Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Informática

Curso de Bacharelado em Informática

APOSTILA PROLOG

ANDRÉ ABE VICENTE
http://www.inf.unioeste.br/~abvicente

Disciplina: Inteligência Artificial (IA)
Professora: Suzan Kelly Borges Piovesan

Cascavel
Setembro de 2005

ÍNDICE

Prolog - Introdução

A programação em lógica, tal como a conhecemos hoje, tem suas raízes no cálculo de predicados, proposto por Frege em 1879. Diversos estudos posteriores foram de grande importância para sua evolução, com destaque para as investigações de Herbrand, Gödel, Tarski, Prawitz,
Robinson e Green;

A primeira implementação da linguagem Prolog foi realizada por Alain Colmerauer e sua equipe, na Universidade de Aix-Marseille em 1972. A formalização semântica da programação com cláusulas de Horn é devida a Kowalski (1974) e a especificação do primeiro "standard" - o Prolog de Edimburgo - foi realizada por Warren e Pereira em 1977;

As principais características que diferenciam os *programas em lógica* dos programas convencionais são as seguintes:

(1) Processamento simbólico,
(2) Soluções heurísticas,
(3) Estruturas de controle e conhecimento separadas,
(4) Fácil modificação,
(5) Incluem respostas parcialmente corretas, e
(6) Incluem todas as soluções possíveis;

Além disso, os sistemas de programação em lógica em geral e a linguagem Prolog em particular possuem as seguintes propriedades:

(1) Funcionam simultaneamente como linguagem de programação e de especificação,
(2)Possuem capacidade dedutiva,
(3)Operam de forma não-determinística,
(4)Permitem a representação de relações reversíveis,
(5)Permitem interpretação declarativa, procedimental e operacional, e
(6)São naturalmente recursivos;

As principais aplicação da programação em lógica são:
(1)Sistemas Baseados em Conhecimento SBCs),
(2)Sistemas de Bases de Dados (BDs),
(3)Sistemas Especialistas (SEs),
(4)Processamento da Linguagem Natural (PLN),
(5)Educação, e
(6)Modelagem de Arquiteturas Não-Convencionais;

O projeto japonês para o desenvolvimento de Sistemas Computacionais de Quinta Geração iniciou em 1982 e foi oficialmente concluído em maio de 1992. Apesar de ficarem aquém do esperado, os resultados produzidos permitem claramente antever o papel preponderante que a programação em lógica deverá representar nos futuros sistemas computacionais;

A crescente necessidade de garantir a qualidade do software substituindo programas por especificações formais diretamente executáveis, aliada à evolução das características do hardware,
que passam a explorar cada vez mais os conceitos de concorrência e paralelismo, tornam a linguagem Prolog uma excelente porta de entrada para a informática do futuro.

Programação Prolog

- A programação em Prolog consiste em estabelecer relações entre objetos e em formular consultas sobre tais relações de forma não numérica, mas simbólica.
- Um programa Prolog é formado por cláusulas. Há três tipos de cláusulas: fatos ou *assertivas*, regras ou *procedimentos* e consultas;
- Uma relação pode ser especificada por meio de:
  - fatos, que estabelecem as tuplas de objetos que satisfazem a relação,
  - regras, que estabelecem condições para a satisfação das relações,
  - combinações de fatos e regras descrevendo a relação;
- Interrogar um programa acerca de suas relações por meio de uma *consulta* corresponde a consultar uma base de conhecimento. A resposta do sistema Prolog consiste em um conjunto de objetos que satisfazem as condições originalmente estabelecidas pela consulta;

Fatos

- Um fato denota uma verdade incondicional,
- Todo fato é composto por um **predicado** que estabelece uma relação entre seus **argumentos** e encerrado por um ponto (.)*;*
- *Exemplo:*

progenitor(maria, josé).
progenitor(joão, josé).
progenitor(joão, ana).
progenitor(josé, júlia).
progenitor(josé, íris).

