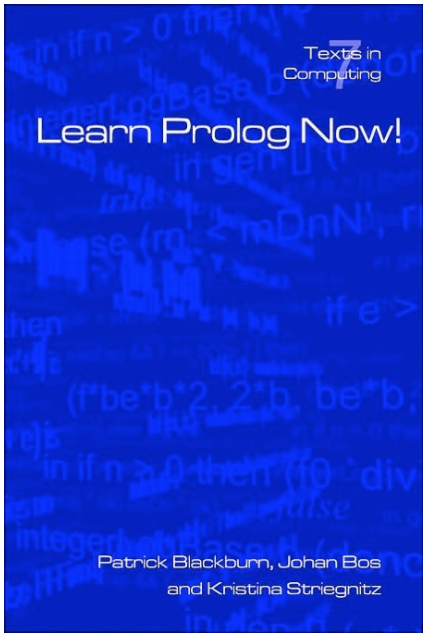
# Paradigma Lógico

Baseado nos slides do livro

Learn Prolog Now!



# Paradigma Lógico

- Teoria
  - Introdução ao Prolog
  - Fatos, Regras e consultas
  - Prolog Sintaxe

- Exercícios
  - Exercícios de LPN capítulo 1
  - Trabalho prático

# SWI Prolog

- Interpretador Prolog disponível gratuitamente.
- Funciona com:
  - Linux,
  - Windows ou
  - Mac OS
- Há muitos mais interpretadores Prolog


- Nem todos tem conformidade com ISO ou livre

# Objetivo desta apresentação(1/2)

- Dê alguns exemplos simples de programas Prolog
- Discutir as três construções básicas em Prolog:
  - Fatos
  - Regras
  - Consultas

# Objetivo desta apresentação(2/2)

- Introduzir outros conceitos, tais como
  - o papel da lógica
  - unificação com a ajuda de variáveis
- Comece o estudo sistemático do Prolog, definindo:
  - Condições
  - átomos, e
  - variáveis

# Prolog

- " Programação com Logica"
- Muito diferente de outras linguagens de programação

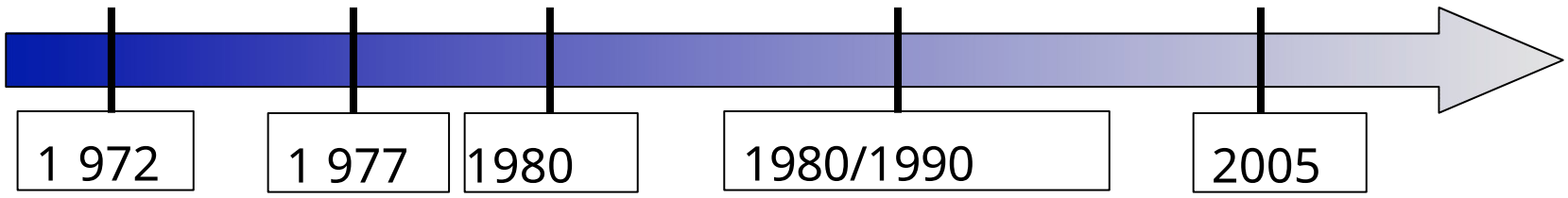
  - Declarativa(não processual)
  - Recursão(não “para” ou “enquanto” laços)
  - Relações(sem funções)
  - Unificação

