláusula 1
grande(elefante).      % Cláusula 2
pequeno(gato).         % Cláusula 3
marrom(urso).          % Cláusula 4
preto(gato)            % Cláusula 5
cinza(elefante).       % Cláusula 6
escuro(Z) :-           % Cláusula 7
  preto(Z).
escuro(Z) :-           % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

30

Significado Procedural

```
grande(urso).        % Cláusula 1
grande(elefante).    % Cláusula 2
pequeno(gato).       % Cláusula 3
marrom(urso).        % Cláusula 4
preto(gato)          % Cláusula 5
cinza(elefante).     % Cláusula 6
escuro(Z) :-         % Cláusula 7
  preto(Z).
escuro(Z) :-         % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 1)
Pergunta inicial:
escuro(X), grande(X).

31

Significado Procedural

```
grande(urso).        % Cláusula 1
grande(elefante).    % Cláusula 2
pequeno(gato).       % Cláusula 3
marrom(urso).        % Cláusula 4
preto(gato)          % Cláusula 5
cinza(elefante).     % Cláusula 6
escuro(Z) :-         % Cláusula 7
  preto(Z).
escuro(Z) :-         % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 2)
Procure de cima para baixo por uma cláusula cuja cabeça unifique com a primeira condição da pergunta escuro(X)

32

Significado Procedural

```
grande(urso).          % Cláusula 1
grande(elefante).      % Cláusula 2
pequeno(gato).         % Cláusula 3
marrom(urso).          % Cláusula 4
preto(gato)            % Cláusula 5
cinza(elefante).       % Cláusula 6
escuro(Z) :-           % Cláusula 7
  preto(Z).
escuro(Z) :-           % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 2)
Procure de cima para baixo por uma cláusula cuja cabeça unifique com a primeira condição da pergunta escuro(X).
Cláusula 7 encontrada
escuro(Z) :- preto(Z).
Troque a primeira condição pelo corpo instanciado da cláusula 7, obtendo a nova lista de condições:
preto(X), grande(X).

33

Significado Procedural

```
grande(urso).          % Cláusula 1
grande(elefante).      % Cláusula 2
pequeno(gato).         % Cláusula 3
marrom(urso).          % Cláusula 4
preto(gato)            % Cláusula 5
cinza(elefante).       % Cláusula 6
escuro(Z) :-           % Cláusula 7
  preto(Z).
escuro(Z) :-           % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 3)
Nova meta: preto(X), grande(X).

Procure por uma cláusula que unifique com preto(X)
Cláusula 5 encontrada
preto(gato)
Esta cláusula não tem corpo, assim a lista de condições depois de instanciada torna-se:
grande(gato).
uma vez que já se provou preto(gato)
X instancia com gato

34

Significado Procedural

```
grande(urso).        % Cláusula 1
grande(elefante).    % Cláusula 2
pequeno(gato).       % Cláusula 3
marrom(urso).        % Cláusula 4
preto(gato)          % Cláusula 5
cinza(elefante).     % Cláusula 6
escuro(Z) :-         % Cláusula 7
  preto(Z).
escuro(Z) :-         % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 4)
Nova meta: grande(gato).

Procure por uma cláusula que unifique com grande(gato).
Nenhuma cláusula é encontrada.
Volte (backtrack) para o Passo 3, desfazendo a instanciação X=gato
Novamente a meta é preto(X), grande(X).

35

Significado Procedural

```
grande(urso).        % Cláusula 1
grande(elefante).    % Cláusula 2
pequeno(gato).       % Cláusula 3
marrom(urso).        % Cláusula 4
preto(gato)          % Cláusula 5
cinza(elefante).     % Cláusula 6
escuro(Z) :-         % Cláusula 7
  preto(Z).
escuro(Z) :-         % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 4)
meta: preto(X), grande(X).

Continue procurando após a Cláusula 5. Nenhuma cláusula é encontrada. Volte (backtrack) ao Passo 2 e continue procurando após a cláusula 7.
Cláusula 8 encontrada:
escuro(Z) :- marrom(Z).
Troque a primeira condição pelo corpo instanciado da cláusula 8, obtendo a nova lista de condições:
marrom(X), grande(X).