Regras

- Definem as condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira.
- Cláusulas Prolog desse tipo são denominadas ***regras*.**
- Há uma diferença importante entre regras e fatos:
  - Um fato é ***sempre* verdadeiro**;
  - Regras especificam algo que ***"pode ser verdadeiro se algumas condições forem satisfeitas".***
- As regras tem:
  - Uma parte de *conclusão* (o lado esquerdo da cláusula - cabeça), e uma parte de *condição* (o lado direito da cláusula - corpo). O símbolo ":-" significa *"se"* e separa a cláusula em *cabeça* e *corpo* da cláusula.
    - Se a condição expressa pelo corpo da cláusula - *progenitor (X, Y)* - é verdadeira então, segue como conseqüência lógica que a cabeça - *filho(Y, X)* - também o é.

- Por outro lado, se não for possível demonstrar que o corpo da cláusula é verdadeiro, o mesmo irá se aplicar à cabeça.

- Exemplo: *pode-se declarar a relação de filho através de simples fatos como*
filho(josé, joão).

Entretanto podemos definir a relação *"filho"* de uma maneira muito mais elegante, fazendo o uso do fato de que ela é o inverso da relação *progenitor* e esta já está definida. Tal alternativa pode ser baseada na seguinte declaração lógica:

Para todo X e Y
Y é filho de X se
X é progenitor de Y.

A declaração acima, escrita em Prolog fica:

filho(Y, X) :- progenitor(X, Y).

"Para todo X e Y, se X é progenitor de Y, então Y é filho de X".

SINTAXE E SEMÂNTICA

OBJETOS

- O sistema reconhece o tipo de um objeto no programa por meio de sua forma sintática.

- Variáveis sempre irão iniciar com letras maiúsculas, enquanto que as constantes não-numéricas, ou átomos, iniciam com letras minúsculas.

- Nenhuma informação adicional, tal como *tipos* de dados precisa ser fornecida para que o sistema reconheça a informação com a qual está lidando.

ÁTOMOS E NÚMEROS

- Os argumentos das relações podem ser constantes (como júlia e íris), denominados atomos, ou objetos genéricos denominados variáveis (como X e Y).
- O alfabeto básico adotado aqui para a linguagem Prolog consiste dos seguintes símbolos:
  - Pontuação: ( ) . ' "
  - Conetivos: , (conjunção e)
    ; (disjunção ou)
    :- (implicação se)
  - Letras: a, b, c, ..., z, A, B, C, ..., Z
  - Dígitos: 0, 1, 2, ..., 9
  - Especiais: + - * / < > = : _ ... etc.

Os ***átomos*** podem ser construídos de três maneiras distintas:

a. Como cadeias de letras e/ou dígitos, podendo conter o caracter especial

sublinhado (_), iniciando obrigatoriamente com letra minúscula. Por exemplo:

```
socrates  x_y
nil       mostraMenu
x47       a_b_1_2
```

b. Como cadeias de caracteres quaisquer, podendo inclusive incluir espaços em branco, desde que delimitados por apóstrofos ('). Por exemplo:

```
'D. Pedro I'
'representação de conhecimento'
'13 de outubro de 1993'
'Robert Kowalski'
```

- Os **números** usados em Prolog compreendem os **números inteiros** e os números reais. A sintaxe dos números inteiros é bastante simples: `1 1812 0 -273`

- Já a sintaxe para **números reais** é : `3.14159 0.000023 -273.16`

VARIÁVEIS

- Variáveis Prolog são cadeias de letras, dígitos e do caracter sublinhado (_), devendo iniciar com este ou com uma letra *maiúscula.* Exemplos de variáveis são:

```
X
Resultado
Objeto2
Lista_de_Associados
_var35
_194
_       (variável anônima)
```

- **O *escopo* de nomes de variáveis** é apenas uma cláusula (ocorre em duas cláusulas diferentes, representa duas variáveis, ocorre dentro da mesma cláusula significar a mesma variável)

- **O *escopo* de Constantes:** o mesmo átomo sempre significa o mesmo objeto ao longo de todo o programa.

ESTRUTURAS

- Objetos estruturados, ou simplesmente *estruturas,* são objetos que possuem vários componentes.

- Para combinar os componentes em uma estrutura é necessário empregar um *functor(*é um símbolo funcional que permite agrupar diversos objetos em um único objeto estruturado).

