

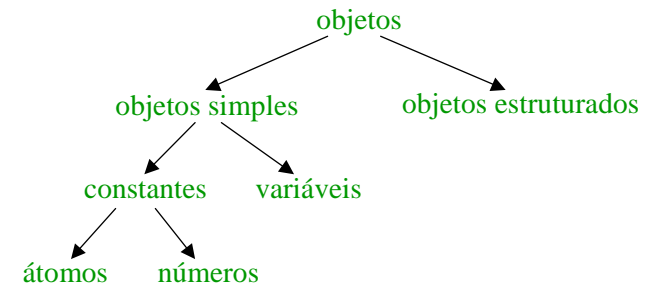
PROLOG

PROgramming in LOGic

Linguagem de Programação Baseada no Cálculo de Predicado de Primeira Ordem

- Apropriada à:
 - **processamento simbólico**, não numérico
 - resolução de problemas que envolvam objetos e relações entre objetos
- Mecanismos Básicos:
 - casamento de padrão
 - estruturas de listas
 - Retrocesso (backtracking) automático

Objetos de Dados



- **Átomos** - cadeias de letras maiúsculas, letras minúsculas, dígitos e caracteres especiais construídas como:
 - cadeias de letras, dígitos e o caracter”_” (underscore), começando com letra minúscula
 - cadeia de caracteres especiais: ::=:, <-->, etc.
 - cadeia de caracteres entre apóstrofes

Ex: `maria`, `a`, `x`, `elemento`, `a1`, `cubo_1`, `ponto_a`
- **Números** - sintaxe usual

Exemplos:

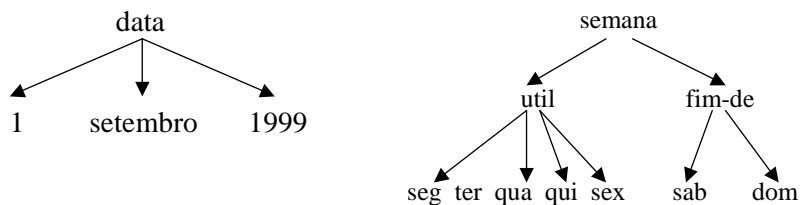
 - inteiros: `1`, `-25`, `4851`, `-9556`
 - ponto flutuante: `1.55`, `-0.55`, `84.756`, `4.1`
- **Variáveis** - cadeias de letras, dígitos e caracter “_”, começando com letra maiúscula ou com o caracter “_”

Ex: `X`, `X1`, `Lista1`, `_abc`, `YZW`, `A123`

- **Estruturas ou Objetos Estruturados** - são objetos de dados que tem vários componentes, podendo cada um deles, por sua vez ser uma estrutura.
- A combinação dos componentes é feita através do **funtor**, que dá um nome para a estrutura:
 - data(1, setembro, 1999)
 - par(primeiro,segundo)
 - semana(seg,ter,qua,qui,sex,sab,dom)
 - semana(util(seg,ter,qua,qui,sex), fim-de(sab,dom))
 - autor('Russel & Norvig')
 - livro(titulo('Inteligencia Artificial'), autor('G. Bittencourt'))

Objetos Estruturados podem ser representados em forma de árvore:

- raiz = funtor
- filhos = componentes



Relações ou Predicados

- Componente principal das construções do Prolog
- Descrevem algum tipo de ligação entre objetos

- pai_de(joao,pedro)
- bonita(maria)
- gosta_de(ana,vinho)
- maior(5,4,3,2,1)
- pequeno(gato)
- sobre(cubo_a, cubo_b)
- soma(5,2,7)

• Predicado X Estrutura

- Estruturas são formalmente idênticas aos predicados
- Todo predicado é uma estrutura
- Nem toda estrutura é um predicado
- Um predicado é uma estrutura que declara coisas que podem ser verdadeiras ou falsas
- Estruturas que nomeiam objetos não fazem declarações e não podem ser verdadeiras ou falsas

- **Um Programa Prolog consiste de:**

- fatos
- regras
- consultas

- **Fatos**

Um fato é uma declaração de que uma determinada relação existe entre certos objetos.

