# PARADIGMA LÓGICO

*PARADIGMAS DE PROGRAMAÇÃO*

## Programação em Lógica (1)

- *Características*
  - *Programas são relações entre E/S*
  - *As características da solução são especificadas, mas o processo completo de obtê-la não o é*
  - *Estilo declarativo, como no paradigma funcional*
  - *Foco no **o que** fazer (declarativo), e não no **como** fazer (imperativo)*

## Programação em Lógica (2)

- *Base*
  - *Axiomas lógicos (proposições) que representam o conhecimento e regras de dedução de novos conhecimentos.*
    - *Ex.: Uma baleia é um mamífero. Um mamífero tem sangue quente. Então, uma baleia tem sangue quente.*
- *Programa*
  - *Conjunto de fatos (conhecidos) e regras (de dedução) que formam a base para formular consultas.*

## Principais aplicações (1)

- *Sistemas Baseados em Conhecimento*
  - *aplicam mecanismos automatizados de raciocínio para a representação e inferência de conhecimento.*
- *Bancos de Dados "Inteligentes"*
  - *empregam "agentes" de busca de dados com base em critérios.*
  - *SGBDs : consultas declaradas em cálculo relacional, que é forma de lógica simbólica*

## Principais aplicações (2)

- *Sistemas Especialistas*
  - *emulam a especialização humana em algum domínio particular.*
  - *Consiste em banco de dados de fatos, processo de inferência, de alguma heurística sobre o domínio, e alguma interface amigável.*
- *Processamento da Linguagem Natural*
  - *desenvolvimento de ferramentas para a comunicação homem-máquina em geral e para a construção de interfaces.*

## Origens: lógica matemática

- *O uso da lógica na representação dos processos de raciocínio originou-se nos estudos de*
  - *Boole (1815-1864)*
  - *De Morgan (1806-1871)*
  - *Teoria posteriormente denominada de Álgebra de Boole*
- *O cálculo de predicados é o subconjunto da lógica matemática que formaliza a estrutura lógica e define o significado dos conectivos e, ou, não, se...então e outros*

## Lógica Simbólica

- *3 necessidades básicas da lógica formal:*
  - *Expressar proposições*
  - *Expressar as relações entre estas proposições*
  - *Descrever como novas proposições podem ser inferidas de outras que se presumem verdadeiras.*

## Cálculo de predicados (1)

- *Proposição – declaração lógica que pode ou não ser verdadeira. Consiste em objetos e relações de objetos entre si.*
- *Objetos – representados por termos simples,, constantes ou variáveis.*
- *Constante – representa objeto.*
- *Variável – representa diferentes objetos em diferentes tempos.*

## Cálculo de predicados (2)

- *Proposição atômicas – termos compostos.*
- *Termo composto – elemento de uma relação matemática, escrito como notação de função matemática.*
- *Composto por 2 partes:*
  - *Functor: símbolo de função que nomeia a relação; e*
  - *Lista ordenada de parâmetros.*
- *1-tupla: termo composto com 1 parâmetro*
- *2-tupla: termo composto com 2 parâmetro*

## Cálculo de predicados (3)

- *Proposições:*

  *homem(joao).  -> 1-tupla*

  *homem(fred).*

  *gosta(beto, bife). -> 2-tupla*
- *A relação homem tem 2 elementos distintos*
- *Constantes: homem, joao, gosta, beto, bife, fred*
- *Declaração de proposição:*
  - *fatos: considerada verdadeira*
  - *consultas: algo que precisa ser determinado*

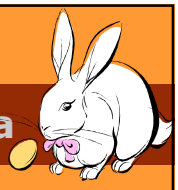
## Cálculo de predicados (4)

- *Proposições compostas – 2 ou mais proposições atômicas, ligadas por conectores lógicos ou operadores, como expressões lógicas das linguagens imperativas.*

## Conectores lógicos

| Representação | Significado |
|---|---|
| $\neg p$ | não p |
| $p \wedge q$ | p e q |
| $p \vee q$ | p ou q |
| $p \Leftrightarrow q$ | p equivale a q |
| $p \Rightarrow q$ | p implica q |

## O Modelo de Programação em Lógica



- *Base do modelo: lógica simbólica*
  - *axiomas lógicos (proposições) que representam o conhecimento e regras de dedução (inferência) de novos conhecimentos*
- *Ex. de proposições*
  - *p: um coelho é um mamífero*
  - *q: um mamífero tem sangue quente*
- *Ex. de dedução*

