

Programação Lógica

Jomi F. Hübner

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas
<http://www.das.ufsc.br/~jomi/das6609>



PGEAS 2012/1

Motivações

- adequação da linguagem ao problema
- programação declarativa
- o programador estabelece a teoria
 - o interpretador é um provador automático de teoremas

Conteúdo

- princípios & motivação
- cláusulas de Horn
- linguagem Prolog
- unificação & inferência & backtracking
- exemplos

Declarativo vs Imperativo

Java

```
boolean member(int e, List<Integer> l) {  
    for (int i: l)  
        if (i == e)  
            return true;  
    return false;  
}
```

Prolog

```
member(E,[E|_]).  
member(E,[_|T]) :- member(E,T).
```

Alto nível de abstração

% fatos (relação entre objetos)

irmao(bob,alice).

irmao(alice,tom).

irmao(tom,ze).

Consulta
?- irmao(bob,ze).
yes

% propriedades da relação

$\forall x,y,z \text{ irmao}(x,y) \wedge \text{irmao}(y,z) \Rightarrow \text{irmao}(x,z)$

irmao(X,Z) :- irmao(X,Y), irmao(Y,Z).

Consulta
?- irmao(bob,ana).
ERROR: Out of local stack

5

Ideal x Real

- lógica proposicional não é muito expressiva
- lógica de predicados precisa de métodos eficientes de prova
- solução
 - limitar a capacidade de expressão da linguagem
 - usar resolução SLD

6

Cláusulas de Horn

- cláusula
 - $P \vee Q \vee \neg R \vee \neg S$
- cláusulas de Horn tem só um literal positivo
 - $Q \vee \neg R \vee \neg S \equiv R \wedge S \Rightarrow Q$

7

SLD-Resolution

(Selective Linear Definite clause resolution)

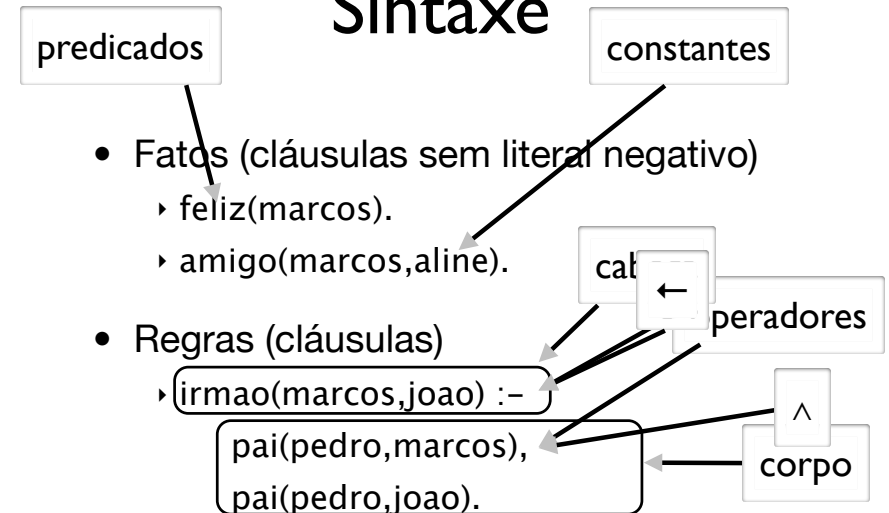
- fórmulas são cláusulas de Horn
- inicia refutando
- resolução entre
 - o último resolvente (linear) e
 - o literal positivo de uma das cláusulas (selected)
- correto e completo

1	P	
2	$R \vee \neg P$	
3	$\neg R \vee S$	
4	$\neg S$	
5	$\neg R$	R, 4, 3
6	$\neg P$	R, 5, 2
7	\perp	R, 6, 1
8	S	

8

Prolog

Sintaxe



10

Variáveis

- Regras (cláusulas)
 - `irmao(X,Y) :-`
 - `pai(Z,X),`
 - `pai(Z,Y).`

Unificação

- verificação de “igualdade” entre termos
 - sendo s e t termos e θ uma substituição
 - se $s\theta = t\theta$, então θ é uma unificação para s e t
- identifica qual o valor das variáveis para que dois predicados sejam iguais
 - $p(a) = p(X)$
 - $p(1,b(u,o),a) = p(1,X,Y)$
 - $p(A,2) = p(3,B)$
 - $p(X,g(Y)) = p(a,g(X))$
- mecanismo de atribuição de valores