- Um functor adequado ao exemplo dado é *data,* então a data correspondente a 13 de

outubro de 1993, pode ser escrita como:

```
data(13, outubro, 1993)
```

- Todos os componentes no exemplo são constantes (dois inteiros e um átomo), entretanto, podem também ser variáveis ou outras estruturas. Um dia qualquer de março de 1996, por exemplo, pode ser representado por:

```
data(Dia, março, 1996)
```

- Sintaticamente todos os objetos em Prolog são denominados *termos*. O conjunto de *termos Prolog*, , é o menor conjunto que satisfaz às seguintes condições:
  - Toda constante é um termo;
  - Toda variável é um termo;
  - Se t1, t2, ..., tn são termos e f é um átomo, então f(t1, t2, ..., tn) também é um termo, onde o átomo f desempenha o papel de um *símbolo funcional n-ário.* Diz-se ainda que a expressão f(t1, t2, ..., tn) é um termo funcional Prolog.

Consultas

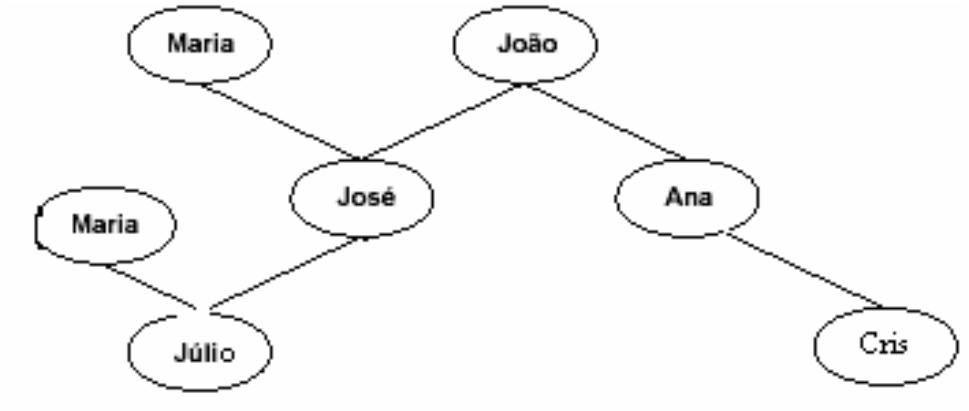
- Uma consulta em Prolog é sempre uma seqüência composta por um ou mais objetivos precedida por um ?-.
- Para obter a resposta, o sistema Prolog tenta satisfazer todos os objetivos que compõem a consulta, interpretando os como uma conjunção.
- Satisfazer um objetivo significa demonstrar que esse objetivo é verdadeiro, assumindo que as relações que o implicam são verdadeiras no contexto do programa.
- Se a questão também contém variáveis, o sistema Prolog deverá encontrar ainda os objetos particulares que, atribuídos às variáveis, satisfazem a todos os sub-objetivos propostos na consulta.
- A particular instanciação das variáveis com os objetos que tornam o objetivo verdadeiro é então apresentada ao usuário.
- Se não for possível encontrar, no contexto do programa, nenhuma instanciação comum de suas variáveis que permita derivar algum dos sub-objetivos propostos então a resposta será "não".
- O sistema Prolog aceita os fatos e regras como um conjunto de axiomas e a consulta do usuário como um teorema a ser provado.
- A tarefa do sistema é demonstrar que o teorema pode ser provado com base nos axiomas representados pelo conjunto das cláusulas que constituem o programa.

Exercícios:

1) Escreva um programa Prolog para representar o seguinte:

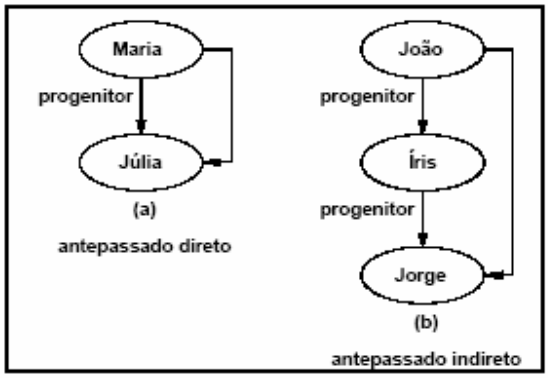
1. *João nasceu em Pelotas e Jean nasceu em Paris.*
2. *Pelotas fica no Rio Grande do Sul.*
3. *Paris fica na França.*
4. Só é gaúcho quem nasceu no Rio Grande do Sul.

