Prolog - Recursão

Usado desde arte (em figuras, telas, etc.), em matemática e em programação

Prolog - Recursão

Prolog - Recursão

Prolog - Recursão

- A idéia de qualquer algoritmo recursivo é simples:
  - Se a instancia do problema é pequena, resolva-a diretamente.
  - Se a instancia é grande, reduza-a a uma instância menor do mesmo problema
- Embora possa confundir programadores iniciantes, usar recursividade com eficiência é uma marca de um bom programador
- Grande importância na Computação
  - Linguagens de programação puramente recursivas
    - Prolog
    - *Lisp* (Auto-cad é programado em *Lisp*).

Prolog - Recursão

- Um método recursivo é capaz de resolver apenas os casos básicos
  - Caso mais simples de um problema (caso base ou caso básico)
- Se a chamada do método for um caso básico, o método retorna resultado

Prolog - Recursão

- Se o método for chamado para um caso mais complexo...
  - Ele divide o problema em duas partes conceituais
    - Uma que o método sabe resolver
    - Outra que o método não sabe resolver
  - A parte do problema que o método não sabe resolver é uma versão mais simples do problema original
  - O método chama uma nova cópia dele mesmo para resolver o problema menor (***chamada recursiva*** ou ***passo de recursão***)
  - O processo se repete até que reste apenas o caso base
  - Ao final de cada *passo de recursão* o método retorna o resultado que é combinado com a parte do problema que o método chamador já havia solucionado para dessa forma chegar a solução do problema todo

Prolog - Recursão

- Os predicados em Prolog podem ser definidos recursivamente
- Um predicado é definido recursivamente se uma ou mais regras em sua definição refere-se a ela mesma

Prolog - Recursão

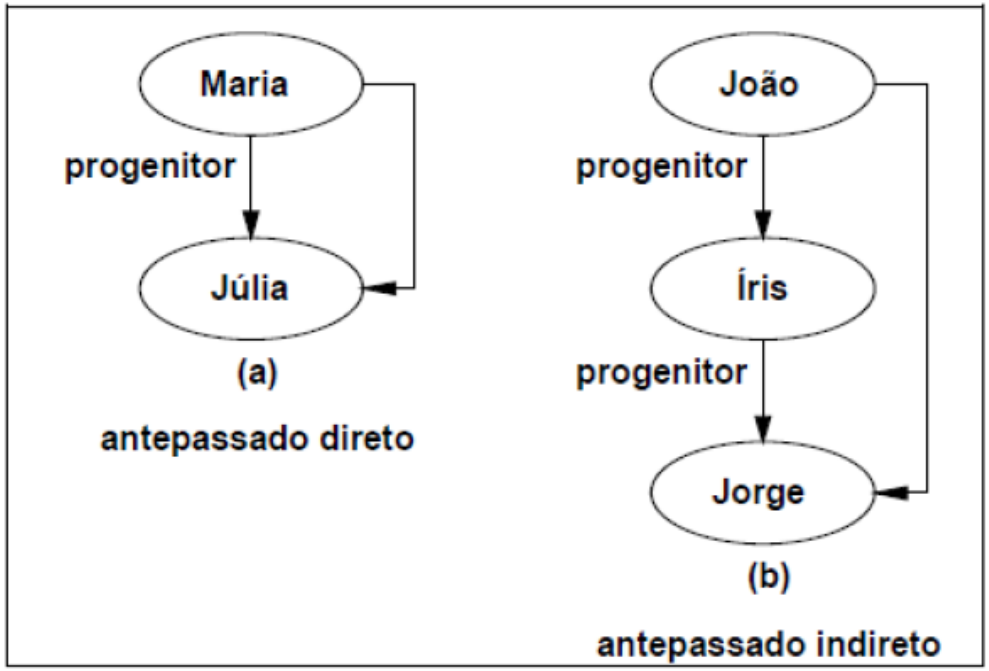


Figura 2.3 Exemplos da relação *antepassado*

Luis, A. M. Palazzo, Introdução à Programação Prolog, Educat, 1997.