# História da Prolog

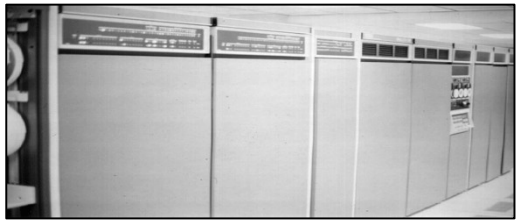
primeiro interpretador Prolog pela

**Alain Colmerauer** e

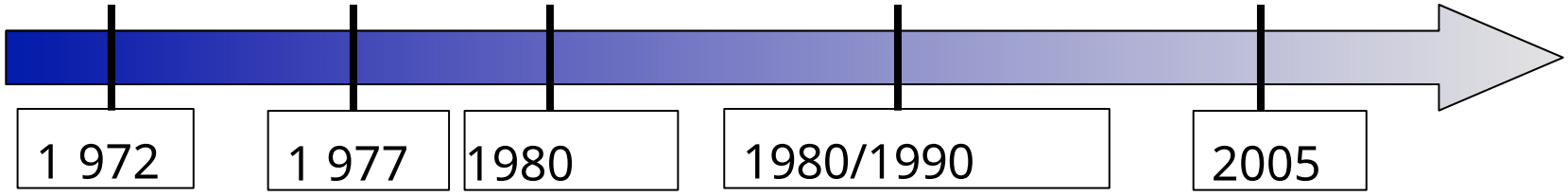
**Philippe Roussel**



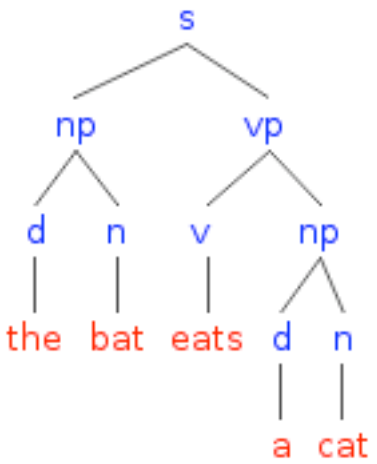
# História da Prolog



implementação de compilador DEC10
por **David HD Warren**



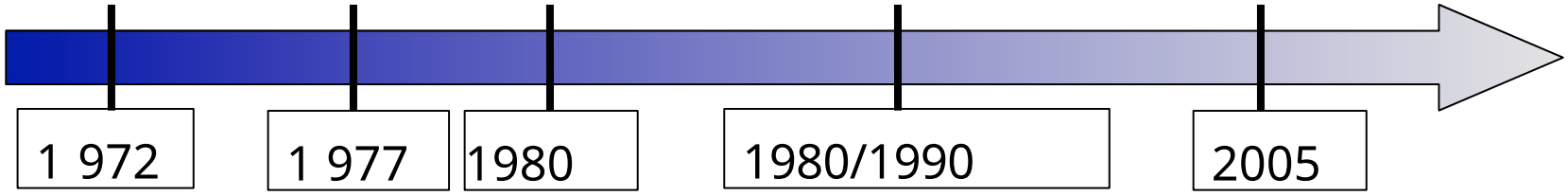
# História da Prolog




Gramática de cláusulas definidas

implementação por

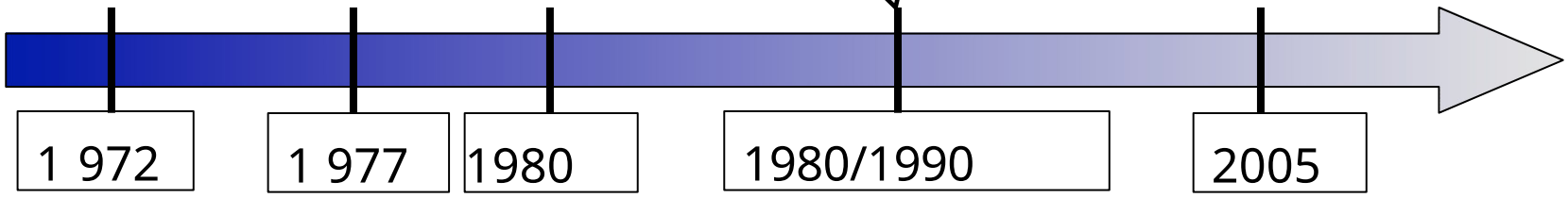
**Pereira** e **Warren**



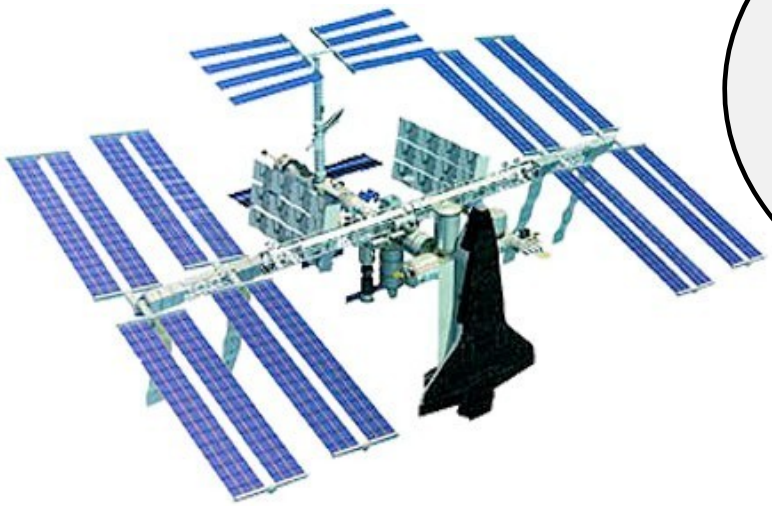
