Representação do Conhecimento e Raciocínio
Programação Lógica
Prolog – parte 2

Inteligência Artificial - 2020/1

Prolog

- Aritmética em prolog
- Listas
- Unificação de listas
- Operações sobre listas

Prolog - Aritmética

- Operadores aritméticos são considerados funtores
  - 2+5 é representado internamente como +(2,5)
  - Para ativar as operações é necessário usar o predicado IS:
  - Sintaxe: X is <expressão>

    onde X (variável)

    <expressão> (expressão aritmética)
  - calcula a expressão e instancia o resultado com a variável X

Prolog - Aritmética

- Exemplos:

```
?- X is 1+2.
X = 3
?- Y is 1+5*(4-2).
Y=11
?- X is 4/2.
X = 2
?- X is 1+2.
X = 3
```

OBS: O predicado IS **NÃO** faz a operação de ATRIBUIÇÃO!

Prolog - Aritmética

- Alguns operadores que podem ser usados com is:

X+Y	abs(X)
X-Y	exp(X)
X*Y	ln(X)
X/Y	log(X)
X//Y (divisão inteira)	sin(X)
X^Y (exponenciação)	cos(X)
-X	sqrt(X)
X mod Y	

Prolog – Operadores Relacionais

- E1 e E2 devem ser expressões aritméticas, que são calculadas antes da aplicação do operador

E1 > E2	
E1 < E2	
E1 >= E2	
E1 =< E2	
X is E1	calcula E1 e unifica o resultado com X
E1 =:= E2	calcula E1 e E2 e testa igualdade
E1 =\= E2	calcula E1 e E2 e testa desigualdade

Prolog – Operadores Relacionais

- Exemplos

```
?- 2+1 < 6-2.
true.

?- 2+1 > (8/4)+5.
false.

?- 1+2 =:= 2+1.
true.

?- 1+2 =:= X.
ERROR: =:=/2: Arguments are not sufficiently instantiated
```

Prolog – Operadores Relacionais

- Comparação entre termos
  - Predicado = (unifica termos)
  - Sintaxe: Termo1 = Termo2
    onde Termo1 e Termo2 podem ou não estar instanciados
- Retorna sucesso se os termos Termo1 e Termo2 unificam.
- Retorna os valores das variáveis instanciadas, quando elas aparecem em um dos termos.

Prolog – Operadores Relacionais

- Comparação entre termos

```
?- 5 = 5.
true
?- fred = fred.
true.
?- X = Y.
X = Y.
?- pai_de(joao,paulo) = pai_de(X,Y).
X = joao,
Y = paulo.
```

```
?- X = 2+5.
X = 2+5
?- X is 2+5.
X = 7
?- 1+2 = 2+1.
false.
?- 1+2 =:= 2+1.
true.
```

Prolog – Operadores Relacionais

- Comparação entre termos

  - Predicado = = (verifica se dois termos são idênticos)

  - Sintaxe: Termo1 == Termo2

    onde Termo e Termo2 podem ou não estar instanciados

- Retorna sucesso se Termo 1 e Termo2 são idênticos.
- As variáveis NÃO são instanciadas.
- As expressões NÃO são calculadas.

Prolog – Operadores Relacionais

- Comparação entre termos

```
| ?- nome == nome.
true.

?- X == X.
true.

?- X == 5.
false.

?- X = 5.
X = 5
```

```
?- X == Y.
False.

?- pred(1) == pred(X).
false.

?- pred(1) = pred(X).
X = 1.
```

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

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

?- X == 2+1.
false.

?- X =:= 2+1.
ERROR: =:=/2: Arguments are
not sufficiently instantiated
```

Prolog – Operadores Relacionais

- Comparação entre termos
  - Predicado \= = (verifica se dois termos não são idênticos)
  - Sintaxe: Termo1 \= = Termo2
- Retorna sucesso se Termo 1 e Termo2 NÃO são idênticos.
- As variáveis NÃO são instanciadas.
- As expressões NÃO são calculadas.

Prolog – Operadores Relacionais

- Comparação entre termos

```
| ?- X \= = Y.
true.

?- X \= = 5.
true.