2) Escreva um programa em prolog, utilizando a arvore genealógica acima, para representas as relações de : pai, mãe, filho, filha, tio, tia, primo, avo e cunhada.



RECURSÃO

Para exemplificar o funcionamento serão construídas duas relações de antepassados: antepassados direto e indireto.



Direto (pai):

```
Para todo X e Z
X é antepassado de Z se
X é progenitor de Z.
```

Em Prolog:

```
antepassado(X, Z) :-
progenitor(X, Z).
```

Avô:

```
antepassado(X, Z) :-
progenitor(X, Y),
progenitor(Y, Z).
```

Bisavô

```
antepassado(X, Z) :-
progenitor(X, Y1),
progenitor(Y1, Y2),
progenitor(Y2, Z).
```

Tataravô

```
antepassado(X, Z) :-
progenitor(X, Y1),
progenitor(Y1, Y2),
```

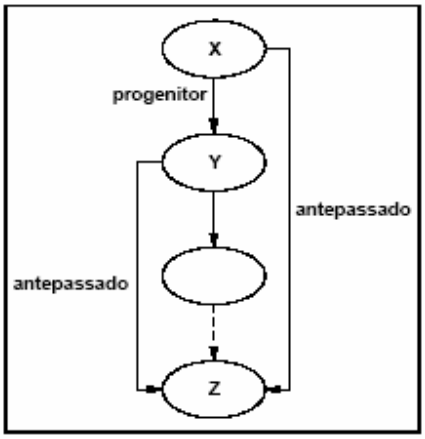
- Construção muito grande;
- Só funciona até um determinado limite (profundidade)

Recursão:

- Correta para a relação *antepassado* que não apresenta qualquer limitação.
- Definir a relação em termos de si própria;

```
Para todo X e Z
X é antepassado de Z se
existe um Y tal que
```

```
progenitor(Y2, Y3),          X é progenitor de Y e
progenitor(Y3, Z).           Y é antepassado de Z.
Etc...
```




```
Ou em Prolog:

antepassado(X, Z) :-
progenitor(X, Y),
antepassado(Y, Z).
```

UNIFICAÇÃO

- A operação mais importante entre dois termos Prolog é denominada *unificação*.

- Realizado quando do acionamento do processo de "prova" realizada pelas consultas ao sistema.

- Toma dois termos como entrada e verifica se eles podem ser unificados. Se os termos não unificam, dizemos que o processo *falha.*

- Se unificam, então o processo é bem-sucedido e as variáveis dos termos que participam do processo são instanciadas com os valores encontrados para os objetos, de modo que os dois termos participantes se tornam idênticos.

- Dados dois termos, diz-se que eles *unificam* se:

(1)Eles são idênticos, ou

(2)As variáveis de ambos os termos podem ser instanciadas com objetos de maneira que, após a substituição das variáveis por esses objetos, os termos se tornam idênticos.

As regras gerais que determinam se dois termos S e T unificam são as seguintes:

- Se S e T são constantes, então S e T unificam somente se ambos representam o mesmo objeto;
- Se S é uma variável e T é *qualquer coisa,* então S e T unificam com S instanciada com T. Inversamente, se T é uma variável, então T é instanciada com S;
- Se S e T são estruturas, unificam somente se:

  (1) S e T tem o mesmo functor principal, e

  (2) Todos os seus componentes correspondentes também unificam.

Exemplos:

1) Por exemplo, os termos data(D, M, 1994) e data(X, março, A) unificam. Uma instanciação que torna os dois termos idênticos é:

D é instanciada com X;
M é instanciada com março;
A é instanciada com 1994.