# História da Prolog

Prolog cresce em popularidade,

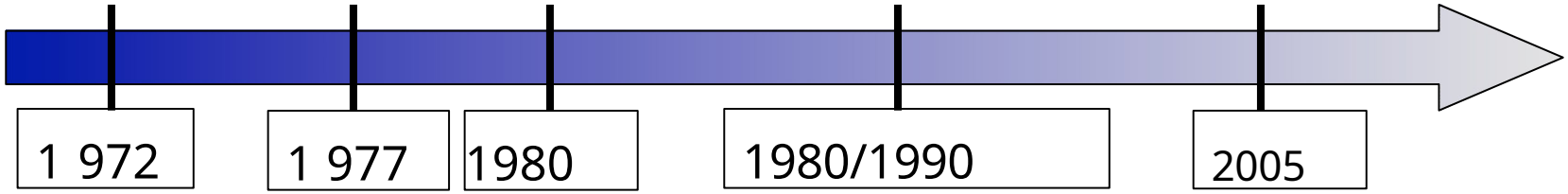
especialmente no Japão e na Europa



# História da Prolog

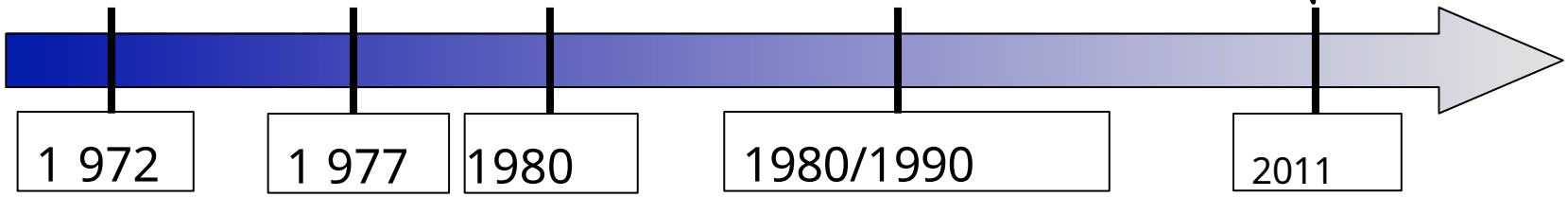


Prolog usado para programar interface de linguagem natural na Estação Espacial Internacional da NASA



# História da Prolog


Peças de **Watson** QA, supercomputador da IBM, foram codificadas em Prolog



# Prolog e Aplicações Web

- programas Prolog são muitas vezes menores
- pequenez incentiva código bem escrito
- portanto, mais fácil de manter