36

Significado Procedural

```
grande(urso).          % Cláusula 1
grande(elefante).      % Cláusula 2
pequeno(gato).         % Cláusula 3
marrom(urso).          % Cláusula 4
preto(gato)            % Cláusula 5
cinza(elefante).       % Cláusula 6
escuro(Z) :-           % Cláusula 7
  preto(Z).
escuro(Z) :-           % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 5)
meta: marrom(X), grande(X).

Procure por uma cláusula que unifique com marrom(X)
Cláusula 4 encontrada
marrom(urso)
Esta cláusula não tem corpo, assim a lista de condições depois de instanciada torna-se:
grande(urso).
uma vez que já se provou marrom(urso)
X instancia com urso

Significado Procedural

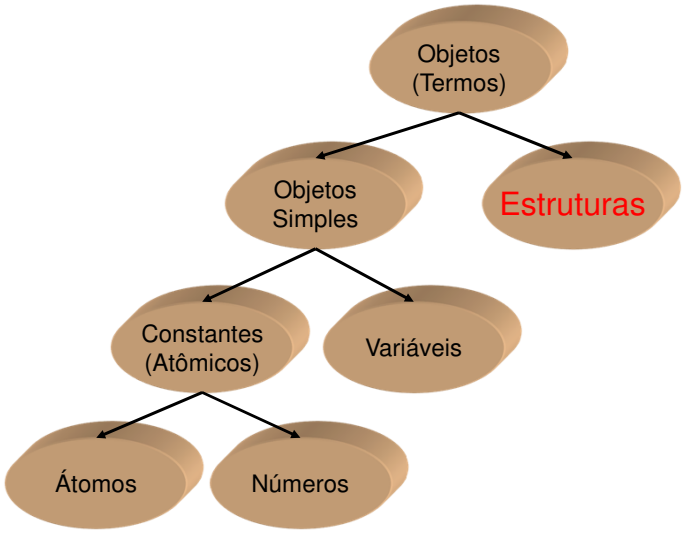
```
grande(urso).          % Cláusula 1
grande(elefante).      % Cláusula 2
pequeno(gato).         % Cláusula 3
marrom(urso).          % Cláusula 4
preto(gato)            % Cláusula 5
cinza(elefante).       % Cláusula 6
escuro(Z) :-           % Cláusula 7
  preto(Z).
escuro(Z) :-           % Cláusula 8
  marrom(Z).

?- escuro(X), grande(X).
```

(Passo 6)
meta: grande(urso).

Procure por uma cláusula que unifique com grande(urso)
Cláusula 1 encontrada
grande(urso)
Esta cláusula não tem corpo, assim a lista de condições torna-se vazia.
Isto indica um término com sucesso e a instanciação correspondente é
X = urso

38

Objetos em Prolog



39

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 objetos simples

40

Estruturas

- De forma a combinar componentes em um objeto simples, deve-se escolher um **functor**
- Um functor para o exemplo da data seria **data**
- Então a data 4 de maio de 2003 pode ser escrita como:

data(4,maio,2003)

41

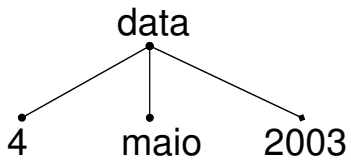
Estruturas

- Qualquer dia em maio pode ser representado pela estrutura:
  - data(Dia,maio,2003)
- Note que **Dia** é uma variável que pode ser instanciada com qualquer objeto em qualquer momento durante execução
- Sintaticamente, todos os objetos de dados em Prolog são termos
- Por exemplo, são termos:
  - maio
  - data(4,maio,2003)

42

Estruturas

- Todo objeto estruturado pode ser representado como uma árvore, onde:
- A raiz da árvore é o functor e os filhos da raiz são os componentes
- Para a estrutura data(4,maio,2003):



43

Estruturas

casou(joao, maria, dia(5, maio, 2000)).

casou(jose, claudia, dia(11, novembro, 1950)).

?-casou(X,Y,dia(11,novembro, 1950)).

?-casou(X,_,dia(_,maio,_)).

?-casou(_,_,dia(_,_,1950)).

44

functor/3

- O predicado (ou procedimento pré-definido de Prolog) functor/3 retorna o funtor e a aridade de uma estrutura

  ❑?- functor(Estrutura,Funtor,Aridade).

45

Exemplo functor/3

❑?- functor(amigos(luisa,felipe),F,A).
F = amigos
A = 2
yes

❑?- functor(gosta(glaucia,jogar(tenis,baralho)),F,A).
F = gosta
A = 2
yes

46

Exemplo functor/3

- ?- functor(maria,F,A).
  F = maria
  A = 0
  yes

47

Predicados para Verificação dos Tipos de Termos

Predicado	É verdadeiro se:
var(X)	X é uma variável não instanciada
nonvar(X)	X não é uma variável ou X é uma variável instanciada
atom(X)	X é um átomo
integer(X)	X é um inteiro
float(X)	X é um número real
atomic(X)	X é uma constante (átomo ou número)
compound(X)	X é uma estrutura

48

Predicados para Verificação de Tipos de Termos