- pai_de(joao, pedro).
- bonita(maria).
- gosta_de(ana, vinho).
- Maior(5, 4, 3, 2, 1).
- Gosta(X, vinho). (fatos universais)
- vezes(0, X, 0).

- **Base de Dados:**

conjunto de fatos

Exemplo 1: FAMÍLIA REAL

% base de dados

pai_de(henrique_pai, henrique).	%1
pai_de(henrique_pai, maria).	%2
pai_de(henrique, elizabeth2).	%3
pai_de(henrique, eduardo).	%4
homem(henrique_pai).	%5
homem(henrique).	%6
homem(eduardo).	%7
mulher(catarina).	%8
mulher(elizabeth1).	%9
mulher(maria).	%10
mulher(elizabeth2).	%11
mulher(ana).	%12
mulher(jane).	%13
mae_de(catarina, maria).	%14
mae_de(ana, elizabeth2).	%15
mae_de(jane, eduardo).	%16
mae_de(elizabeth1, henrique).	%17

%base de conhecimento

filho_de(Y,X) :- pai_de(X,Y), homem(Y).	%18
pai_ou_mae(X,Y) :- pai_de(X,Y).	%19
pai_ou_mae(X,Y) :- mae_de(X,Y).	%20
predecessor(X,Y) :- pai_ou_mae(X,Y).	%21
predecessor(X,Y) :- pai_ou_mae(X,Z), predecessor(Z,Y).	%22

- **Consultas**

São o meio de recuperar informação em um programa lógico

1) Confirmação - a busca é realizada até encontrar uma resposta, confirmando ou negando o que foi perguntado.

| ?- pai_de(henrique,eduardo).
yes

2) Recuperação - todos os valores que satisfazem a consulta são recuperados.

| ?- pai_de(X,maria).
X = henrique_pai ;
no

| ?- pai_de(henrique,X).
X = elizabeth2 ;
X = eduardo

| ?- pai_de(X,eduardo).
X = henrique

| ?- mae_de(X,henrique).
X = elizabeth1

| ?- mae_de(X,maria).
X = catarina;
no

| ?- mae_de(catarina,X).
X = maria

| ?- mae_de(X,catarina).
no

| ?-mae_de(X,Y).
X = catarina ,
Y = maria ;
X = ana ,
Y = elizabeth2 ;
X = jane ,
Y = eduardo ;
X = elizabeth1 ,
Y = henrique

- **Consultas Compostas**

São interpretadas como conjunção.

| ?- pai_de(X,elizabeth2),pai_de(X,eduardo).

(Existe um valor para X que torne as duas partes da consulta verdadeiras ao mesmo tempo?)

X = henrique ;
no

| ?- pai_de(X,eduardo),pai_de(Y,X).

(Quem é o avô de eduardo?)

X = henrique ,
Y = henrique_pai ;
no

| ?- pai_de(henrique_pai,X),pai_de(X,Y).

(Quem são os netos de henrique_pai?)

X = henrique ,
Y = elizabeth2 ;
X = henrique ,
Y = eduardo ;
no

Unificação

- Dois termos **unificam** se:

- são idênticos, ou
- as variáveis em ambos os termos podem ser **instanciadas** em objetos de tal forma que após a substituições das variáveis por esse objetos os termos se tornam idênticos.