**Fonte:**

http://www.pathwayslms.com/swipltuts/

# Idéia básica do Prolog

- Descrever a situação de interesse

  Declara/cria uma base de Conhecimento

- Faça uma pergunta
- Prolog:
  - logicamente deduz novos fatos sobre a situação que nós descrevemos
  - retorna essas deduções como respostas

# Consequências

- Pense declarativa, não de maneira imperativa
  - Desafiador
  - Exige uma mentalidade diferente
- Linguagem de alto nível
  - Não tão eficiente quanto, digamos, C
  - Bom para prototipagem rápida
  - Útil em muitas aplicações de IA (representação do conhecimento, inferência)

# Base de Conhecimento 1

```
mulher(mia).
mulher(jody).
mulher(yolanda).
tocaGuitarra(jody).
festa.
```

# Base de Conhecimento 1

```
mulher(mia).
mulher(jody). mulher(yolanda).
tocaGuitarra(jody).
festa.
```

```
? - mulher(mia).
 sim

? - tocaGuitarra(jody).
 sim

? - tocaGuitarra(mia).
não
```

# Base de Conhecimento 1

```
mulher(mia).
mulher(jody).
mulher(yolanda).
tocaGuitarra(jody).
festa.
```

```
? - tatuada(jody).
ERROR: Undefined procedure: tatuada/1 (DWIM could not correct goal)
? -
```

# Base de Conhecimento 1

```
mulher(mia).
mulher(jody).
mulher(yolanda).
tocaGuitarra(jody).
festa.
```

```
?- festa.  sim
?-
```

# Base de Conhecimento dois

feliz(yolanda). fato

escutaMusica(mia). fato

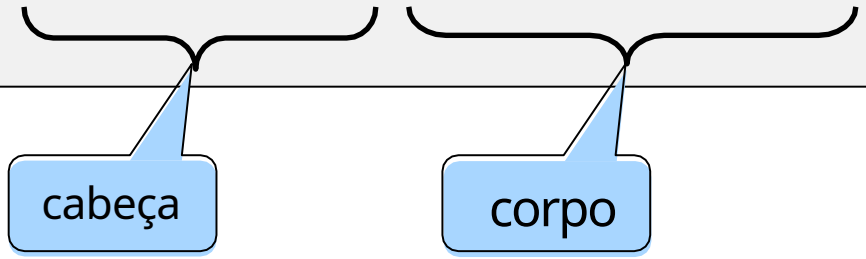
escutaMusica(yolanda):- feliz(yolanda). regra

tocaGuitarra(mia):- escutaMusica(mia). regra

tocaGuitarra(yolanda):- escutaMusica(yolanda). regra

# Base de Conhecimento dois

```
feliz(yolanda).  escutaMusica(mia).

escutaMusica(yolanda):- feliz(yolanda).

tocaGuitarra(mia):-  escutaMusica(mia).

   tocaGuitarra(yolanda):- escutaMusica(yolanda).
```



# Base de Conhecimento dois

```
feliz(yolanda).
escutaMusica(mia).
escutaMusica(yolanda):- feliz(yolanda).
tocaGuitarra(mia):- escutaMusica(mia).
tocaGuitarra(yolanda):- escutaMusica(yolanda).
```

```
? - tocaGuitarra(mia).
 sim
? - tocaGuitarra(yolanda).
 sim
```

# cláusulas

```
feliz(yolanda).
escutaMusica(mia).
escutaMusica(yolanda):- feliz(yolanda).
tocaGuitarra(mia):- escutaMusica(mia).
tocaGuitarra(yolanda):- escutaMusica(yolanda).
```

Há cinco cláusulas nesta base de conhecimento: dois fatos, e três regras.

O fim de uma cláusula é marcado com um ponto final.