```
?- var(Z), Z = 2.
Z = 2
?- Z = 2, var(Z).
no
?- integer(Z), Z = 2.
no
?- Z = 2, integer(Z), nonvar(Z).
Z = 2
?- atom(3.14).
no
?- atomic(3.14).
yes
?- atom(==>).
yes
?- atom(p(1)).
no
?- compound(p(1)).
yes
```

49

Verificação de Tipo: atom/1

```
?- atom(a).
yes

?- atom(7).
no

?- atom(X).
no
```

50

Verificação de Tipo : atom/1

```
?- X=a, atom(X).
X = a
yes

?- atom(X), X=a.
no
```

51

Verificação de Tipo : atomic/1

```
?- atomic(maria).
yes

?- atomic(5).
yes

?- atomic(gosta(vicente,maria)).
no
```

52

Verificação de Tipo : var/1

```
?- var(maria).
no

?- var(X).
yes

?- X=5, var(X).
no
```

53

Verificação de Tipo : nonvar/1

```
?- nonvar(X).
no

?- nonvar(maria).
yes

?- nonvar(23).
yes
```

54

Unificação de Termos

- Dois termos unificam (*matching*) se:
  - Eles são idênticos ou
  - As variáveis em ambos os termos podem ser instanciadas a objetos de maneira que após a substituição das variáveis por esses objetos os termos se tornam idênticos
- Por exemplo, há unificação entre os termos
  - data(D,M,2003) e data(D1,maio,A)
    - instanciando D = D1, M = maio, A = 2003

55

Unificação

- Prolog unifica

mulher(X)

com

mulher(maria)

e assim instancia a variável **X** com o átomo **maria**.

56

Unificação

maria e **maria** unifica
mulher(maria) e **mulher(maria)** unifica
vicente e **maria** não unifica
mulher(maria) e **mulher(glaucia)** não unifica

Unificam?
maria e **X**
mulher(Z) e **mulher(maria)**
gosta(maria,X) e **gosta(X,vicente)**

57

Unificação de Termos

```
?- data(D,M,2003) = data(D1,maio,A),
   data(D,M,2003) = data(15,maio,A1).
D = 15
M = maio
D1 = 15
A = 2003
A1 = 2003

?- triângulo = triângulo, ponto(1,1) = X,
   A = ponto(4,Y), ponto(2,3) = ponto(2,Z).
X = ponto(1,1)
A = ponto(4,_G652)
Y = _G652
Z = 3
```

58

Unificação de Termos

- Por outro lado, não há unificação entre os termos
  - data(D,M,2003) e data(D1,M1,1948)
  - data(X,Y,Z) e ponto(X,Y,Z)
- A unificação é um processo que toma dois termos e verifica se eles unificam
  - Se os termos não unificam, o processo falha (e as variáveis não se tornam instanciadas)
  - Se os termos unificam, o processo tem sucesso e também instancia as variáveis em ambos os termos para os valores que os tornam idênticos

59

Unificação de Termos

- As regras que regem se dois termos **S** e **T** unificam são:
  - se **S** e **T** são constantes, então **S** e **T** unificam somente se são o mesmo objeto
  - se **S** for uma variável e **T** for qualquer termo, então unificam e **S** é instanciado para **T**
  - se **S** e **T** são estruturas, elas unificam somente se
    - **S** e **T** têm o mesmo functor principal e
    - todos seus componentes correspondentes unificam

60

Unificação de Termos

- Unificação em Prolog é diferente da unificação em Lógica. A unificação Lógica requer a verificação de ocorrência de uma variável em um termo (*occurs check*), a qual, por razões de eficiência, não é implementado em Prolog

- Exemplo:
  - **?-** X = f(X).
  - X = f(f(f(f(f(f(f(f(f(...)))))))))
  - X= f(**)

61

Unificação de Termos

?- k(s(g),Y) = k(X,t(k)).

62

Unificação de Termos

?- k(s(g),Y) = k(X,t(k)).
X=s(g)
Y=t(k)
yes
?-

63

Unificação de Termos

?- k(s(g),t(k)) = k(X,t(Y)).

64

Unificação de Termos

?- k(s(g),t(k)) = k(X,t(Y)).
X=s(g)
Y=k
yes
?-

65

Unificação de Termos

?- gosta(X,X) = gosta(marcelo,maria).

66

Unificação – mais exemplos