Exemplos:

Termo 1 : data(25,maio, Ano)

Termo 2 : data(D,maio,1983)

Resultado : D = 25

Ano = 1983

Termo 1 : data(D1,abril, A)

Termo 2 : data(D2,M,1900)

Resultado : D1 = D2

M = abril

A = 1900

Termo 1: pai_de(henrique, filhos(eduardo,elizabeth2))

Termo 2: pai_de(henrique,X)

Resultado: X = filhos(eduardo,elizabeth2)

- Regras para decidir se dois termos unificam:

- Se S e T são constantes então S e T unificam se e só se são o mesmo objeto;
- se S é uma variável e T é qualquer termo, então unificam e S é instanciada com T; vice-versa com a variável T instanciada com S;
- se S e T são estruturas, elas unificam se e só se S e T tem o mesmo funtor principal e todos os elementos correspondentes unificam.

Termo 1	Termo 2	Resultado da Unificação
henrique	henrique	unificam
eduardo	henrique	não unificam
X	par(a,b)	X = par(a,b)
2.35	Y	Y = 2.35
data(25,maio,Ano)	data(D,maio,1983)	D = 15 Ano = 1983
data(D1,abril,A)	data(D2,M,1900)	D1 = D2 M = abril A = 1900
data(17,marco,2000)	date(17,M,2000)	não unificam
pai_de(X,eduardo)	pai_de(henrique,Y)	X = henrique Y = eduardo

- **Regras**

Permitem definir novas relações em termos das já existentes.

Definir as relações de *filho* e *filha*:

`filho_de(Y,X) :- pai_de(X,Y), homem(Y).`

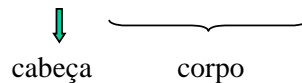
`filha_de(Y,X) :- pai_de(X,Y), mulher(X).`

Definir a relação de avô:

`avo_de(X,Z) :- pai_de(X,Y), pai_de(Y,Z).`

Forma Geral:

A :- B1, B2,, Bn.



A - objetivo, meta

Bi - sub-objetivos, condições

Para provar A provamos B1, B2,....., Bn

- **Base de Conhecimento:**

conjunto de regras

- **Cláusulas:**

Regras, fatos e consultas são chamadas Cláusulas de Horn, ou somente Cláusulas

- Como o programa Prolog usa as regras:

Colocada a consulta:

`| ?- filho_de(eduardo,henrique).`

Não há no programa fatos sobre a relação `filho_de`.

É necessário usar as regras

- A consulta é comparada com a cabeça das regras que definem a relação `filho_de`, na sequência.
- Ocorre uma unificação entre a consulta e a cabeça da regras %18, com as instanciações:

Y = eduardo

X = henrique

- A regra fica:

`filho_de(eduardo,henrique):-`

`pai_de(henrique,eduardo), homem(eduardo)`

- O objetivo é substituído pelos sub-objetivos:

`pai_de(henrique,eduardo), homem(eduardo)`

que devem ser verdadeiros ao mesmo tempo.

- Os sub-objetivos são provados, pois são fatos no programa.

Logo, o objetivo também é verdadeiro e a resposta é:

`yes.`

- **Mais de uma regra sobre a mesma relação:**

- Duas ou mais regras sobre a mesma relação indicam formas alternativas de provar um objetivo, portanto correspondem ao operador “ou”.

```

pai_ou_mae(X,Y) :- pai_de(X,Y).           %19
pai_ou_mae(X,Y) :- mae_de(X,Y).           %20

```

```

| ?- pai_ou_mae(X,elizabeth2).
X = henrique ;
X = ana ;
no

```

```

| ?- pai_ou_mae(X,eduardo).
X = henrique ;
X = jane ;
no

```

```

| ?- pai_ou_mae(jane,X).
X = eduardo

```

```

| ?- pai_ou_mae(henrique,Y).
Y = elizabeth2 ;
Y = eduardo ;
no

```

- **Regras Recursivas**

- Recursão: operação em que um objeto é usado em sua própria definição.
- Regras recursivas: definidas em termos delas mesmas.

