Московский авиационный институт
(государственный технический университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа №2

по курсу
«Логическое программирование»

Выполнил: Баскаков О.А.
Группа: 08-306
№ по списку: 2
Руководитель: Левинская М.А.
Оценка:
Дата:

Москва
2011 г.

Задание

Написать и отладить Пролог-программу решения логической задачи, проанализировать эффективность, безопасность и непротиворечивость решения.
Вариант №3:
Как-то раз случай свел в купе известного астронома, поэта, прозаика и драматурга. Это были Алексеев, Борисов, Константинов и Дмитриев. Оказалось, что каждый из них взял с собой книгу, написанную одним из пассажиров этого купе. Алексеев и Борисов углубились в чтение, предварительно обменявшись купленными книгами. Поэт читал пьесу (т.е. драму). Прозаик, очень молодой человек, выпустивший свою первую книгу, говорил, что он никогда ничего не читает по астрономии. Борисов купил в дорогу одно из произведений Дмитриева. Никто из пассажиров не покупал и не читал книги, написанные им самим. Что читал каждый из них? Кто кем был?

Реализация

Задача решается подбором трех списков, задающих состояние (Sw,Sr,Sh), которые показывают, кем что написано, прочитано, и какую книгу пассажир имел с собой.
Решение задачи состоит в проверке условий предикатами:

```
Written (Who, What, Solution);
Reading (Who, What, Solution);
Has     (Who, What, Solution).
```

Соответствие условиям гарантируют функции test(Sw,Sr,Sh) и test2(A,Sw,Sr,Sh).
Логику программы выражает последовательность условий:

```
test(Sw,Sr,Sh) :-
    not(written(dmitriev, proza,Sw)),
    reading(alexeev, X1,Sr), has(borisov, X1,Sh),
    reading(borisov, X2,Sr), has(alexeev, X2,Sh),
    has(borisov, X3,Sh), written(dmitriev,X3,Sw),
    written(X4, poet,Sw), reading(X4, drama,Sr),

    written(X5, proza,Sw),
    not(reading(X5, astronom,Sr)),
    not(has(X5, astronom,Sh)).
    #~ Для каждого условия своя переменная X1..X5
```

Для проверки того что никто не прочитал и не покупал свою книгу:

```
test2(A,Sw,Sr,Sh) :-
    written(A, X,Sw), not(reading(A,X,Sr)), not(has(A,X,Sh)).
```

Так как позиция в списке соответствует конкретному человеку:

```
people(X):-
     X = [alexeev, borisov, konstan, dmitriev].

zzz(alexeev, X, [X,_,_,_]).
zzz(borisov, X, [_,X,_,_]).
zzz(konstan, X, [_,_,X,_]).
zzz(dmitriev, X, [_,_,_,X]).
```

Перебор возможных решений определим предикатом permute:

```
permute([],[]).
permute(L, [H|T]):- remove(H,L,L1), permute(L1,T).

books(Y):- Y = [astronom, poet, proza, drama].

solList(X):-
     books(Y),
     permute(Y , X).
```

Поиск решения на swi-prolog:

```
#! /usr/bin/prolog -s !#

findSol(X) :-
      people(Y),
      solList(Sw),
      solList(Sr),
      solList(Sh),

      test(Sw,Sr,Sh),
      test2(alexeev,Sw,Sr,Sh),
      test2(borisov,Sw,Sr,Sh),
      test2(dmitriev,Sw,Sr,Sh),
      test2(konstan,Sw,Sr,Sh),

      write('HERE answer:\n'),
      write(Y),
      write(' who\n'),
      write(Sw),
      write(' write\n'),
      write(Sr),
      write(' read\n'),
      write(Sh),
      write(' have\n'),
      X = (Y,Sw,Sr,Sh).
```

Тестирование:

```
oleg@debian:~/LP$ ./2.pro
% /home/oleg/LP/2.pro compiled 0.01 sec, 7,420 bytes
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.10.1)

?- findSol(_).
HERE answer:
[alexeev,borisov,konstan,dmitriev] who
[poet,astronom,proza,drama] write
[drama,proza,poet,astronom] read
[proza,drama,poet,astronom] have
true ;
false.
?-
% halt
```

Заметим, результатом являются 1 решение.

Сравним с оптимизированной программой на языке Python:

Логика примет вид:

```
def test( pp, ww, rr, hh):
      for i in range(len(pp)):
            if( ww[i]=='proza')and(not(rr[i]!='astronom')):return False
            if( ww[i]=='proza')and(not(hh[i]!='astronom')):return False
            if( ww[i]=='poet')and(not(rr[i]=='drama')):return False
      #~ GSOM 2 lvl
      index = dict( zip( pp, range(len(pp)) ) )
      if(not(rr[index['alexeev'] ]==hh[index['borisov'] ])): return False
      if(not(hh[index['alexeev'] ]==rr[index['borisov'] ])): return False
      if(not(hh[index['borisov'] ]==ww[index['dmitriev'] ])): return False
      if(ww[index['dmitriev']] == 'proza' ): return False
      return True
```

Сопоставление с образцом заменяется индексацией структурой map:
```
index = dict( zip( pp, range(len(pp)) ) )
```
Следствие A=>B на If(A and not(B)).

Протестируем…
```
oleg@debian:~/LP$ python3.1 prolog2.py
['alexeev', 'borisov', 'konstan', 'dmitriev']
['poet', 'astronom', 'proza', 'drama']
['drama', 'proza', 'poet', 'astronom']
['proza', 'drama', 'poet', 'astronom']

FINISH in 1458 iterations.

1 is name
2 is who
3 is read
4 is have
```

Оптимизация дала 1458 итераций против 13842 = (4!^3) на прологе.
Подобного результата можно достичь механизмом отсечений.

Замечания

Можно попытаться удовлетворить условиям задачи, перебирая только 1 параметр, но долгие часы отладки показали, что при этом либо теряются все решения, либо программа зацикливается. Достаточно перебора в 2х плоскостях, если полностью выразить 1 предикат через остальные.
Эффективность – если использовать отсечения дерева решений.
Непротиворечивость – проверка всех условий.
Безопасность – вследствие полного перебора на ограниченном множестве.
*подтверждение ответа программой на Python.

Выводы

```
[alexeev,  borisov,   konstan,   dmitriev]  who
[poet,     astronom,  proza,     drama]     write
[drama,    proza,     poet,      astronom]  read
[proza,    drama,     poet,      astronom]  have
```

Т.е. Алексеев был поэтом, читал пьесу, и купил прозу в дорогу …
Конкретное решение выполнено не совсем в декларативном стиле, т.к. полный перебор легко выполнить на большинстве языков. В свою очередь пролог позволяет легко задавать логику приложения в естественном виде (предикатами). Механизм Pattern matching дает возможность сделать код более наглядным. Он используется для эффективной обработки алгебраических типов данных.