Логическое программирование

PROLOG –

ПРОграммирование

ЛОгики

Стили

- Императивный
  - модель в некоторой формальной системе,
  - решение на императивном языке программирования в терминах компьютной системы
- Декларативный
  - человек лишь описывает решаемую задачу,
  - а поиском решения занимается декларативная система программирования

  Высокая скорость разработки приложений, меньший размер исходного кода, легкость записи знаний на декларативных языках, более понятные, по сравнению с императивными языками, программы.

Декларативная программа

- Программа на Прологе не задает последовательность действий, а лишь описывает имеющуюся информацию и задает вопрос.
- Интерпретатор Пролога, делая логические выводы из имеющейся информации, ищет ответ на этот вопрос, или в терминах Пролога **доказывает цель**

PROLOG программа

- Информация, составляющая Пролог-программу, представляется в виде **фактов** и **правил**, которые вместе образуют **базу данных**.
- **Вопрос**, заданный к базе данных, называется **целью**.

Факты

- **Факт** – это то, что известно наверняка, т.е. заведомо правильно
- Например:

```
студент(иванов).  ; Иванов является
  студентом
нравится(коля,лена).  ; Коле нравится Лена
дарить(витя,лена,цветы).  ; Витя подарил
                            Лене цветы.
сотрудник(петров,30).  ; Петров является
     сотрудником фирмы, его возраст - 30 лет.
```

Предикаты

- Предикат имеет вид: *p(a1,a2,...,an)*, где p имя предиката, а a1,a2,...,an - аргументами предиката.
- Аргументами могут быть:
  - константы (числа, строки),
  - символьные атомы (идентификаторы, начинающиеся с маленькой буквы),
  - переменные (идентификаторы, начинающиеся с заглавной буквы или символа подчеркивания),
  - списки,
  - структуры.

Вопросы

Пусть заданы факты

```
студент(иванов).
нравится(коля,лена).
```

Зададим вопросы

```
?-студент(иванов). ;     верно ли, что Иванов -студент?
ДА                 ;     ответ Пролога

?-студент(сидоров).;   верно ли, что Сидоров -студент?
НЕТ

?-нравится(коля,лена). ; верно ли, что Коле нравится
  Лена?
ДА
```

Доказательство цели

- Сам вопрос в Прологе называется **целевым утверждением** или **целью**.
- Процесс поиска ответа называется **согласованием целевого утверждения**, или **доказательством цели**.

Переменные

Вопросы из предыдущего примера требуют ответа да/нет. Пусть нам требуется узнать подробности.

```
студент(иванов).
нравится(коля,лена).
нравится(иванов,лена).
?-студент(Кто). ; Кто является студентом?
Кто=иванов ;
НЕТ
?-нравится(Кому, лена). ; Кому нравится
Лена?
Кому=коля ;
Кому=иванов ;
НЕТ
?-нравится(Кому,Кто). ; Кто кому нравится?
Кому=коля, Кто=лена ;
Кому=иванов, Кто=лена ;
НЕТ
```

Согласование переменных

Синтаксически переменные определяются как идентификаторы, начинающиеся с заглавной буквы. Кроме этого, переменными являются идентификаторы, начинающиеся с символа подчеркивания, например, "_кто", "_х" и т.д.

При согласовании аргументов, являющихся переменными, возможно несколько случаев:

- Переменные величины могут иметь значение, в этом случае, они называются **связанными**. Переменные, которым не присвоено никакое значение, называются **свободными**.
- Перед началом согласования цели и утверждения базы все переменные свободны.
- Если свободная переменная согласовывается с числом, символьным атомом или структурой, то этой переменной присваивается соответствующее значение, и она при дальнейшем согласовании становится связанной.
- При согласовании связанной переменной она заменяется своим значением.

- Если одна переменная свободная, а другая связанная, то свободной переменной присваивается значение связанной.
- При согласовании связанной переменной с константой выполняется проверка на равенство. Таким образом, согласование переменных объединяет вместе как операцию присваивания, так и операцию сравнения.
- Если при согласовании обе переменные были свободными, то они успешно согласуются, оставаясь при этом свободными, но становятся **сцепленными**. Если при дальнейшем согласовании одна из сцепленных переменных получит какое-либо значение, то вторая сцепленная переменная получит такое же значение.
- Рассмотренные варианты исчерпывают возможные случаи согласования предикатов

Анонимные переменные

- Переменная величина, обозначаемая `X=_`, называется **анонимной** и имеет при согласовании две особенности:
- анонимная переменная согласуется с любым аргументом, оставаясь при этом свободной;
- значения анонимных переменных не выводятся в качестве результата вопроса.