?- X \= = X.
false.
```

Prolog – Listas

- Listas
  - Principal estrutura da linguagem Prolog
  - É uma sequência ordenada de elementos
  - Pode ter qualquer comprimento
  - Elementos de listas podem ser simples ou estruturados (inclusive listas)

- No Prolog, geralmente são denotadas por colchetes e elementos separados por vírgulas

[ ] (lista vazia)

[a, b, c]

[maria, joao, pedro, carlos]

[1, 329, -15, par(a,b), X, [2, c, Y], 2000]

Prolog – Listas

- Listas são divididas em:
  - cabeça - primeiro elemento
  - cauda - o que resta tirando o primeiro elemento
- Exemplos:

[a, b, c]
cabeça: a
cauda: [b,c]

[X, Y, 234, abc]
cabeça: X
cauda: [Y, 234, abc]

Prolog – Listas

- Representação interna das listas

  - As partes da lista são combinadas pelo funtor • (ponto):
    - (Cabeça, Cauda)

  - Essa forma é usada como representação interna, por motivo de padronização da linguagem.

  - Para a programação usamos a forma sintática abreviada, que é equivalente:

[a, b, c] equivale a • (a, • (b, • (c, [ ])))

Prolog – Listas

- Padrão de listas
  - Padrão de lista é uma representação genérica em que a barra vertical separa a cabeça da cauda da lista
  - [X|Y] - lista com cabeça X e cauda Y
    - Representa listas com pelo menos um elemento
  - A barra vertical pode separar também mais de um elemento no início da lista do restante da lista
  - [X,Y | Z] - lista com elementos X e Y e cauda Z
    - Representa listas com pelo menos dois elementos

Prolog – Listas

- Padrão de listas
  - Símbolos antes da barra são ELEMENTOS
  - Símbolo após a barra é LISTA

Prolog – Listas

- Unificação de listas

  - Os padrões de listas são muito utilizados nas operações de unificação

  - Lista 1: [a1, a2, a3, a4]
  - Lista 2: [X | Y]
  - Resultados da unificação:

```
?- [a1,a2,a3,a4] = [X|Y].
X = a1,
Y = [a2, a3, a4].
```

Lista 1	Lista 2	Resultado
[a1, a2, a3, a4]	[X | Y]	X = a1 Y = [a2, a3, a4]
[a1]	[X | Y]	X = a1 Y = [ ]
[ ]	[X | Y]	não unifica
[[a, b], c, d]	[X | Y]	X = [a, b] Y = [c, d]
[[ana, Y] | Z]	[[X, foi], ao, cinema]	X = ana Y = foi Z = [ao, cinema]
[a, b, c, d]	[X, Y | Z]	X = a Y = b Z = [c, d]
[ana, maria]	[X, Y | Z]	X = ana Y = maria Z = [ ]
[ana, maria]	[X, Y, Z]	não unifica

Prolog – Listas

- Operações sobre listas

  - Operações sobre listas frequentemente usam busca recursiva.

  - São o mecanismo principal para programação em Prolog.

  - O programa é construído com base nas duas partes da lista: *cabeça* e *cauda*.

Prolog – Listas

- Operações sobre listas
  - Como construir programas que realizam operações sobre listas?
  - PROBLEMA: Verificar se um elemento é membro de uma lista.
  - O raciocínio para construção de um programa em Prolog começa com a identificação de parâmetros envolvidos (listas, estruturas, elementos, etc)
  - Todos os elementos envolvidos serão argumentos de uma relação que define a operação principal
  - A operação principal é, portanto, definida como uma relação entre os parâmetros envolvidos.

Prolog – Listas

- Operações sobre listas
  - Como construir programas que realizam operações sobre listas?

  - PROBLEMA: Verificar se um elemento é membro de uma lista.
  - No exemplo colocado, temos dois parâmetros (objetos):
    - Lista
    - Elemento
  - A operação principal (predicado) é a verificação de pertinência ou não do elemento à lista

Prolog – Listas

- Operações sobre listas
  - Como construir programas que realizam operações sobre listas?

  - PROBLEMA: Verificar se um elemento é membro de uma lista.
  - É necessário definir nomes para a relação e para os parâmetros:
    - Relação: pertence
    - Parâmetros: X, L
  - Depois de definidos os elementos envolvidos, estruturamos a solução como um processo recursivo, explorando o recurso de acessar diretamente a cabeça da lista e a cauda da lista

Prolog – Listas

- Operações sobre listas
  - Como construir programas que realizam operações sobre listas?

  - PROBLEMA: Verificar se um elemento é membro de uma lista.
  - Estruturação da solução:
    - X é membro de L se:

      - X é a cabeça de L, ou
      - X é membro da cauda de L

Prolog – Programas que usam listas

- Programa "pertence"

```
pertence(X, [X | Y ]).              % cláusula 1

pertence(X, [ Z | Y]) :- pertence(X,Y).      % cláusula 2
```

Prolog – Programas que usam listas

- Programa "pertence"
  - Após definir o programa é possível consultá-lo:

?- pertence(a, [1,2,a,c,b]).
true.

?- pertence(a, [1,2,3]).
false.

?- pertence(a, [1,2,3,[a, b, c], 4]).
false.

