Paradigma de Programação Lógico

2 - Fundamentos - Exercícios

Atenção: Nos exercícios que pedem a resposta do Prolog a uma dada consulta, é interessante fazer o exercício simulando a execução antes de testar com o Prolog. Fazendo isto você exercitará o entendimento do funcionamento do Prolog.

2.1) [pip 1.2] Dado a seguinte base de conhecimento

```
homem(albert).
homem(edward).

mulher(alice).
mulher(victoria).

% o primeiro argumento é o(a) filho(a),
% o segundo argumento a mãe e o terceiro o pai
pais(edward, victoria, albert).
pais(alice, victoria, albert).

irma_de(X, Y) :-
    mulher(X),
    pais(X, M, F),
    pais(Y, M, F).
```

como o Prolog responde a consulta `irma_de(alice, X).`

2.2) [lpn 1.5] Dado a seguinte base de conhecimento

```
bruxo(rony).
bruxo(X) :- temVassoura(X), temVarinha(X).
temVarinha(harry).
jogadorQuadribol(harry).
temVassoura(X) :- jogadorQuadribol(X).
```

como o Prolog responde as seguintes consultas?

```
bruxo(rony).
bruxa(rony).
bruxo(hermione).
bruxa(hermione).
bruxo(harry).
bruxo(Y).
bruxa(Y).
```

2.3) [lpn 1.5] Dado a seguinte base de conhecimento

```
likes(sam,Food) :- indian(Food), mild(Food).
likes(sam,Food) :- chinese(Food).
likes(sam,Food) :- italian(Food).
likes(sam,chips).

indian(curry).
indian(dahl).
indian(tandoori).
indian(kurma).

mild(dahl).
```

```
mild(tandoori).
mild(kurma).

chinese(chow_mein).
chinese(chop_suey).
chinese(sweet_and_sour).

italian(pizza).
italian(spaghetti).
```

como o Prolog responde as seguintes consultas?

```
likes(sam,dahl).
likes(sam,chop_suey).
likes(sam,pizza).
likes(sam,chips).
likes(sam,curry).
```

2.4) [lpn 2.1] Quais dos seguintes par de termos unificam? Quando for o caso, dê o valor instanciado para cada variável que levou a unificação.

a. `bread = bread`
b. `'Bread' = bread`
c. `'bread' = bread`
d. `Bread = bread`
e. `bread = sausage`
f. `food(bread) = bread`
g. `food(bread) = X`
h. `food(X) = food(bread)`
i. `food(bread, X) = food(Y, sausage)`
j. `food(bread, X, beer) = food(Y, sausage, X)`
k. `food(bread, X, beer) = food(Y, kahuna_burger)`
l. `food(X) = X`
m. `meal(food(bread), drink(beer)) = meal(X,Y)`
n. `meal(food(bread), X) = meal(X,drink(beer))`

2.5) [lpn 1.3] Quantos fatos, regras, cláusulas e predicados existem na seguinte base de conhecimento? Qual é a cabeça de cada regra e quais são as metas que elas contêm?

```
mulher(vincent).
mulher(mia).
homem(jules).
pessoa(X) :- homem(X); mulher(X).
ama(X, Y) :- pai(X, Y).
pai(Y, Z) :- homem(Y), filho(Z, Y).
pai(Y, Z) :- homem(Y), filha(Z, Y).
```

2.6) Dado a seguinte base de conhecimento

```
pai(adao, abel).
pai(adao, caim).
pai(adao, sete).
pai(caim, enoque).
pai(enoque, irad).
pai(irad, meujael).
pai(meujael, metusael).
pai(metusael, lameque).
pai(lameque, jabal).
pai(lameque, jubal).
```

```
pai(lameque, tubalcaim).
pai(lameque, naama).

mae(eva, abel).
mae(eva, caim).
mae(eva, sete).
mae(ada, jabal).
mae(ada, jubal).
mae(zila, tubalcaim).
mae(zila, naama).

homem(sete).
homem(caim).
homem(jabal).
homem(jubal).
homem(tubalcaim).

mulher(naama).
```

defina (e teste) as seguintes regras em Prolog:

a. X é homem se X é pai.
b. X é mulher se X é mãe.
c. X e Y são irmãos se X e Y têm o mesmo pai ou mesma mãe.
d. X e Y são casados se eles têm um filho.
e. `avo(X, Y)`, X é avó de Y.
f. `irma(X, Y)`, X é irmã Y.
g. `irmao(X, Y)`, X é irmão Y.
h. `e_pai(X)`, X é pai.
i. `e_mae(X)`, X é mãe.
j. `e_filho(X)`, X é filho.
k. `ancestral(X, Y)`, X é ancestral de Y.

2.7) Defina um predicado `triangulo(A, B, C, T)` que é verdadeiro se `A`, `B` e `C` formam um triângulo do tipo `T`, onde `T` pode ser `equilatero` (todos os lados iguais), `isosceles` (pelo menos dois lados iguais) ou `equilatero` (todos os lados diferentes). Exemplo

```
?- triangulo(3, 3, 3, T).
T = equilatero.

?- triangulo(2, 3, 3, isosceles).
true.

?- triangulo(2, 3, 3, escaleno).
false.
```

2.8) Defina um predicado `area(F, A)` que é verdadeiro se a área da figura `F` é `A`. Considere que `F` deve ser uma estrutura que representa uma figura. Exemplo

```
?- area(circulo(2), A).
A = 12.56.

?- area(quadrado(4), A).
A = 16.

?- area(retangulo(3, 4), 12).
true.
```

2.9) Defina uma predicado `soma_quadrado_maiores(A, B, C, S)` que é verdadeiro se `S` é a soma dos quadrados dos dois maiores números entre `A`, `B` e `C`. Exemplo

```
?- soma_quadrado_maiores(3, 4, 5, S).
S = 41.
```

2.10) [pp99 2.01] Defina um predicado `primo(X)` que é verdadeiro se `X` é um número primo. Exemplo

```
?- primo(7).
true.

?- primo(4).
false.
```

2.11) Defina um predicado `quantidade_primos(I, F, Q)` que é verdadeiro se `Q` é a quantidade de números primos entre `I` e `F`.

```
?- quantidade_primos(2, 10, Q).
Q = 4.
```

2.12) Defina um predicado `perfeito(X)` que é verdadeiro se `X` é um número perfeito. Um número é perfeito se a soma dos seu divisores próprios é igual a ela. Por exemplo, o número 6 é perfeito, pois $6 = 1 + 2 + 3$. O número 28 também é perfeito, pois $28 = 1 + 2 + 4 + 7 + 14$.

```
?- perfeito(6).
true.

?- perfeito(9).
false.
```

Referências

- [pp99]. 99 problemas para resolver em (Prolog)
- [lpn]. Lear Prolog Now
- [pip]. Programming in Prolog.

Licença