

DCC024 Linguagens de Programação
2022.1

Introdução a Prolog

Haniel Barbosa



Programação Lógica

- ▶ Paradigma de programação baseado em *resolução de restrições*
- ▶ O usuário escreve as condições que a solução para o problema deve satisfazer
- ▶ A execução do programa consiste em automaticamente derivar uma solução
- ▶ Utilizaremos a linguagem Prolog como base

Programas em Prolog

- ▷ Um programa em Prolog consiste em uma série de *fatos e regras*
- ▷ Esses comandos são escritos utilizando *termos*
- ▷ Um termo é:
 - ▶ um *átomo*
 - um *identificador* começando com letra minúscula. Ex.: atom, x, y.
 - ▶ um *número*
 - um identificador para um número inteiro ou ponto flutuante. Ex.: 123, -5.5
 - ▶ uma *variável*
 - um *identificador* começando com letra maiúscula, ou `_`. Ex.: X, Var, `_`
 - ▶ um *termo composto*
 - um átomo seguindo por termos entre parênteses. Ex.: `x(Y, Z)`, `parent(kim, holly)`
 - Chamamos átomos usados para escrever termos compostos como *predicados*. Como `parent` acima.

Programas em Prolog

- ▷ *Fatos* são termos seguidos por um ponto
- ▷ Um fato estabelece que algo é verdade.
- ▷ Um programas em Prolog pode ser escrito como uma série de fatos.

```
parent(kim, holly).  
parent(margaret, kim).  
parent(margaret, kent).  
parent(ester, margaret).  
parent(herbert, margaret).  
parent(herbert, jean).
```

Rodando programas em Prolog

- ▶ Existem diferentes interpretadores para Prolog. Utilizaremos `swipl`
- ▶ O interpretador carrega comandos em Prolog e lhe permite fazer consultas.
- ▶ Considere de novo os fatos

```
parent(kim, holly).  
parent(margaret, kim).  
parent(margaret, kent).  
parent(ester, margaret).  
parent(herbert, margaret).  
parent(herbert, jean).
```

- ▶ Exemplos de consultas:

```
?- parent(kim, holly).  
?- parent(kim, X).  
?- parent(X, Y).  
?- parent(Y, ester).
```

Rodando programas em Prolog

- ▶ Existem diferentes interpretadores para Prolog. Utilizaremos `swipl`
- ▶ O interpretador carrega comandos em Prolog e lhe permite fazer consultas.
- ▶ Considere de novo os fatos

```
parent(kim, holly).  
parent(margaret, kim).  
parent(margaret, kent).  
parent(esther, margaret).  
parent(herbert, margaret).  
parent(herbert, jean).
```

- ▶ Exemplos de consultas:

```
?- parent(kim, holly).  
?- parent(kim, X).  
?- parent(X, Y).  
?- parent(Y, esther).
```

- ▶ Consultas com variáveis requerem *instanciação* dessas variáveis

Rodando programas em Prolog

- ▷ Diferentes instanciações podem ser obtidas pressionando espaço ou ;

```
?- parent(X, margaret).
```

```
X = esther ;
```

```
X = herbert.
```

- ▷ Consultas podem ser sequências conjuntivas ou disjuntivas de termos, arbitrariamente aninhadas:

```
?- parent(margaret, X), parent(X, holly).
```

```
?- parent(margaret, X); parent(X, holly).
```

```
?- parent(margaret, X), (parent(X, holly); parent(X, jean)).
```

- ▷ Como consultar se `esther` tem um bisneto?

Rodando programas em Prolog

- ▷ Diferentes instanciações podem ser obtidas pressionando espaço ou ;

```
?- parent(X, margaret).
```

```
X = esther ;
```

```
X = herbert.
```

- ▷ Consultas podem ser sequências conjuntivas ou disjuntivas de termos, arbitrariamente aninhadas:

```
?- parent(margaret, X), parent(X, holly).
```

```
?- parent(margaret, X); parent(X, holly).
```

```
?- parent(margaret, X), (parent(X, holly); parent(X, jean)).
```

- ▷ Como consultar se esther tem um bisneto?

```
?- parent(esther, C), parent(C, GC), parent(GC, GGC).
```


Rodando programas em Prolog

- ▷ Diferentes instanciações podem ser obtidas pressionando espaço ou ;

```
?- parent(X, margaret).
```

```
X = esther ;
```

```
X = herbert.
```

- ▷ Consultas podem ser sequências conjuntivas ou disjuntivas de termos, arbitrariamente aninhadas:

```
?- parent(margaret, X), parent(X, holly).
```

```
?- parent(margaret, X); parent(X, holly).
```

```
?- parent(margaret, X), (parent(X, holly); parent(X, jean)).
```

- ▷ Como consultar se esther tem um bisneto?

```
?- parent(esther, C), parent(C, GC), parent(GC, GGC).
```

```
C = margaret,
```

```
GC = kim,
```

```
GGC = holly
```

Programas em Prolog com regras

- ▷ Regras especificam como novos fatos podem ser derivados, dadas certas condições

```
greatGrandparent(GGP, GGC) :- parent(GGP, GP), parent(GP, P), parent(P, GGC).
```