  *Se um coelho é um mamífero e um mamífero tem sangue quente Então um coelho tem sangue quente.*

  *Se* $p \wedge q$ *Então* $r$

## Base do modelo: lógica simbólica (1)

- *Uma relação é estabelecida entre objetos e resulta nos valores verdadeiro e falso*
  - *Ex.:*

    $a > b$ *é a relação* $R(a,b)$*, sendo* $R$ *a relação* $>$

## Base do modelo: lógica simbólica (2)

- *Programação em lógica consiste em implementar relações e formular consultas como:*
  - *Dados a e b, determinar se R(a,b) é verdadeira.*
  - *Dado a, encontrar todo y tal que R(a,y) é verdadeira.*
  - *Dado b, encontrar todo x tal que R(x,b) é verdadeira.*
  - *Encontrar todo x e todo y tal que R(x,y) é verdadeira.*

## Base do modelo: lógica simbólica (3)

- *Cláusulas de Horn*
  - *Tipos especiais de proposições usadas para resolução*
  - *Tem uma parte mais importante **h** (cabeça), que é um atributo, e um **corpo**, que é uma lista de atributos p1, p2, ... pn*
  - *Representação de proposições na forma de uma sentença "se p então q"*
    - *Representação: q ← p*

## Base do modelo: lógica simbólica (4)

- *Cláusulas de Horn*
  - *Representação pode ter 2 formas:*
    - *Cabeça (lado esq.) com única proposição atômica – declarar relações*
    - *Cabeça vazia – declarar fatos*
  - *exemplos:*

  *nevando(C)← precipitação(C)^congelando(C)*

  *pai(beto,maria)*

  *- pode usar , ao invés de ^*

## Base do modelo: lógica simbólica (5)

- *Cláusulas de Horn*
  - *4 classificações:*
    - *Incondicional*
    - *Positiva*
    - *Condicional*
    - *Negativa*

## Cláusulas de Horn (1)

- *Cláusula incondicional (fato): não contém condições. Representação: q ←*
  - *Ex.: 3 é um número inteiro*

    *inteiro(3) ←*
- *Cláusula positiva (regra): Contém uma ou mais conclusões. Representação: q ← p*
  - *Ex.: Todo número natural é um número inteiro*

    *inteiro(N) ← natural(N)*

## Cláusulas de Horn (2)

- *Cláusula condicional (regra): contém uma ou mais condições.*
  - *Representação: q ← p1 p2*
  - *Ex.: Para que um número seja natural, ele deve ser inteiro e positivo*

natural(N) ← inteiro(N) ∧ N>0

cabeça da cláusula (consequente): conclusão

corpo da cláusula (antecedente): condições a serem satisfeitas

## Cláusulas de Horn (3)

- *Cláusula negativa (consulta): não contém conclusões. Representação: ←p*
  - *Ex.: Será 3 um número inteiro?: ← inteiro(3)*
  - *Processo de dedução: Se 3 é um no. natural (fato) e todo no. natural é um inteiro, então prova-se que 3 é um número inteiro.*

**Fatos e Regras**

```
inteiro(3).
inteiro(5).
inteiro(-3).
natural(N) :- inteiro(N) , N>0.
```

**Consulta**

```
?- natural(3).
yes
?- natural(-3).
no
```

## Cláusulas de Horn (4)

- *Resolução:*
  - *Quando aplicada às cláusulas de Horn, a resolução diz que se **h** é a cabeça de uma cláusula e ela corresponde a um dos termos de uma outra cláusula, então aquele termo pode ser substituído por **h**.*

    *h ← termos*

    *t ← t1, h, t2  => t ← t1, termos, t2*

## Cláusulas de Horn (5)

- *Instanciação:*
  - *Atribuição de valores a variáveis durante a resolução.*
- *Unificação ("match"):*
  - *Processo de correspondência de padrões que determina que instanciações, em particular, podem ser feitas a variáveis ao mesmo tempo em que se faz uma série de resoluções simultâneas.*

## Modelagem de programas em lógica

- *Especifica o conhecimento a ser usado na solução do problema: fatos e regras*
- *Procedimentos de resolução, baseados em um conjunto de cláusulas e um objetivo: consulta ao programa*
- *Projeto de soluções: abordagem de redes semânticas*