Prolog - Recursão

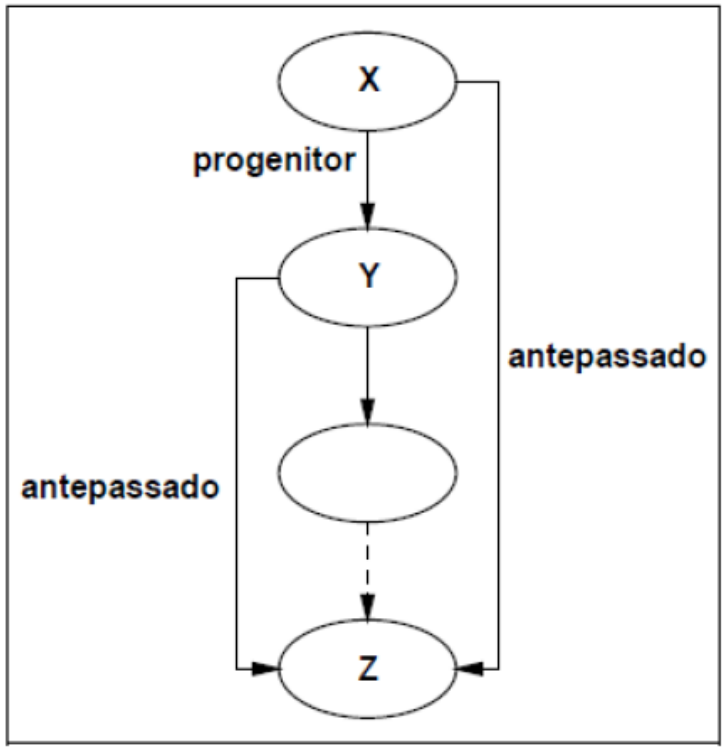


Figura 2.4 Formulação recursiva da relação *antepassado*

Luis, A. M. Palazzo, Introdução à Programação Prolog, Educat, 1997.

Exemplo - Descendente

```
filho(brigite,caroline).
filho(caroline,donna).

descende(X,Y):- filho(X,Y).
descende(X,Y):- filho(X,Z), filho(Z,Y).
```

Exemplo - Descendente

```
filho(ana,brigite).
filho(brigite,caroline).
filho(caroline,donna).
filho(donna,emilia).


descende(X,Y):- filho(X,Y).
descende(X,Y):- filho(X,Z), filho(Z,Y).
```

Exemplo - Descendente

```
filho(ana,brigite).
filho(brigite,caroline).
filho(caroline,donna).
filho(donna,emilia).


descende(X,Y):- filho(X,Y).
descende(X,Y):- filho(X,Z), filho(Z,Y).
```

```
?- descende(ana,donna).
no
?-
```

Exemplo - Descendente

```
filho(ana,brigite).
filho(brigite,caroline).
filho(caroline,donna).
filho(donna,emilia).

descende(X,Y):- filho(X,Y).
descende(X,Y):- filho(X,Z), descende(Z,Y).
```

```
?- descende(A,B).
A=ana,
B=brigite .
```

Exemplo - Descendente

```
filho(ana,brigite).
filho(brigite,caroline).
filho(caroline,donna).
filho(donna,emilia).

descende(X,Y):- filho(X,Z), descende(Z,Y).
descende(X,Y):- filho(X,Y).
```

```
?- descende(A,B).
A=ana,
B=emilia.
```

Exemplo – Fatorial

- Vamos pensar um pouco

0! = 1
1! = 1

2! = 2 * 1!
3! = 3 * 2!
4! = 4 * 3!

Exemplo – Fatorial

- Vamos pensar um pouco

$5! = 5 * 4!$

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

(3 * 2!)

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

(3 * 2!)

(2 * 1!)

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

(3 * 2!)

(2 * 1!)

(1 * 0!)

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

(3 * 2!)

(2 * 1!)

(1 * 0!)

1

Exemplo – Fatorial

- Vamos pensar um pouco

5! = 5 * 4!

(4 * 3!)

(3 * 2!)

(2 * 1!)