```
vertical( linha(ponto(X,Y), ponto(X,Z))).

horizontal( linha(ponto(X,Y), ponto(Z,Y))).
```

67

Unificação – mais exemplos

```
vertical( linha(ponto(X,Y), ponto(X,Z))).

horizontal( linha(ponto(X,Y), ponto(Z,Y))).
```

```
?- vertical(linha(ponto(1,1),ponto(1,3))).
yes
?-
```

68

Unificação – mais exemplos

```
vertical( linha(ponto(X,Y), ponto(X,Z))).

horizontal( linha(ponto(X,Y), ponto(Z,Y))).
```

```
?- vertical(linha(ponto(1,1),ponto(1,3))).
yes
?- vertical(linha(ponto(1,1),ponto(3,2))).
no
?-
```

69

Unificação – mais exemplos

```
vertical( linha(ponto(X,Y), ponto(X,Z))).

horizontal( linha(ponto(X,Y), ponto(Z,Y))).
```

```
?- horizontal(linha(ponto(1,1),ponto(1,Y))).
Y = 1;
no
?-
```

70

Unificação – mais exemplos

```
vertical( linha(ponto(X,Y), ponto(X,Z))).

horizontal( linha(ponto(X,Y), ponto(Z,Y))).
```

```
?- horizontal(linha(ponto(2,3),Ponto)).
Ponto = ponto(_554,3);
no
?-
```

71

Unificação

- Exercício

- Suponha a seguinte relação:
  - casou(joao, maria, dia(5, maio, 1980)).
  - casou(andre, fernanda, dia(11, agosto, 1950)).
  - casou(adriano, claudia, dia(15, outubro, 1973)).

- Questione:
  - Qual a data do casamento de Maria?