# predicados

```
feliz(yolanda).
escutaMusica(mia).
escutaMusica(yolanda):- feliz(yolanda).
tocaGuitarra(mia):- escutaMusica(mia).
tocaGuitarra(yolanda):- escutaMusica(yolanda).
```

Há três ***predicados*** nesta base de conhecimento:

feliz, escutaMusica e tocaGuitarra

# Base de Conhecimento 3

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch).
tocaGuitarra(butch):- escutaMusica(butch).
```

# expressando Conjunção

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch).
tocaGuitarra(butch):- escutaMusica(butch).
```

A vírgula “," expressa conjunção em Prolog(ou seja AND)

# Base de Conhecimento 3

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch).
tocaGuitarra(butch):- escutaMusica(butch).
```

```
?  - tocaGuitarra(vincent).
não
?-
```

# Base de Conhecimento 3

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch).
tocaGuitarra(butch):- escutaMusica(butch).
```

```
?  - tocaGuitarra(butch).
 sim
?-
```

# expressando disjunção (ou)

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch).
tocaGuitarra(butch):- escutaMusica(butch).
```

```
feliz(vincent).  escutaMusica(butch).
tocaGuitarra(vincent):- escutaMusica(vincent), feliz(vincent).
tocaGuitarra(butch):- feliz(butch); escutaMusica(butch).
```

# Prolog e Lógica

- Claramente, Prolog tem algo a ver com a lógica

| | Prolog | Logica |
|---|---|---|
| Implicação | A:- B | B implica em A |
| Conjunção | A,B | A ∧ B (E ou AND) |
| Disjunção | A;B | A ∨ B (OU ou OR) |

- Uso de inferência (modus ponens)
- Negação: /+

# Base de Conhecimento 4

```
mulher(mia).
mulher(jody).
mulher(yolanda).

ama(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).
```

# instanciação variável

```
mulher(mia).
mulher(jody).
mulher(yolanda).

ama(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).
```

```
? - mulher(X).

X = mia
```

# Pedindo Alternativas com ;

```
mulher(mia).
mulher(jody).
mulher(yolanda).

ama(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).
```

```
? - mulher(X).
X = mia;
X = jody;
X = yolanda;
 não
```

# Base de Conhecimento 4

```
mulher(mia).
mulher(jody).
mulher(yolanda).

ama(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).
```

```
? - amores(marsellus, X), mulher(X).
X = mia
Sim
? -
```

# Base de Conhecimento 4

```
mulher(mia).
mulher(jody).
mulher(yolanda).

amores(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).
```

```
? - amores(pumpkin, X), mulher(X).
não
?-
```

# Base de Conhecimento 5

```
amores(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).