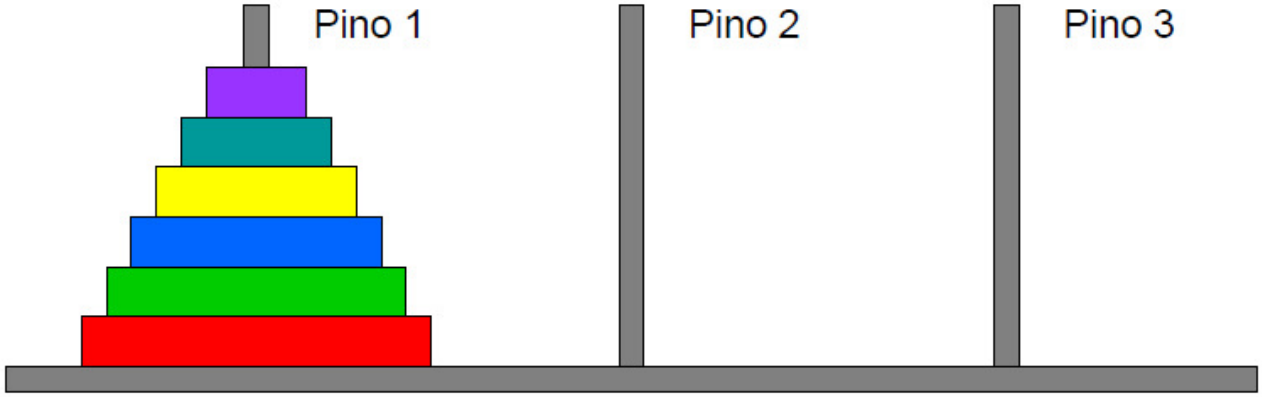
(1 * 0!)

1

Exemplo – Torre de Hanoi

- Em um templo no Extremo Oriente um grupo de sacerdotes tentam mover uma pilha de discos dourados de um pino para outro.
- A pilha inicial possui 64 discos organizados de baixo para cima por tamanho decrescente
- Apenas um disco pode ser movido por vez
- Jamais um disco pode ser colocado sobre outro menor que ele
- Além dos pinos de Origem e de Destino existe um pino de armazenamento Temporário que pode ser usado
- Segundo a lenda quando os sacerdotes terminarem de mover os discos o mundo ira acabar!!!

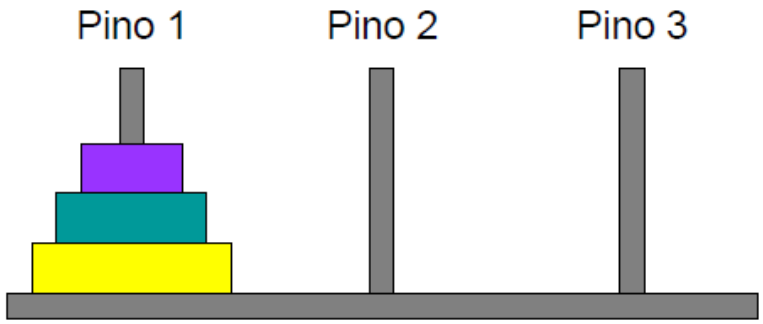
Exemplo – Torre de Hanoi



- Uma solução iterativa para este problema não é tão obvia
- Se abordarmos o problema por meio da recursão, obtemos uma solução simples e elegante para o problema

Mover n discos pode ser visualizado em termos de mover n-1 discos → Recursão

Exemplo – Torre de Hanoi



- Mover n-1 discos do pino 1 para o Pino 2, utilizando o Pino 3 como armazenamento temporário
- Mover o ultimo disco (o maior) do Pino 1 para o Pino 3
- Mover os n-1 discos do Pino 2 para o Pino 3, utilizando o Pino 1 como área de armazenamento temporário

- O algoritmo atinge o caso base quando restar apenas um único Pino para ser movido

Aritmética em Prolog

- Prolog oferece uma série de ferramentas básicas de aritmética.‼
- Tanto para números reais quanto para inteiros.

Aritmética	Prolog
2 + 3 = 5	?- 5 is 2+3.
3 x 4 = 12	?- 12 is 3*4.
5 – 3 = 2	?- 2 is 5-3.
3 – 5 = -2	?- -2 is 3-5.
4 : 2 = 2	?- 2 is 4/2.
1 é o resto da divisão de 7 por 2	?- 1 is mod(7,2).