- Por outro lado, os termos data(D, M, 1994) e data(X, Y, 94) não unificam, assim como não unificam data(X, Y, Z) e ponto(X, Y, Z).

2)

```
triângulo(ponto(1, 1), A, ponto(2, 3))
```

com

```
triângulo(X, ponto(4, Y), ponto(2, Z))
```

- O processo de unificação começa pela predicado (o *functor* principal). Como ambos os functores unificam, o processo parte para a unificação dos argumentos, onde a unificação dos pares de argumentos correspondentes ocorre. Assim o processo completo pode ser visto como a seguinte seqüência de operações de unificação simples:

```
triângulo = triângulo
   ponto(1, 1) = X
   A = ponto(4, Y)
ponto(2, 3) = ponto(2, Z)
```

O processo completo de unificação é bem sucedido porque todas as unificações na seqüência acima também o são. A instanciação resultante é:

```
X = ponto(1, 1)
A = ponto(4, Y)
Z = 3
```

e.

OPERADORES E ARITMÉTICA

Tipos de operadores do prolog

- Os nomes dos operadores são átomos e sua prioridade encontra-se delimitada por valores inteiros cujo intervalo depende da implementação.

- Assumiremos aqui que esse intervalo varie entre 1 e 1200. Há três tipos básicos de operadores, conforme a tabela abaixo:

Tipos de Operadores Prolog

OPERADORES	TIPO		
Infixos	xfx	xfy	yfx
Prefixos	fx	fy	-
Posfixos	xf	yf	-

Operadores Pré-definidos

```
   :- op(1200, xfx, ':-').
   :- op(1200, fx   [':-', '?-']).
   :- op(1100, xfy, ';').
   :- op(1000, xfy, ',').
   :- op( 700, xfx, [is, =, <, >, =<, >=,
==, =\=, \==, =:=]).
   :- op( 500, yfx, [+, -]).
   :- op( 500, fx,  [+, -, not]).
   :- op( 400, yfx, [*,/,div]).
   :- op( 300, xfx, mod).
   :- op( 200, xfy, ^).
```

- Prioridade se dará da esquerda para direita que pode ser modificada com parênteses ( ).
- Operadores também podem ser definidos pelo programador, sendo realizada por um tipo especial de cláusulas, denominadas *diretivas*, que atuam como definidoras de operadores. Exemplo:

```
:- op(600, xfx, tem).
```

Isso informa ao sistema que se deseja usar *tem* como um operador de prioridade 600 e cujo tipo é "xfx".

ARITMÉTICA

- O enfoque de Prolog é a computação simbólica, onde as necessidades de cálculo são comparativamente modestas.
- Assim, o instrumental da linguagem Prolog destinado a computações numéricas é algo simples . Alguns operadores pré-definidos:
-

OPERADOR	PRIORIDADE	TIPO	SIGNIFICADO
+	500	yfx	adição
-	500	yfx	subtração

*	400	yfx	multiplicação
/	400	yfx	divisão
div	400	yfx	divisão inteira
mod	300	xfx	resto da divisão inteira
^	200	xfy	potenciação

- Tais operadores, excepcionalmente, *executam* uma certa operação, porém é necessário introduzir uma indicação adicional para ordenar a execução a ação necessária através do operador especial **"is".**
- Exemplo

```
?-X is 1 + 2.
X = 3

?-X is 3/2, Y is 3 div 2.
X=1.5     Y=1
```

- O argumento à esquerda do operador "is" deve ser um objeto simples. O argumento à direita deve ser uma expressão aritmética, composta de operadores aritméticos, variáveis e números.