```
pertence(X, [X | Y ]).   % cláusula 1

pertence(X, [ Z | Y]) :- pertence(X,Y).      % cláusula 2
```

Prolog – Programas que usam listas

- Programa "pertence"

  - Um programa Prolog pode ser consultado de várias formas

  - Esse programa, escrito para verificar se um elemento pertence a uma lista, pode ser usado para recuperar todos os elementos da lista

```
?- pertence (X, [a,b,c]).
X = a ;
X = b ;
X = c ;
false.
```

```
pertence(X, [X | Y ]).   % cláusula 1

pertence(X, [ Z | Y]) :- pertence(X,Y).      % cláusula 2
```

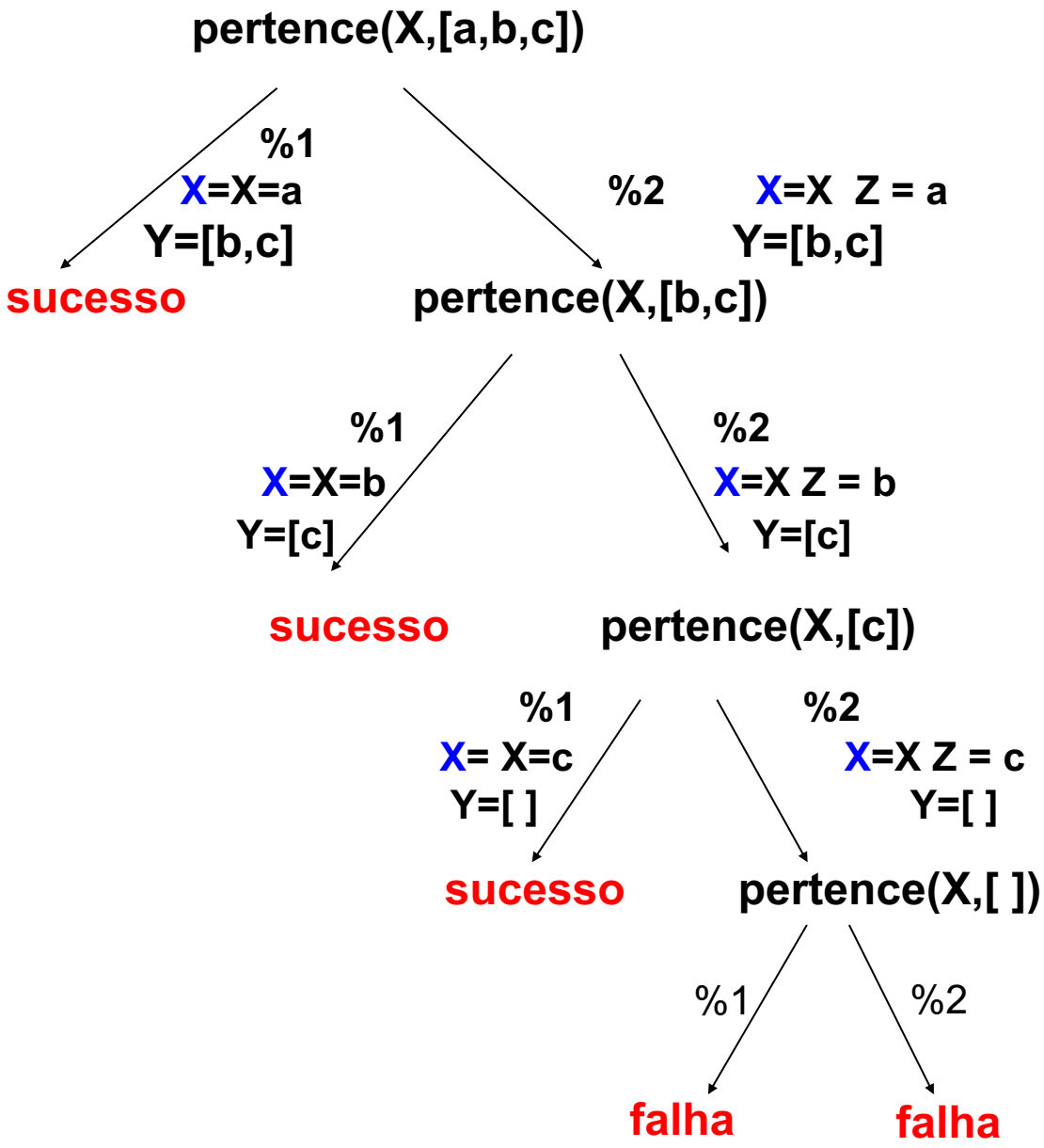
Prolog – Programas que usam listas

```
pertence(X, [X | Y ]).     % 1

pertence(X, [ Z | Y]) :- pertence(X,Y). % 2


Consulta:

?- pertence(X,[a,b,c]).
X = a;
X = b;
X = c;
false.
```



Prolog – Programas que usam listas

- Variável anônima

- A variável anônima pode ser representada em Prolog pelo caracter _ (undescore)

- Essa variável pode ser usada sempre que o valor instanciado em algum ponto da execução do programa não será utilizado futuramente. Sua função é a eficiência, por reduzir uso de memória e processamento.

- O programa que usa variável anônima gera um resultado equivalente ao que não usa.

Prolog – Programas que usam listas

- Variável anônima no programa "pertence"

```
pertence(X, [X| _ ]).                % cláusula 1

pertence(X, [ _ | Y]) :- pertence(X,Y).      % cláusula 2
```

- Próxima aula:

- Mais exemplos de programas que utilizam listas