- Exemplo: Definir a relação de *predecessor*
Considerar a base Família Real

```

pai_ou_mae(X,Y) :- pai_de(X,Y).           %19
pai_ou_mae(X,Y) :- mae_de(X,Y).           %20
predecessor(X,Y) :- pai_ou_mae(X,Y).       %21
predecessor(X,Y) :- pai_ou_mae(X,Z),
                        predecessor(Z,Y).   %22

```

```

| ?- predecessor(henrique_pai,X).
X = henrique ;
X = maria ;
X = elizabeth2 ;
X = eduardo ;
no

| ?- predecessor(X,henrique).
X = henrique_pai ;
X = elizabeth1 ;
no

```

```

| ?- predecessor(X,eduardo).
X = henrique ;
X = jane ;
X = henrique_pai ;
X = elizabeth1 ;
no

```

- Observação1) :

Variáveis DIFERENTES em uma mesma cláusula
PODEM assumir o mesmo valor

Exemplo: Definir a relação de irmã

```

pai_ou_mae(tom,bob).
pai_ou_mae(tom,liz).
pai_ou_mae(bob,ana).
pai_ou_mae(bob,pat).
pai_ou_mae(pat,jim).]
mulher(ana).
mulher(pat).
irma(X,Y) :- pai_ou_mae(Z,X),
              pai_ou_mae(Z,Y),
              mulher(Y).
```

- Consulta:

```
| ?- irma(pat,X).
```

```
X = ana ;
```

```
X = pat ;
```

```
no
```

A segunda resposta significa Pat é irmã dela mesma.

O Prolog inclui essa resposta porque não há restrição de que
X e Y devam ser diferentes.

- Observação 2) - Escopo de Variáveis:

O escopo de todas as variáveis é limitado a uma única
cláusula. Não existem variáveis globais.

```

pai_ou_mae(tom,bob).
pai_ou_mae(tom,liz).
pai_ou_mae(bob,ana).
pai_ou_mae(bob,pat).
pai_ou_mae(bob,joe).
pai_ou_mae(pat,jim).
mulher(ana).
mulher(pat).
homem(joe).
homem(bob).
irma(X,Y) :- pai_ou_mae(Z,X),
              pai_ou_mae(Z,Y),
              mulher(Y).
irmao(X,Y) :- pai_ou_mae(Z,X),
              pai_ou_mae(Z,Y),
              homem(Y).
sobrinho(X,Y) :- irma(X,Z),
                  pai_ou_mae(Z,Y).
sobrinho(X,Y) :- irmao(X,Z),
                  pai_ou_mae(Z,Y).
```


- **Backtracking**

Técnica de busca que consiste em sistematicamente testar todos os caminhos alternativos que levam a uma solução

- **Árvore AND/OR (E/OU)**

- Nós AND - representam conjunções

- Nós OR - representam disjunções (caminhos alternativos)

- A árvore AND/OR de um programa Prolog representa todos os caminhos que levam a uma solução, isto é, permitem provar o objetivo.

A :- B1, B2.

A :- B3.

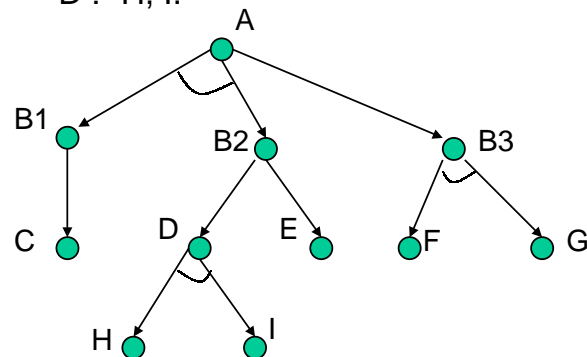
B1 :- C.

B2 :- D.

B2 :- E.

B3 :- F, G.

D :- H, I.



- **Árvore de Execução**

Árvore que representa o processo de execução de uma consulta

Nós representam objetivos (simples ou compostos)

Arcos representam substituições de um objetivo por sub-objetivos pela unificação de objetivos com fatos ou regras