  - Qual o mês do casamento de Andre?
  - Quem casou antes de Adriano, considerando somente o ano de casamento?
  - Quem casou a menos de 30 anos (considerando apenas o ano)?

72

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

73

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

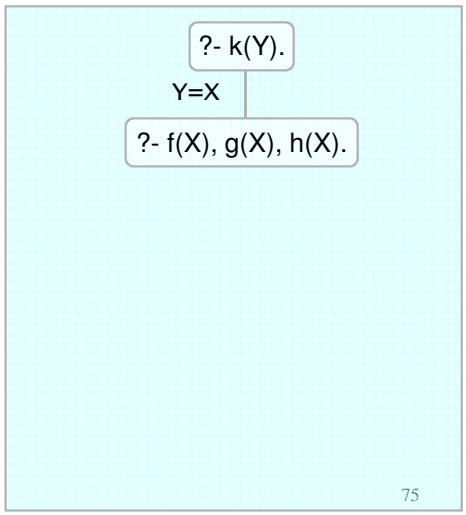
?- k(Y).

74

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

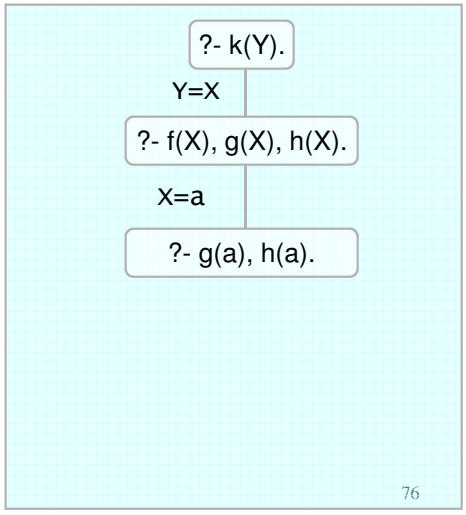


75

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

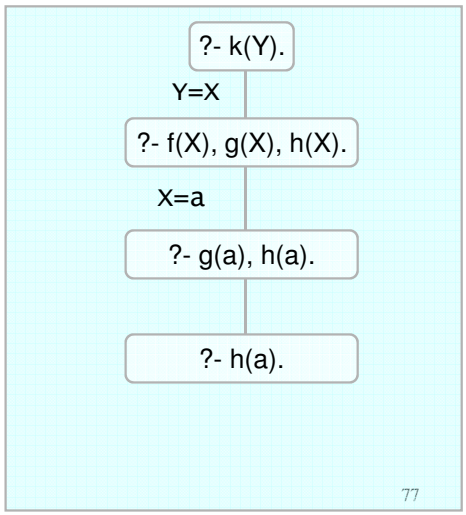


76

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

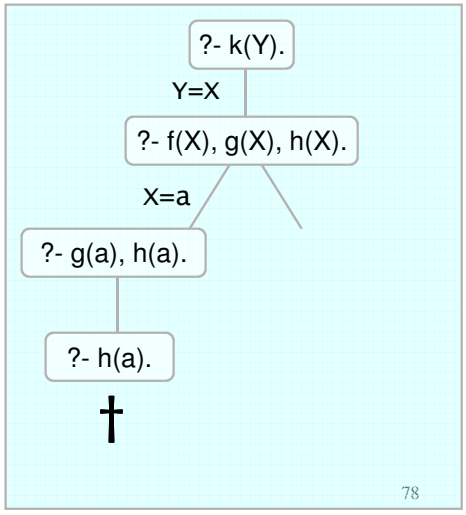


77

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

?- k(Y).

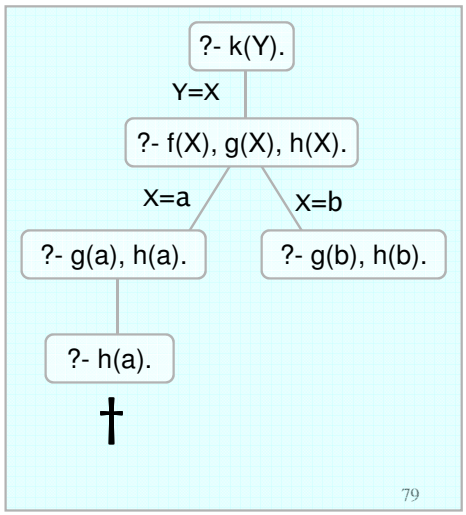


78

Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

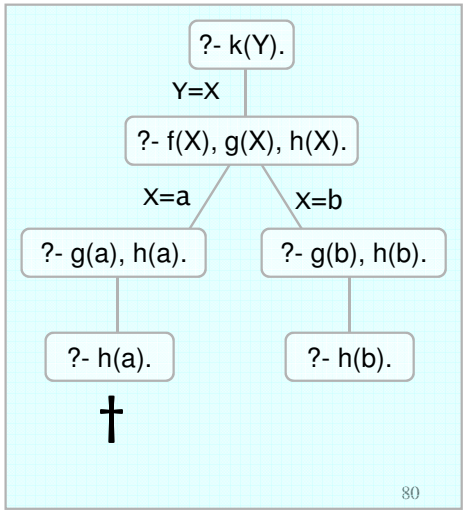
?- k(Y).



Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

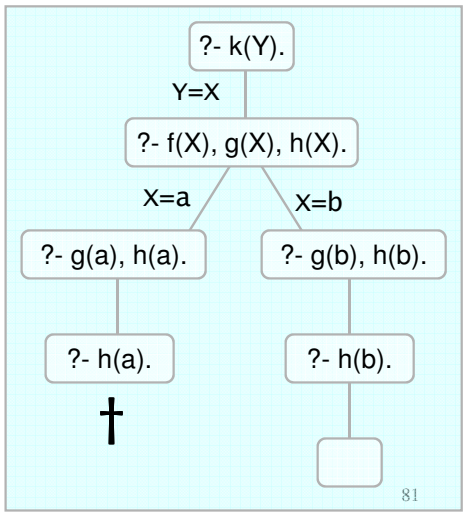
?- k(Y).



Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

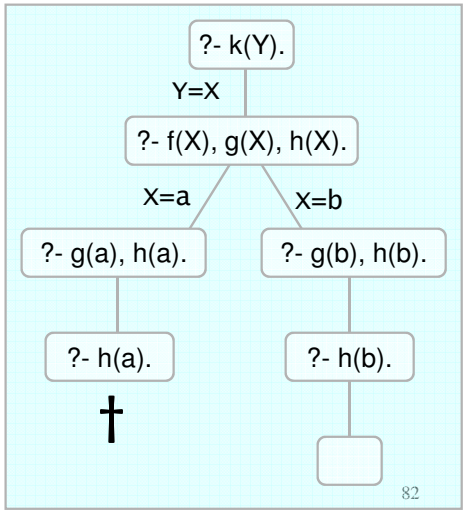
```
?- k(Y).
Y=b
```



Exemplo: árvore de busca

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
Y=b;
no
?-
```