$X=a, g(Y)=g(X)$
 $X=a, Y=X$
 $X=a, Y=a$

Unificador mais geral

- $p(X,Y) = p(g(Z),Z)$
 - $\theta = \{ X/g(Z), Y/Z \}$
 - $\theta' = \{ X/g(a), Y/a, Z/a \}$
 - θ é mais geral que θ'
- uma substituição θ é mais geral que θ' se houver uma substituição w tal que
 - $\theta' = \theta w$ $w = \{ Z/a \}$ no exemplo
- o unificador mais geral (mgu) é a substituição mais geral para dois termos

13

Inferência & Backtracking

chefe(a).	
chefe(b).	Consulta
chefe(c).	?- chefe_legal(b).
legal(X) :- p1(X).	y Consulta
legal(X) :- p2(X).	?- chefe_legal(a).
p1(b).	no
p2(c).	Consulta
chefe_legal(X) :-	?- chefe_legal(X).
chefe(X),	X=b
legal(X).	

14

Recursão

	Consulta
% fatos (relação entre objetos)	?- irmao(bob,chico).
irmao(bob,alice).	ERROR: Out of local
irmao(alice,tom).	stack
irmao(tom,ze).	yes

% propriedades da relação
 $\forall x,y,z \text{ irmao}(x,y) \wedge \text{irmao}(y,z) \Rightarrow \text{irmao}(x,z)$
 irmao(X,Z) :- irmao(X,Y), irmao(Y,Z).

15

Remoção da recursão a Esquerda

% fatos (relação entre objetos)
 eh_irmao(bob,alice).
 eh_irmao(alice,tom).
 eh_irmao(tom,ze).

% propriedades da relação
 $\forall x,y,z \text{ irmao}(x,y) \wedge \text{irmao}(y,z) \Rightarrow \text{irmao}(x,z)$
 irmao(X,Y) :- eh_irmao(X,Y).
 irmao(X,Z) :- eh_irmao(X,Y), irmao(Y,Z).

16

SAT solver

- Verificar se uma fórmula com cinco proposições é satisfatível

17

SAT solver

Java

```
boolean sat(Formula f) {  
    for (boolean A: { true, false})  
        for (boolean B: { true, false})  
            for (boolean C: { true, false})  
                for (boolean D: { true, false})  
                    for (boolean E: { true, false})  
                        if (satisfaz(A,B,C,D,E,f))  
                            return true;  
    return false;  
}
```

18

SAT solver

Prolog

```
bool(true). bool(false).  
satisfaz(A,B,C,D,E) :-
```

```
%  $(A \vee B \vee \neg C) \wedge (A \vee \neg C) \wedge \dots$ 
```

```
satisfaz(A,B,C,D,E) :-  
    c1(A,B,C), c2(A,C), ...
```

```
c1(true,_,_). c1(_,true,_). c1(_,_,false).
```

```
c2(true,_) c2(_,false).
```

```
...
```

Consulta

```
?- bool(A), bool(B),  
    bool(C), bool(D),  
    bool(E),
```

Exercícios

19

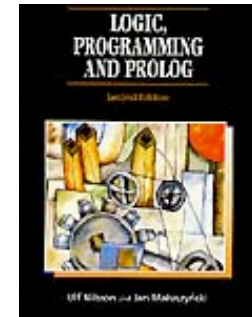
Histórico

- 1970: interpretador (Edinburgh)
- Fifth Generation Computer (Japan)
- Hoje temos interpretadores
 - comerciais (Quintus Prolog)
 - open source (SWI-Prolog, g-Prolog)
- Extensões, pesquisas, ...
 - YAP-Prolog (high-performance, ...)
 - Qu-Prolog (quantifiers, concurrent programming, ...)
 - ECLiPSe (constraint programming, ...)

21

Bibliografia

- Logic, Programming and Prolog (2ed) by Ulf Nilsson and Jan Maluszynski
- download em
<http://www.ida.liu.se/~ulfni/lpp>



22

Bibliografia



23