Пример

```
родители(мария,николай,анна).
        ; Мария и Николай являются
            родителями Анны.
родители(ольга,сергей,дмитрий)
        ; Ольга и Сергей – родители
                Дмитрия.
?-родители(_,Отец,_).
Отец=николай ;
Отец=сергей ;
НЕТ
```

С другой стороны

```
?-родители(X,Отец,Y).
```

является корректным вопрососм, но приводит к выдаче значений переменных X и Y, что не является необходимым.

```
?-родители(X,Отец,Y).
X=мария,Отец=николай,Y=анна ;
X=ольга,Отец=сергей,Y=дмитрий ;
НЕТ
```

Правила (задают условия вида ЕСЛИ… ТО…)

```
нравится(лена,витя).
нравится(андрей,музыка).
нравится(лена,Человек) :-
            нравится(Человек,музыка).
```

Зададим вопрос:

```
?-нравится(лена,Кто).
Кто=витя ;
Кто=андрей ;
НЕТ
```

Пример

```
женщина(анна).
женщина(елена).
отец(алексей,елена).
мать(ольга,елена).
родители(алексей,ольга,анна).
родители(X,Y,Z) :-мать(X,Z),отец(Y,Z).
сестра(X,Y) :-
  женщина(X),родители(M,F,X),родители(M,F,Y).
```

Зададим теперь вопрос: Кто кому является сестрами?

```
?-сестра(Кто,Кому).
Кто=анна, Кому=анна ;
Кто=анна, Кому=елена ;
НЕТ
```

Исправим

```
сестра(X,Y) :-женщина(X),
    родители(M,F,X),
    родители(M,F,Y),
    различны(X,Y).
```

Предикат «дедушка»

```
дедушка(X,Y) :-
    отец(X,Z),отец(Z,Y).
дедушка(X,Y) :-
    отец(X,Z),мать(Z,Y).
дедушка(X,Y) :-
    отец(X,Z),родитель(Z,Y).
```

Пример рекурсивного предиката «Предок»

```
предок(X,Y) :-родитель(X,Y).
предок(X,Y) :-родитель(X,Z),
    предок(Z,Y).
```

Пример рекурсивного предиката FOR

```
For(X,_,X).
For(I,M,X):-I1=I+1,I1<M,
  for(I1,M,X).
```

Структура программы на PROLOGe

```
DOMAINS
  person=symbol
PREDICATES
  like(person,person)

CLAUSES
  like(alex,olga).
  like(olga,alex).
  like(victor,music).
  like(olga,X) :- like(X,music).

GOAL
  like(olga,Who).
```

СПИСКИ

PROLOG	LISP
[1,2,3]	(1 2 3)
[[a]]	((a))
[[a],b,[c,d]]	((a) b (c d))
[]	() или nil.

Объявление списков

```
DOMAINS
   list=integer*
   list1=list*
PREDICATES
   p(list1).
```

Пример корректного предиката P:

```
p([[1,2], [3], [4,5]]).
```

Разбиение списка на голову и хвост

Конструкция [X|Y], где X – голова списка, Y – хвост списка.

Пример:

```
p([[1,2],3,[4]]).
?-p([X|Y]).
X=[1,2], Y=[3,[4]];
НЕТ
```

Предикат – аналог CONS позволяет как собирать список из головы и хвоста, так и разбивать список на голову и хвост.

```
cons(X, Y, [X|Y]).
?- cons([1], [2,3], Z).
Z=[[1],[2,3]];
НЕТ
?-cons(X,Y,[1,2,3]).
X=1, Y=[2,3];
НЕТ
```

Сумма элементов одноуровневого числового списка

```
sum([],0).
sum([X1|X2,Y) :- sum(X2,Y1), Y
  is Y1+X1.
```

Проверим, как работает наш предикат:

```
?-sum([1,2,4],X).
X=7
```

Элемент X содержится в списке Y

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

Проверим работоспособность написанного предиката:

```
?-member(a,[b,c,a,z]).
ДА
?-member(x, [b,c,a,z]).
НЕТ
```

И в качестве аргумента может быть переменная

```
?-member(X,[b,c,a,z]).
   X=b;
   X=c;
   X=a;
   X=z;
```

Построить предикат append(X,Y,Z), соединяющий два списка вместе

```
append([],X,X).
append([X|Y],Z,[X|P]):-
                        append(Y,Z,P).

?-append([1,2],[3,4,5],Z).
Z=[1,2,3,4,5]
```

Работает во все стороны

```
?- append([1,2,3],X,[1,2,3,4,5]).
X=[4,5]
?- append(X,[3,4,5],[1,2,3,4,5]).
X=[1,2]
```

И даже из каких подсписков состоит список

```
?- append(X,Y,[1,2,3,4]).
X=[], Y=[1,2,3,4];
X=[1], Y=[2,3,4];
X=[1,2], Y=[3,4];
X=[1,2,3], Y=[4];
X=[1,2,3,4], Y=[];
```

Определить предикат memq(X,Y), проверяющий, содержится ли выражение X в списке Y.

```
memq(X,X).
memq(X,[Y|Z]) :- memq(X,Y).
memq(X,[Y|Z]) :- memq(X,Z).
```

Проверка работоспособности:

```
?- memq(a,[s,z,[a,b],x]).
```

ДА

```
?-memq(a,[s,z,[y,b],x]).
```

НЕТ