ciumento(X, Y):- amores(X, Z), amores(Y, Z).
```

# Base de Conhecimento 5

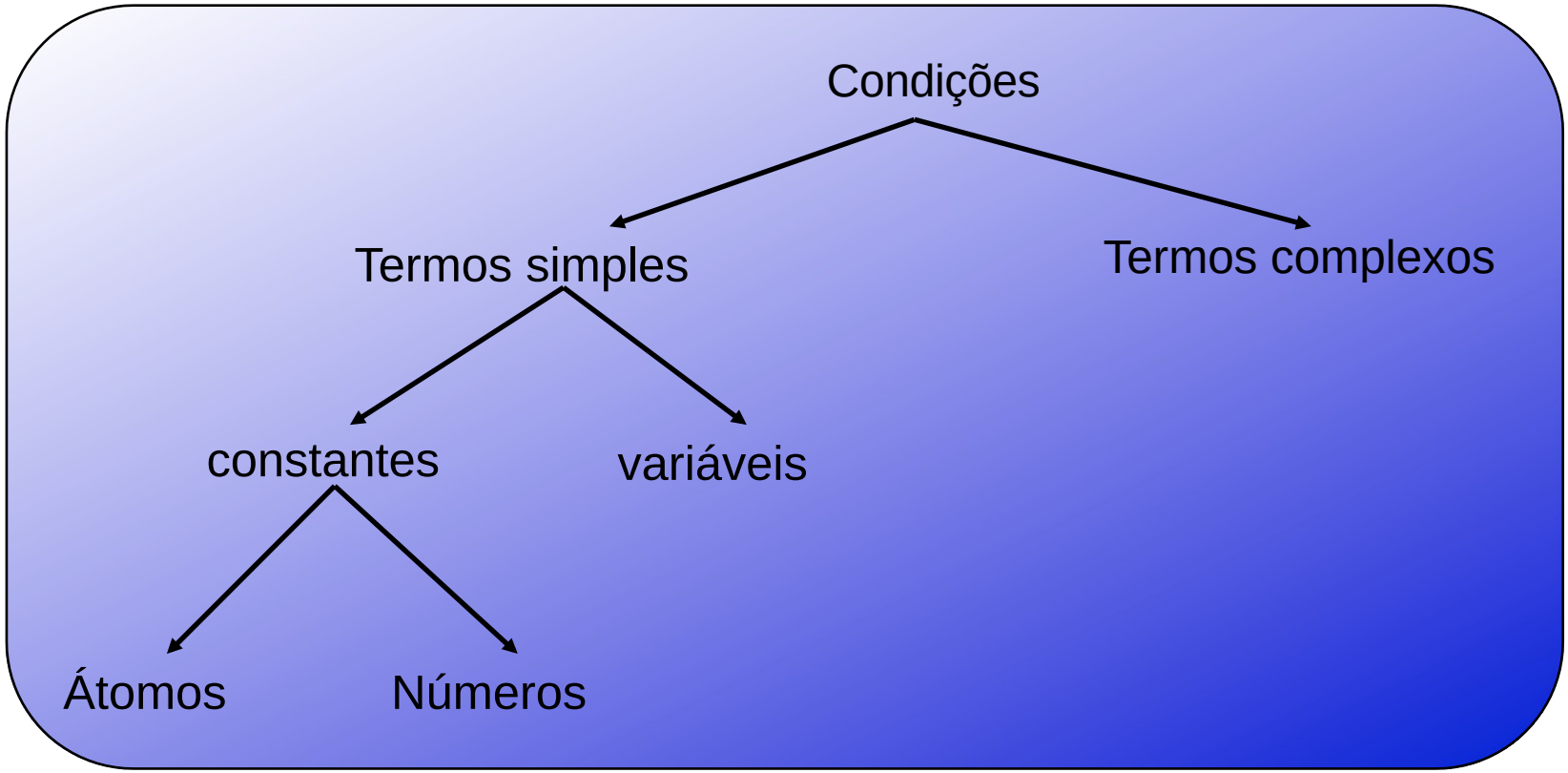
```
amores(vincent, mia).
amores(marsellus, mia).
amores(pumpkin, honey_bunny).
amores(honey_bunny, pumpkin).