Aritmética em Prolog

```
?- 10 is 5+5.
true

?- 4 is 2+3.
false

?- X is 3 * 4.
X=12
true

?- R is mod(7,2).
R=1
true
```

Aritmética em Prolog

```
somaTresDepoisDuplica(X, Y):-
    Y is (X+3) * 2.
```

Aritmética em Prolog

```
somaTresDepoisDuplica(X, Y):-
    Y is (X+3) * 2.
```

```
?- somaTresDepoisDuplica(1,X).
X=8.


?- somaTresDepoisDuplica(2,X).
X=10.
```

Aritmética em Prolog

- É importante saber que +, -, / e ∗, na verdade, não realizam operação aritmética alguma.

- 3+2, 4/2, 4-5 são apenas termos comuns do Prolog em uma notação mais amigável:
  3+2 é na verdade **+(3,2)** e assim por diante.

- Expressões tais como 3+2, 4-7 e 5/5 são termos comuns do Prolog

– Functor: +, -, /, ∗
– Aridade: 2
– Argumentos: inteiros

Aritmética em Prolog

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

Aritmética em Prolog

```
?- X = 3 + 2.
X = 3+2
true

?-
```

Aritmética em Prolog

```
?- X = 3 + 2.
X = 3+2
true

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

?-
```

Aritmética em Prolog

- Para forçar o Prolog a realmente avaliar as expressões aritméticas, temos que usar o operador: **is**
- O predicado **is** é um predicado de dois argumentos em Prolog.
- Devido ao fato deste predicado não ser um predicado comum do Prolog, existem algumas restrições em seu uso.

O predicado is

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

O predicado is

```
?- X is 3 + 2.
X = 5.


?-
```

O predicado is

```
?- X is 3 + 2.
X = 5.


?- 3 + 2 is X.
```

O predicado is

```
?- X is 3 + 2.
X = 5


?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?-
```

O predicado is

```
?- X is 3 + 2.
X = 5


?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- Resultado is 2+2+2+2+2.
```

O predicado is

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- Resultado is 2+2+2+2+2.
Resultado = 10.


?-
```

Restrições ao uso de is

- Temos liberdade para usar variáveis no lado direito do predicado **is**.
- Mas, quando o Prolog realmente realizar a avaliação, as variáveis devem estar instanciadas com um termo sem variáveis do Prolog.
- Este termo deve ser uma expressão aritmética.

Aritmética em Prolog

- A aritmética também é envolvida na comparação de valores numéricos
- A verificação se o produto de 277 por 37 é maior que 10000 pode ser especificada pelo objetivo

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

o operador ">" também força a avaliação de expressões

Aritmética em Prolog

- Exemplo
  - Recuperar os nomes das pessoas nascidas entre 1970 e 1980 inclusive usando a relação nasceu:

```
?-nasceu(Nome, Ano), Ano >= 1970, Ano =< 1980.
```

Aritmética em Prolog

- Diferença existente entre o operador de unificação e o operador =:=
  - X = Y → unificação dos objetos X e Y, instanciando, se for o caso, alguma variável em X e Y.
  - X =:= Y → avaliação aritmética sem causar a instanciação de nenhuma variável.

Aritmética em Prolog

- Exemplos

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

True.

?-1+2 = 2+1.

False.

?-1+A = B+2.

A=2,

B=1.

Aritmética em Prolog

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

Luis, A. M. Palazzo, Introdução à Programação Prolog, Educat, 1997.

Aritmética em Prolog

- Recursivamente calcular o fatorial de um número inteiro dado:

```
fatorial(0, 1).
fatorial(X, Y) :-
    X1 is X-1,
    fatorial(X1, Y1),
    Y is X*Y1.
```

Aritmética em Prolog

- Recursivamente calcular a sequência de Fibonacci.

```
fib(0, 0).
fib(1, 1).
fib(X, Y) :-
    X > 1,
    X2 is X – 2, fib(X2, Y2),
    X1 is X – 1, fib(X1, Y1),
    Y is Y1 + Y2.
```

Fim

- Obrigado pela presença!