Operadores de Comparação (não precisam do is)

OPERADOR	PRIORIDADE	TIPO	SIGNIFICADO
>	700	xfx	maior que
<	700	xfx	menor que
>=	700	xfx	maior ou igual a
=<	700	xfx	menor ou igual a
=:=	700	xfx	valores iguais
=\=	700	xfx	valores diferentes

- o operador acima também forçam a avaliação de expressões.

```
?-277 * 37 > 10000.
sim
```

Funções Pré-Definidas em Prolog

FUNÇÃO	SIGNIFICADO
abs(X)	Valor absoluto de X
acos(X)	Arco-cosseno de X
asin(X)	Arco-seno de X
atan(X)	Arco-tangente de X
cos(X)	Cosseno de X
exp(X)	Valor de "e" elevado a X
ln(X)	Logaritmo natural de X
Log(X)	Logaritmo decimal de X
Sin(X)	Seno de X
sqrt(X)	Raiz quadrada de X

tan(X)	Tangente de X
round(X,N)	Arredonda X para N casas decimais
Pi	Valor de p com 15 casas decimais
Random	Um número aleatório entre 0 e 1

Exemplos

a) Computar o máximo divisor comum de dois números. Dados dois inteiros positivos, X e Y, seu máximo divisor comum D pode ser encontrado segundo três casos distintos:

(1)Se X e Y são iguais, então D é igual a X;

(2)Se X<Y, então D é igual ao mdc entre X e a diferença X-Y;

(3)Se X>Y, então cai-se no mesmo caso (2), com X substituído por Y e vice-versa.

As três cláusulas Prolog que que expressam os três casos acima são:

```
mdc(X, X, X).
mdc(X, Y, D) :-
      X < Y,
      Y1 is Y-X,
      mdc(X, Y1, D).
mdc(X, Y, D) :-
      X > Y,
      mdc(Y, X, D).
```

Informática
Inteligência Artificial
Suzan Kelly Borges Piovesan

Lista de Exercícios 001

Nome _____

1) Escreva um predicado `valida_compra/3` que verifica se uma compra de um cliente é válida. Os argumentos devem ser o cliente, o item e a quantidade solicitada. O predicado deve ter sucesso apenas se o cliente for um cliente válido e com uma boa taxa de crédito e quantidade do item esteja em estoque em estoque e a quantidade solicitada seja inferior ao estoque disponível.

2) Escreva um predicado `bhaskara/3` que deve receber os três coeficientes de um polinômio de grau dois e apresentar as duas raízes possíveis. Utilize a fórmula de Bháskara, que é dada por:

$$x_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

$$x_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

3) Escreva um predicado `maior/3` que recebe dois valores nos dois primeiros argumentos e retorna o maior deles no terceiro argumento.

4) Escreva um predicado `volume_esfera(R,Volume)` que devolve em Volume o volume da esfera de raio R. O volume é dado por $V = \frac{4}{3}\pi R^3$

5) Escreva um predicado distancia(X1,Y1,X2,Y2,D) onde D deve ser a distância entre os pontos (X1,Y1) e (X2,Y2) no plano cartesiano. A distância é dada por $D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$

6) Se o peso de um corpo é dado por $\vec{P} = m\vec{g}$, onde m é a massa indicada em Kg e g é aceleração da gravidade. Considerando que a aceleração da gravidade na Terra é de 9,8 m/s^2 e a aceleração na Lua é de , escreva um predicado

peso(Massa,Local,Peso) que deve indicar o peso de um corpo, sendo informada sua massa e o local onde tal corpo se encontra.