Exemplo2: árvore de busca

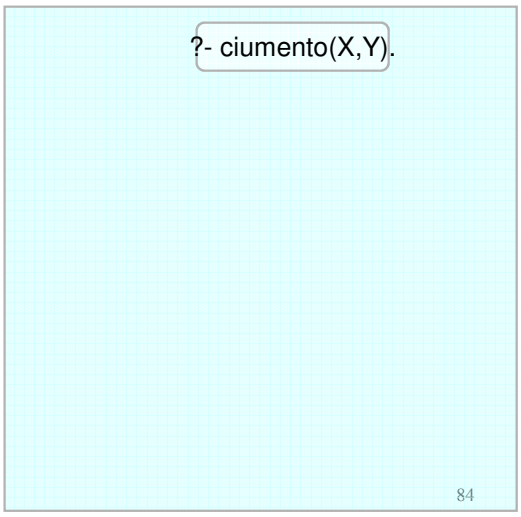
```
gosta(vicente,maria).
gosta(marcelo,maria).

sente_ciume(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
?- sente_ciume(X,Y).
```


Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

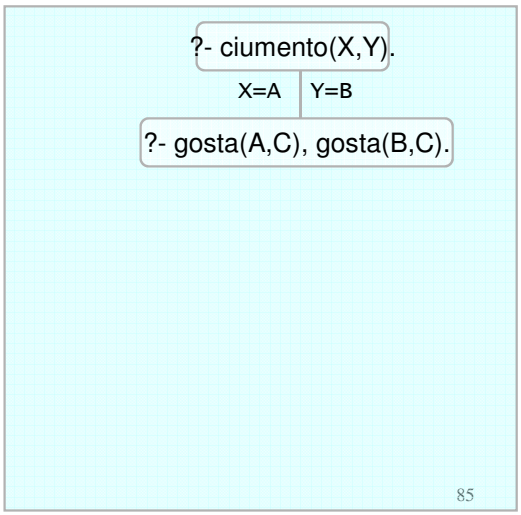
```
?- ciumento(X,Y).
```



Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
?- ciumento(X,Y).
```

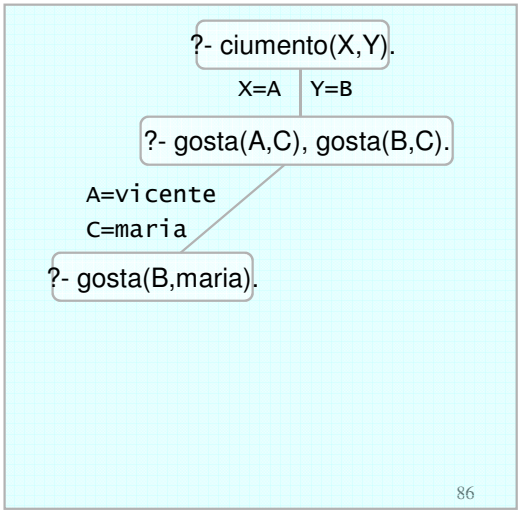


85

Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
?- ciumento(X,Y).
```