## Redes semânticas (1)

*Usadas para representar relacionamentos*

- *rede semântica simples: 1 → 1*

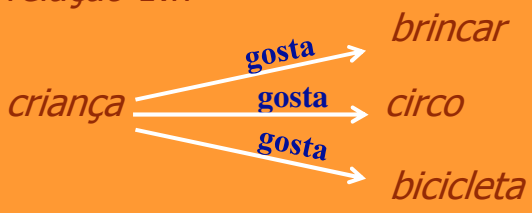
*porta* —**amassou**→ *dedo*

*criança* —**adora comer**→ *chocolate*

## Redes semânticas (2)

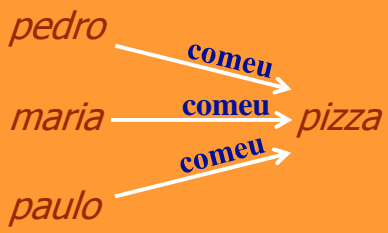
*Usadas para representar relacionamentos*

- *relação 1:n*



## Redes semânticas (3)

- *relação n:1*



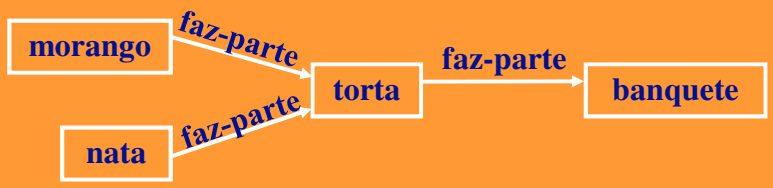

## Redes semânticas (4)

- *relação transitiva*


- *relação " é-um"*

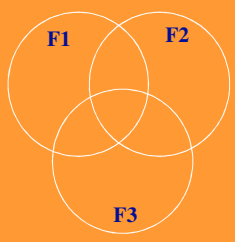
## Redes semânticas (5)

- *relacionamento "faz-parte"*




## Redes semânticas (6)

- *A abordagem de grupo*
- *Fatos:*
  - *F1: Pedro, Maria e Paulo gostam de ler*
  - *F2: Pedro, Paulo e Vera gostam de cinema*
  - *F3: Paulo, Maria e Vera gostam de boliche*
- *Questões:*
  - *X gosta de ler?*
  - *Quem gosta de boliche?*
  - *Quem gosta de boliche e cinema?*



## Programação em Lógica e Prolog

- *1º interpretador (1972, Aix-Marseille/França)*
- *Programas*
  - *Cláusulas*
    - *Fatos e regras (parâmetros em minúsculas)*
    - *Consultas (parâmetros em maiúsculas)*
  - *Programa é um "mundo fechado"*
  - *Mecanismos*
    - *Unificação ("match") e "backtracking"*

## Programas: fatos e regras (1)

- *Fatos agrupados se constituem na base de dados*

```
Fato 1: Brasília e Florianópolis são cidades.
        cidade(florianopolis).
        cidade(brasilia).
Fato 2: Brasil é um país.
        pais(brasil).
Fato 3: SC é um estado.
        estado(sc).
Fato 4: Brasília é a capital do Brasil.
        capital(brasil,brasilia).
Fato 5: Florianópolis é a capital de SC.
        capital(sc, florianopolis).
```

## Programas: fatos e regras (2)

- *Regras estabelecem relações mais complexas entre objetos*

```
capital_pais(X,Y) :- pais(X), cidade(Y), capital(X, Y).

capital_estado(X,Y) :- estado(X), cidade(Y), capital(X, Y).
```

*" Y é capital do pais X, se X **é-um** pais e Y **é-uma** cidade e Y é capital de X."*

## Exemplo em PROLOG

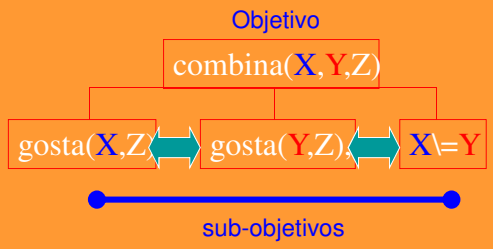
```
/* Base de Dados */
cidade(florianopolis).
cidade(brasilia).
pais(brasil).
estado(sc).
capital(brasil,brasilia).
capital(sc, florianopolis).
capital_pais(X,Y) :- pais(X), cidade(Y), capital(X, Y).
capital_estado(X,Y) :- estado(X), cidade(Y), capital(X, Y).
```

```
/* Consultas */
?- cidade(florianopolis).
   Yes
?- capital(brasil,X).
   X = brasilia ;
   No
```

```
/* Consultas */
?- capital_pais(X,brasilia).
   X = brasil ;
   No
?- capital_estado(sc,Y).
   Y = florianopolis ;
   No
```