- ▷ O termo à esquerda de :- é verdadeiro se o da direita o for

- ▷ Regras normalmente definem predicados

- ▶ O predicado `greatGrandparent(GGP, GGC)` é verdadeiro quando:
 - GGP tem um filho que tem um filho que tem um filho que é GGC

- ▷ Regras podem ser compostas

```
grandparent(GP, GC) :- parent(GP, P), parent(P, GC).  
greatGrandparent(GGP, GGC) :- grandparent(GGP, P), parent(P, GGC).
```

- ▷ A ordem é irrelevante. As regras acima são equivalentes a

```
greatGrandparent(GGP, GGC) :- grandparent(GGP, P), parent(P, GGC).  
grandparent(GP, GC) :- parent(GP, P), parent(P, GC).
```

Programas em Prolog com regras

- ▶ Regras podem ser definidas recursivamente

```
ancestor(X, Y) :- parent(X, Y).
```

```
ancestor(X, Y) :- parent(Z, Y), ancestor(X, Z).
```

- ▶ Como vocês definiriam um predicado `sibling(X,Y)`?

Programas em Prolog com regras

- ▶ Regras podem ser definidas recursivamente

`ancestor(X, Y) :- parent(X, Y).`

`ancestor(X, Y) :- parent(Z, Y), ancestor(X, Z).`

- ▶ Como vocês definiriam um predicado `sibling(X,Y)`?

`sibling(X,Y) :- parent(P,X), parent(P, Y).`

Programas em Prolog com regras

- ▶ Regras podem ser definidas recursivamente

```
ancestor(X, Y) :- parent(X, Y).
```

```
ancestor(X, Y) :- parent(Z, Y), ancestor(X, Z).
```

- ▶ Como vocês definiriam um predicado sibling(X,Y)?

```
sibling(X,Y) :- parent(P,X), parent(P, Y).
```

- ▶ Note que X e Y podem ser instanciados para o mesmo termo

Programas em Prolog com regras

- ▶ Regras podem ser definidas recursivamente

```
ancestor(X, Y) :- parent(X, Y).
```

```
ancestor(X, Y) :- parent(Z, Y), ancestor(X, Z).
```

- ▶ Como vocês definiriam um predicado sibling(X,Y)?

```
sibling(X,Y) :- parent(P,X), parent(P, Y).
```

- ▶ Note que X e Y podem ser instanciados para o mesmo termo

```
sibling(X,Y) :- parent(P,X), parent(P, Y), not(X=Y).
```

- ▶ Prolog possui alguns predicados pré-definidos. Um deles é a igualdade: =

Listas em Prolog

- ▷ Prolog possui um tipo pré-definido para listas
- ▷ Termos entre colchetes e separados por vírgulas

?- X = [1, haniel, true].

?- [1,2|X] = [1,2,3,4,5].

- ▷ Como definir um predicado `append(X,Y,Z)`?

Listas em Prolog

- ▷ Prolog possui um tipo pré-definido para listas
- ▷ Termos entre colchetes e separados por vírgulas

?- X = [1, haniel, true].

?- [1,2|X] = [1,2,3,4,5].

- ▷ Como definir um predicado `append(X,Y,Z)`?

`append([], L, L).`

`append([Head|TailA], B, [Head|TailC]) :- append(TailA, B, TailC).`

Listas em Prolog

- ▷ Prolog possui um tipo pré-definido para listas
- ▷ Termos entre colchetes e separados por vírgulas

?- X = [1, haniel, true].

?- [1,2|X] = [1,2,3,4,5].

- ▷ Como definir um predicado `append(X,Y,Z)`?

`append([], L, L).`

`append([Head|TailA], B, [Head|TailC]) :- append(TailA, B, TailC).`

- ▷ Como definir um predicado `member(X,Y)`?

Listas em Prolog

- ▷ Prolog possui um tipo pré-definido para listas
- ▷ Termos entre colchetes e separados por vírgulas

?- X = [1, haniel, true].

?- [1,2|X] = [1,2,3,4,5].

- ▷ Como definir um predicado `append(X,Y,Z)`?

`append([], L, L).`

`append([Head|TailA], B, [Head|TailC]) :- append(TailA, B, TailC).`

- ▷ Como definir um predicado `member(X,Y)`?

`member(X, [X|_]).`

`member(X, [_|Y]) :- member(X, Y).`

Listas em Prolog

- ▷ Prolog possui um tipo pré-definido para listas
- ▷ Termos entre colchetes e separados por vírgulas

?- X = [1, haniel, true].

?- [1,2|X] = [1,2,3,4,5].

- ▷ Como definir um predicado `append(X,Y,Z)`?

`append([], L, L).`

`append([Head|TailA], B, [Head|TailC]) :- append(TailA, B, TailC).`

- ▷ Como definir um predicado `member(X,Y)`?

`member(X, [X|_]).`

`member(X, [_|Y]) :- member(X, Y).`

- ▷ Como vocês definiriam um predicado `permutation(X,Y)`?