7) Uma pessoa empresta uma certa quantia em dinheiro a uma taxa de juros de 12,5% ao mês, se o montante emprestado for inferior a $1000,00 e 9,8% em caso contrário. Escreva um predicado *débito(Montante,Meses)* que escreve o valor devido pelo tomador do empréstimo.

```
%LISTA 1 ////////////////////////////////
%EX. 1
nasceu(joao,pelotas).
nasceu(jean,paris).
fica(pelotas,rio_grande_sul).
fica(paris,franca).
gaucho(X):-
           nasceu(X,Y),
           fica(Y,rio_grande_sul).

%EX. 2
progenitor(joao,jose).
progenitor(joao,ana).
progenitor(maria1,jose).
progenitor(jose,julio).
progenitor(maria2,julio).
progenitor(ana,cris).
femea(maria1).
femea(maria2).
femea(ana).
femea(cris).
macho(joao).
macho(jose).
macho(julio).
pai(X,Y):-
          macho(X),progenitor(X,Y).
mae(X,Y):-
          femea(X),progenitor(X,Y).
filho(X,Y):-
          macho(X),progenitor(Y,X).
filho(X,Y):-
          femea(X),progenitor(Y,X).
tio(X,Y):-
          macho(X),progenitor(Z,X),progenitor(Z,W),progenitor(W,Y).
tia(X,Y):-
          femea(X),progenitor(Z,X),progenitor(Z,W),progenitor(W,Y).
primo(X,Y):-

macho(X),progenitor(Z,I),progenitor(Z,K),progenitor(I,X),progenitor(K,Y).
avo(X,Y):-
          macho(X),progenitor(X,K),progenitor(K,Y).
casado(X,Y):-
             progenitor(X,K),progenitor(Y,K).
cunhada(X,Y):-
          femea(X),casado(X,K),filho(K,Z),filho(Y,Z).
cunhada(X,Y):-
          femea(X),casado(Y,K),filho(K,Z),filho(X,Z).

%LISTA ARITIMÉTICA ////////////////////////////////

%EX. 1
cliente(joao,700).
cliente(maria,1000).
item(sabao,20,5).
item(bombril,80,10).

validacompra(Cli,Item,Qtdsol):-
                                  cliente(Cli,Cred),
                                  item(Item,Qtd,Preco),
```

```
                                    Qtdsol =< Qtd,
                                    (Qtdsol*Preco) =< Cred.
%Ex. 2
bhaskara(A,B,C,R1,R2):-
        D is ((B*B)-(4*A*C)),
        D >= 0,
        R1 is ((-B+sqrt(D))/2*A),
        R2 is ((-B-sqrt(D))/2*A).

%Ex. 3
predicado(X,Y,Z):-
       ((X>Y),Z is X);
       ((Y>X),Z is Y).

%Ex. 4
volume_esfera(R,Volume):-
      Volume is 4/3*3.14*R*R*R.

%Ex. 5
distancia(X1,Y1,X2,Y2,D):-
      Z is ((X2-X1)*(X2-X1))+((Y1-Y2)*(Y1-Y2)),
      Z >= 0,
      D is sqrt(Z).

%Ex. 6
aceleracao_gravitacional(terra,9.8).
aceleracao_gravitacional(lua,2.1).
peso(Massa,Local,Peso):-
     aceleracao_gravitacional(Local,G),
     Peso is (Massa*G).

%Ex. 7
debito(Montante,Meses,Divida):-
     ((Montante < 1000), Juro is (9.8/100));
     ((Montante >= 1000), Juro is (12/100)),
     Divida is (2000*Juro*Meses).
```

LISTAS

[Ver slides]

```
% Author:
% Date: 5/9/2005
 %write(codigo demonstracao de codigo funcoes lista).

% Ocorrencia de Elementos
membro(X,[X|_]).
membro(X,[_|C]):-
     membro(X,C).

%Concatenaçao de Listas
conc([], L, L).
conc([X | L1], L2, [X | L3]) :-
        conc(L1, L2, L3).

% Remocao
remover(X,[X|C],C).
remover(X,[Y|C],[Y|D]):-
    remover(X,C,D).

%Inversao de Listas
```

```
inverter([], []).
inverter([X | Y], Z) :-
   inverter(Y, Y1),
   conc(Y1, [X], Z).

%Sublistas
sublista(S, L) :-
   conc(L1, L2, L),
   conc(S, L3, L2).

%permutacao

permutar([], []).
permutar([X | L], P) :-
  permutar(L, L1),
  inserir(X, L1, P).

%tamanho

tamanho([_ | R], N) :-
tamanho(R, N1),
N is N1+1.

%soma
soma([], 0).
soma([X | Y], S) :-
soma(Y, R),
S is R+X.

%produto

produto([], 0).
produto([X], X).
produto(L, P) :-

prod(L, P).
prod([], 1).
prod([X | Y], P) :-
prod(Y, Q),
P is Q*X.

mostra(X,Y):-
% write('digite seu nome e depois digite .'),

 write('seu nome é'),
 nl,
 write(X),
 write('sua idade é'),
 write(Y).
```

Apêndice A [Slides Representação de Conhecimento]

Apêndice B [Slides Listas]