ciumento(X, Y):- amores(X, Z), amores(Y, Z).
```

```
? - ciumento(marsellus, W).
 W =  vincent
? -
```

# Sintaxe de Prolog

- Q: O que exatamente são fatos, regras e consultas construído a partir de?
- A: termos Prolog

# termos Prolog



# átomos

- Uma sequência de caracteres de letras maiúsculas, letras minúsculas, dígitos ou sublinhado, começando com uma letra minúscula
  - Exemplos: **butch, big_kahuna_burger, tocarGuitarra**

# átomos

- Uma sequência de caracteres de letras maiúsculas, letras minúsculas, dígitos ou sublinhado, começando com uma letra minúscula
  - Exemplos: **butch, big_kahuna_burger, tocaGuitarra**
- Uma sequência arbitrária de caracteres entre aspas
  - Exemplos: **'vincent', 'Cinco agitação dólar', '@ $%'**

# átomos

- Uma sequência de caracteres de letras maiúsculas, letras minúsculas, dígitos ou sublinhado, começando com uma letra minúscula
  - Exemplos: **butch, big_kahuna_burger, tocaGuitarra**

- Uma sequência arbitrária de caracteres entre aspas
  - Exemplos: **'vincent', 'Cinco agitação dólar', '@ $%'**

- Uma sequência de caracteres especiais
  - Exemplos: **:,** **;.:-**

# Números

- integer:

  12, -34, 22342

- float:

  34573.3234,
  0.3435

# variáveis

- Uma sequência de caracteres de letras maiúsculas, letras minúsculas, dígitos, ou sublinhado, começando com qualquer uma letra maiúscula ou um sublinhado

- Exemplos:

  **X, Y, Variavel, Vincent, _tag**

# Termos complexos

- Átomos, números e variáveis são blocos de construção para **termos complexos**
- Termos complexos são construídas a partir de um **funtor** diretamente seguida por uma sequência de **argumento**
  - Argumentos são colocados entre parênteses, separados por vírgulas
  - O functor deve ser um átomo

# Exemplos de termos complexos

- Exemplos temos visto antes:
  - tocaGuitarra(jody)
  - ama(vincent, mia)
  - ciumento(marsellus, W)
-

# Aridade

- O número de argumentos um termo complexo tem é chamada de aridade _____
- Exemplos:

**mulher(mia)** é um termo com 1 aridade

**ama(vincent, mia)** tem aridade 2

**pai(pai(butch))** aridade 1

# Aridade é importante

- Você pode definir dois predicados com a mesma functor mas com aridade diferente
- Prolog iria tratar isso como dois predicados diferentes!

- Na documentação Prolog, aridade de um predicado é geralmente indicada com o sufixo "/" seguido de um número para indicar a aridade
- Lembra um pouco sobrecarga da Orientacao a Objetos

# Exemplo de aridade

```
feliz(yolanda).  escutaMusica(mia).
escutaMusica(yolanda):- feliz(yolanda).
tocaGuitarra(mia):- escutaMusica(mia).
tocaGuitarra(yolanda):- escutaMusica(yolanda).
```

- Isto define base de conhecimento
  - feliz / 1
  - escutaMusica / 1
  - tocaGuitarra / 1

# Objetivo deste capítulo

- Discutir **unificação** em Prolog

- Explicar a **Estratégia de pesquisa do** Prolog
  - Prolog deduz novas informações a partir de fatos e regras, utilizando modus ponens.
    - A ser abordado em Inteligência Artificial

# Unificação

- Considere a seguinte base de conhecimento

```
mulher(mia).

mulher(jody).

mulher(yolanda).

amores(vincent, mia).

amores(marsellus, mia).

amores(pumpkin, honey_bunny).

amores(honey_bunny, pumpkin).