## Mecanismo de retrocesso (*backtracking*)

```
/* base de dados*/
gosta(pedro,leitura).
gosta(maria,leitura).
gosta(paulo,leitura).
gosta(pedro,cinema).
gosta(paulo,cinema).
gosta(vera,cinema).
gosta(pedro,boliche).
gosta(maria,boliche).
gosta(vera,boliche).
```




```
/* Existe algo Z que X e Y gostam ? */
combina(X,Y,Z):- gosta(X,Z),gosta(Y,Z), X\=Y.

/* Consulta */
?- combina(maria,pedro,Z).
```

```
Z = leitura ;
Z = boliche ;
No
```

## Método de corte (CUT): !

- **Cut** (!) *é uma estrutura de controle do Prolog*
- *Útil para controlar e limitar as buscas*
  - *Quando encontrado, o objetivo é considerado satisfeito imediatamente*
  - *evita backtracking*
- *Nenhuma cláusula alternativa é considerada, portanto, o* **cut** *elimina a capacidade de retroceder*

```
combina(X,Y,Z):- gosta(X,Z),gosta(Y,Z), X\=Y, !.
?- combina(maria,pedro,Z).
```

```
Z = leitura ;
No
```

## Programação em Lógica

- *Pontos positivos*
  - *Em princípio, todos do paradigma funcional (simplicidade sintática, facilidade em descrever problemas recursivos, ...)*
  - *Permite concepção da aplicação em um alto nível de abstração (através de associações entre E/S).*
- *Pontos negativos*
  - *Certa ineficiência em comparação com linguagens de programação "tradicionais".*
  - *Linguagens usualmente não possuem tipos, nem são de alta ordem.*

## Outros exemplos de programas em Prolog

- *O programa em si:*
  - *hello_world :- write('Hello World!').*
- *A consulta:*

```
?- hello_world.
Hello World!
Yes
```

## Outros exemplos de programas em Prolog

- *Speaks:*

```
speaks(allen, russian).
speaks(bob, english).
speaks(mary, russian).
speaks(mary, english).
talkswith(P1,P2) :-
   speaks(P1,L) , speaks(P2,L), P1\=P2.
```

## Outros exemplos de programas em Prolog

- *As consultas:*

```
?- speaks(Who, russian).
Who= allen;
Who= mary;
No
?- talkswith(bob, allen).
No
?- talkswith(Who, allen).
Who= mary;
No
```

## Outros exemplos de programas em Prolog

- *Árvore genealógica:*

```
parent(A,B) :- father(A,B).
parent(A,B) :- mother(A,B).
grandparent(C,D) :- parent(C,E) , parent(E,D).

mother(mary, sue).        father(john, sue).
mother(mary, bill).       father(john, bill).
mother(sue, nancy).       father(bob, nancy).
mother(sue, jeff).        father(bob, jeff).
mother(jane, ron).        father(bill, ron).
```

## Outros exemplos de programas em Prolog

- *As consultas:*

```
?- grandparent(Who, ron).
Who= mary;
Who= john;
No
?- father(bob, Y).
Y= nancy;
Y= jeff;
No
```

## Outros exemplos de programas em Prolog

- *Outras relações a serem construídas:*

  *sibling(X,Y):- parent(W,X) , parent(W,Y), X\=Y.*

- *A consulta:*

  *?- sibling(P1, P2).*

  *P1= sue, P2=bill;*

  *P1= nancy, P2=jeff;*

  *P1= sue, P2=bill;*

  *P1= nancy, P2=jeff;*

  *No*

*PROBLEMA: respostas repetidas !!!*

## FONTES BIBLIOGRÁFICAS

*Ver plano de ensino:*

- *Linguagens de programação: princípios e paradigmas – TUCKER&NOONAN, cap15 (fotocópia)*
- *Conceitos de linguagens de programação – SEBESTA, cap16*