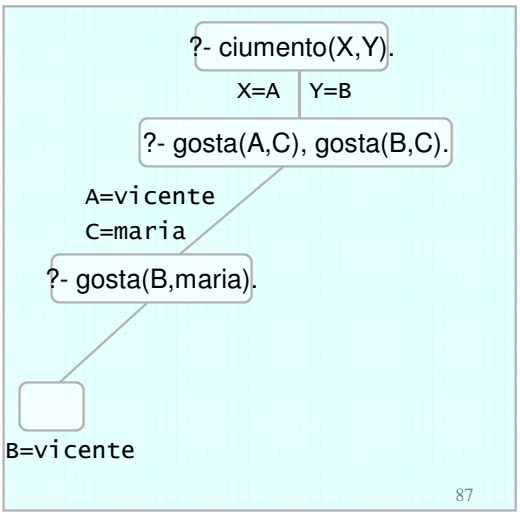


86

Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
?- ciumento(X,Y).
X=vicente
Y=vicente
```

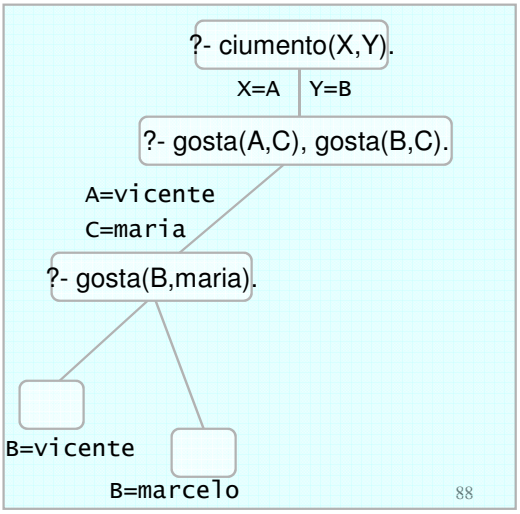


87

Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
?- ciumento(X,Y).
X=vicente
Y=vicente;
X=vicente
Y=marcelo
```

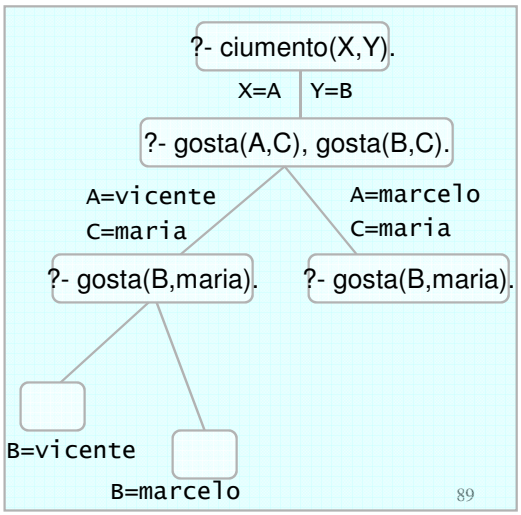


88

Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

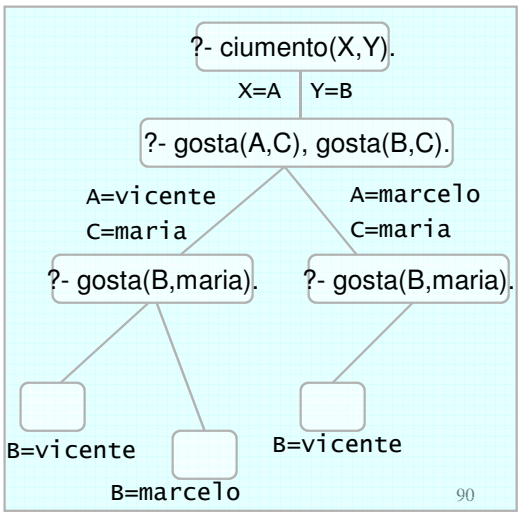
```
?- ciumento(X,Y).
X=vicente
Y=vicente;
X=vicente
Y=marcelo;
```



Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

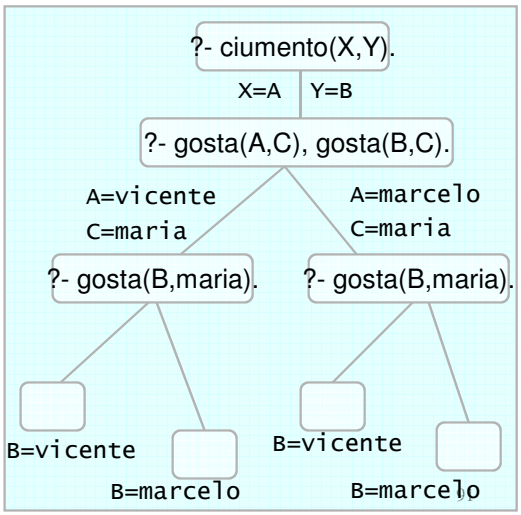
```
....
X=vicente
Y=marcelo;
X=marcelo
Y=vicente
```



Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

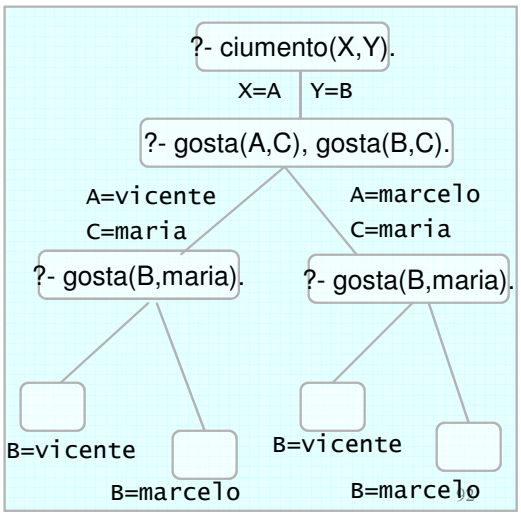
```
....
X=marcelo
Y=vicente;
X=marcelo
Y=marcelo
```



Exemplo2: árvore de busca

```
gosta(vicente,maria).
gosta(marcelo,maria).

ciumento(A,B):-
    gosta(A,C),
    gosta(B,C).
```

```
....
X=marcelo
Y=vicente;
X=marcelo
Y=marcelo;
no
```



Recursão

- Exemplo de várias versões de

  descendente/2

93

descendente/2 (versão 1)

```
filha(brigite,carolina).
filha(carolina,debora).

descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), filha(Z,Y).
```

94

descendente/2 (versão 1)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).


descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), filha(Z,Y).
```

95

descendente/2 (versão 1)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).


descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), filha(Z,Y).
```

```
?- descendente(anna,debora).
no
?-
```

96

descendente/2 (versão 2)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).


descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), filha(Z,Y).
descendente(X,Y):- filha(X,Z), filha(Z,U), filha(U,Y).
```

```
?-
```

97

descendente/2 (versão 3)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).


descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), descendente(Z,Y).
```

```
?-
```

98

descendente/2 (versão 3)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).


descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), descendente(Z,Y).
```

```
?- descendente(anna,debora).
yes
```

99

descendente/2 (versão 3)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).

descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), descendente(Z,Y).
```

```
?- descendente(A,B).
A=anna
B=brigite
```

100

descendente/2 (versão 4)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).

descendente(X,Y):- descendente(Z,Y), filha(X,Z).
descendente(X,Y):- filha(X,Y).
```

```
?- descendente(A,B).
ERROR: OUT OF LOCAL STACK
```

101

descendente/2 (versão 5)

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).

descendente(X,Y):- filha(X,Z), descendente(Z,Y).
descendente(X,Y):- filha(X,Y).
```

```
?- descendente(A,B).
A=anna
B=emily
```

102

descendente/2

```
filha(anna,brigite).
filha(brigite,carolina).
filha(carolina,debora).
filha(debora,emily).

descendente(X,Y):- filha(X,Y).
descendente(X,Y):- filha(X,Z), descendente(Z,Y) .
```

```
?- descendente(A,B).
```

103

Exercício

Considere as seguintes premissas:

- Todos os animais têm pele. Peixe é um tipo de animal, pássaros são outro tipo e mamíferos são um terceiro tipo. Normalmente, os peixes têm nadadeiras e podem nadar, enquanto os pássaros têm asas e podem voar. Se por um lado os pássaros e os peixes põem ovos, os mamíferos não põem. Embora tubarões sejam peixes, eles não põem ovos, seus filhotes nascem já formados. Salmão é um outro tipo de peixe, e é considerado uma delícia. O canário é um pássaro amarelo. Uma avestruz é um tipo de pássaro grande que não voa, apenas anda. Os mamíferos normalmente andam para se mover, como por exemplo uma vaca. As vacas dão leite, mas também servem elas mesmas de comida (carne). Contudo, nem todos os mamíferos andam para se mover. Por exemplo, o morcego voa.

Considere ainda que existem os seguinte animais:

- Piupiu, que é um canário; Nemo, que é um peixe; Tutu, que é um tubarão; Mimosa, que é uma vaca; Vamp, que é um morcego; Xica, que é uma avestruz; Alfred, que é um salmão.

Defina fatos e regras prolog que representam as premissas acima e formule as consultas:

1. O piupiu voa?
2. Qual a cor do piupiu?
3. A Xica voa?
4. A Xica tem asas?
5. O Vamp voa? Tem asas? Poem ovos?
6. Quais os nomes dos animais que põem ovos?
7. Quais os nomes dos animais que são comestíveis?
8. Quais os nomes dos animais que se movem andando?
9. Quais os nomes dos animais que se movem nadando mas não põem ovos?

104

Desafio

- O que acontece abaixo?
  - Não há base consultada.