ciumento(X, Y):- amores(X, Z), amores(Y, Z).
```

# Unificação

- Exemplo de unificação

**mulher(X)**

com

**mulher(mia)**

instanciará a variável **X** com o átomo **mia.**

# Unificação

- Definição - dois termos unificam, se:
  - se eles são o mesmo termo, ou
  - se eles contêm variáveis que podem ser uniformemente instanciadas com termos de tal maneira que os termos resultantes são iguais

# Unificação

Isso significa que:

- **mia** e **mia** unificam
- **42** e **42** unificam
- **mulher(mia)** e **mulher(mia)** unificam

# Unificação

- Isso significa que:
  - **mia** e **mia** unificam
  - **42** e **42** unificam
  - **mulher(mia)** e **mulher(mia)** unificam
- Isto também significa que:
  - **vincent** e **mia** não unificam
  - **mulher(mia)** e **mulher(jody)** não unificam

# Unificação

- E sobre os termos:
  - **mia** e **X**
  - **mulher(Z)** e **mulher(mia)**
  - **amores(mia, X)** e **amores(X, vincent)**

# Instanciações

- Quando Prolog unifica dois termos, ele tenta executar todas combinações de instanciação necessárias, de modo que os termos sejam iguais depois usando toda a base de conhecimento e as regras aplicáveis disponíveis
- Assemelha-se ao raciocínio lógico humano
- Busca meios para provar uma teoria usando seu conhecimento
- Isso faz com que a unificação de um mecanismo de programação muito poderoso

# Definição 1/3

1. Se T1 e T2 são constantes, então

   T1 e T2 unificam se eles são o mesmo átomo, ou o mesmo número

# Definição revista 2/3

1. Se T1 e T2 são constantes, então

   T1 e T2 unificam se eles são o mesmo átomo, ou o mesmo número

2. Se T1 é uma variável e T2 representa qualquer tipo de termo, então

   T1 e T2 unificam e T1 é instanciado para T2

   (e vice versa)

# Definição revista 3/3

1. Se T1 e T2 são constantes, então

   T1 e T2 unificam se eles são o mesmo átomo, ou o mesmo número

2. Se T1 é uma variável e T2 representa qualquer tipo de termo, então

   T1 e T2 unificam e T1 é instanciado para T2
   (e vice versa)

3. Se T1 e T2 são termos complexos, em seguida, eles unificam se:
   1. Eles têm o mesmo funtor e aridade, e
   2. todos os seus argumentos correspondentes unificam e
   3. as instâncias variáveis são compatíveis.

# Exemplo com termos complexos

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

? -

# Programação com Unificação

```
vertical(linha(ponto(X, _), ponto(X, _))).

horizontal(linha(ponto(_, Y), ponto(_, Y))).
```

# Programação com Unificação

```
vertical(linha(ponto(X, _), ponto(X, _))).


horizontal(linha(ponto(_, Y), ponto(_, Y))).
```

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

# Programação com Unificação

```
vertical(linha(ponto(X, _), ponto(X, _))).


horizontal(linha(ponto(_, Y), ponto(_, Y))).
```

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

? - vertical(linha(ponto(1,1), ponto(3,2))).
false.
?-
```

# Programação com Unificação

```
vertical(linha(ponto(X, _), ponto(X, _))).

horizontal(linha(ponto(_, Y), ponto(_, Y))).
```

```
? - horizontal(linha(ponto(1,1), ponto(1, Y))).
Y = 1;
false.
?-
```

# Programação com Unificação

```
vertical(linha(ponto(X, _), ponto(X, _))).


horizontal(linha(ponto(_, Y), ponto(_, Y))).
```

```
? - horizontal(linha(ponto(2,3), Ponto)).
Ponto =  ponto(_554,3);
false.
?-
```

# Exercícios com Unificação

# Exercício 2.1

Qual dos seguintes pares de termos unificam? Se necessário, dar as instâncias de variáveis que levam a unificação bem sucedida.

1. pão = pão
2. 'Pão' = pão
3. 'pão' = pão
4. Pão = pão
5. pão = salsicha
6. alimentos(pão) = pão
7. alimentos(pão) = X

# Exercício 2.1

8. alimentar(X) = comida(pão)

9. alimentos(pão, X) = comida(Y, salsicha)

10. alimentos(pão, X, cerveja) = comida(Y, salsicha, X)

11.alimentos(pão, X, cerveja) = comida(Y, kahuna_burger)

12.comida(X) = X

13.refeição(alimentos(pão), bebidas(cerveja)) = refeição(X, Y)

14.refeição(alimentos(pão), X) = refeição(X, bebida(cerveja))

# Exercício 2.2

Estamos trabalhando com a seguinte base de conhecimento:

```
house_elf(dobby).
bruxa(hermione).
bruxa('McGonagall').
bruxa(rita_skeeter).
assistente(harry).
magica(X):-house_elf(X).
magica(X):-bruxa(X).
magica(X):-assistente(X).
```

Qual das seguintes consultas estão satisfeitas?

1. -? magica(house_elf).
2. -? assistente(harry).
3. -? magica(assistente).
4. -? magica( 'McGonagall').
5. -? magica(Hermione).