**Дубинин В.Н.**

**Программирование на языке логического программирования SWI Prolog**

***Методические указания по выполнению лабораторных работ***

**Содержание**

Лабораторная работа N 1. Основы работы в системе SWI Prolog

Лабораторная работа N 2. Работа с отношениями на языке Пролог

Лабораторная работа N 3. Работа со списками на языке Пролог

Лабораторная работа N 4.Работа с базами данных на языке Пролог

Лабораторная работа N 5. Трансформация формул

**Лабораторная работа N 1**

**ОСНОВЫ РАБОТЫ В СИСТЕМЕ SWI Prolog**

ЦЕЛЬ РАБОТЫ: получение основных навыков работы в системе SWI Prolog, изучение технологии подготовки и отладки Пролог-программ.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Запустить систему SWI Prolog.

2. Ввести бинарное отношение РОДИТЕЛЬ (англ.:parent) и

представить его в виде Пролог-программы. При этом запись

parent(tom,bob) означает, что tom является родителем bob'a. Симво-

лы tom и bob пишутся со строчной буквы, так как являются констан-

тами языка Пролог. Переменные языка Пролог начинаются с заглавных

букв.

Отношение РОДИТЕЛЬ представим следующей Пролог-программой,

включающей только одни факты:

parent(pame,bob).

parent(tom,bob).

parent(tom,lize).

parent(bob,anna).

parent(bob,pate).

parent(pate,jhim).

Следует отметить, что после каждого предложения языка Пролог,

в том числе после фактов, обязательно ставится точка.

3. Войти в текстовый редактор (например, Notepad)

и набрать представленную в п. 2 Пролог-программу, записав ее с расширением .pl.

Загрузить этот файл в Пролог-систему (пункт меню Consult системы SWI Prolog).

4. Составить и задать набранной программе вопросы, наиболее

полно отражающие возможности языка Пролог для данного примера, за-

фиксировать ответы Пролог-системы на каждый вопрос.

Ниже перечислены вопросы, их кодировка на языке Пролог, отве-

ты Пролог-системы и их интерпретация.

ВОПРОС 1: "Является ли Bob родителем Pate ?".

?-parent(bob,pate).

yes

Цель достигнута, успех.

ВОПРОС 2: "Является ли Lize родителем Pate ?"

?-parent(lize,pate).

no

Цель недостижима, неудача.

ВОПРОС 3: "Является ли Tom родителем Ben ?"

?-parent(tom,ben).

no

Цель недостижима, неудача, так как Ben не упоминается.

ВОПРОС 4: "Кто является родителем Lize ?"

?-parent(X,lize).

X=tom

Цель достижима при X=tom, родителем Lize является Tom.

ВОПРОС 5: "Кто дети Bob'a ?"

?-parent(bob,X).

X=anna;

X=pate;

no

Цель достижима при X=anna, X=pate, дети Bob'a - Anna, Pate.

Пролог-система выдает несколько ответов. К поиску нескольких отве-

тов ее побуждает знак ";" вводимый пользователем после получения

очередного ответа.

ВОПРОС 6: "Кто чей родитель ?"

?-parent(X,Y).

X=pame

Y=bob ;

X=tom

Y=bob

...

Цель достижима при X=pame и Y=bob, при X=tom и Y=bob и т.д.

Pame является родителем Tom'a, Tom является родителем Bob'a и т.д.

ВОПРОС 7: "Кто является родителем родителя Jhim'a ?"

?-parent(Y,jhim),parent(X,Y).

X=bob

Y=pate

Конъюнкция целей достижима при X=bob и Y=pate, родителем родителя

Jhim'a является Bob.

ВОПРОС 8: "Кто внуки Tom'a ?"

?-parent(tom,X),parent(X,Y).

X=bob

Y=anna ;

X=bob

Y=pate

Конъюнкция целей достижима при X=bob и Y=anna, при X=bob и Y=pate.

Внуками Tom'a являются Anna и Pate.

ВОПРОС 9: "Есть ли у Anna и Pate общий родитель ?"

?-parent(X,anna),parent(X,pate).

X=bob

Цель достижима при X=bob. Общий родитель - Bob.

5. Добавить информацию о том, каков пол людей, участвующих в

отношении родитель, с использованием унарных отношений ЖЕНЩИНА

(англ.: woman) и МУЖЧИНА (англ.:man):

woman(pame).

woman(lize).

woman(pate).

woman(anna).

man(tom).

man(bob).

man(jhim).

6. Определить отношение МАТЬ (англ.: mother) следующим образом:

mother(X,Y):-parent(X,Y),woman(X).

7. Составить и задать системе вопросы, относящиеся к отноше-

нию МАТЬ.

8. Определить рекурсивно отношение ПРЕДОК (predok) следу-

ющим образом:

predok(X,Z):-parent(X,Z).

predok(X,Z):-parent(X,Y),predok(Y,Z).

9. Задать системе вопросы, относящиеся к отношению ПРЕДОК:

"Кто является предком Pame ?"

?-predok(pame,X).

X=bob ;

X=anna ;

X=pate ;

X=hime ;

no

10. Включить режим трассировки (с помощью встроенных предикатов trace, notrace) и выполнить заново вопросы

из п.п. 4 и 7, наблюдая процесс доказательства целей интерпрета-

тором языка Пролог.

11. Оформить отчет, в который включить Пролог-программу, воп-

росы, представленные средствами Пролога, и ответы на данные воп-

росы, выданные Пролог-системой.

Получить справочные данные о предикатах системы SWI Prolog можно с помощью предиката apropos, вводя в режиме консоли

?-apropos.

или

?-apropos(<имя предиката>).

**Лабораторная работа N 2**

**РАБОТА С ОТНОШЕНИЯМИ НА ЯЗЫКЕ ПРОЛОГ**

ЦЕЛЬ РАБОТЫ: изучение реляционного подхода к программированию на

языке Пролог, получение практических навыков работы с отношениями

на языке Пролог (на примере отношений родства).

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.

2. Составить базовые отношения РОДИТЕЛЬ, ЖЕНЩИНА, МУЖЧИНА,

СУПРУГИ, представить полученные отношения в виде нагруженного ори-

ентированного графа. Каждому бинарному отношению при этом будет

соответствовать свой тип дуг.

3. Представить на языке Пролог базовые отношения.

4. Определить в виде правил и представить на языке Пролог от-

ношения родства (в соответствии со своим вариантом).

3. Набрать в текстовом редакторе системы Arity Prolog (или

любом другом тексовом редакторе) текст Пролог-программы.

4. Задать Пролог-программе вопросы, касающиеся представленных

родственных отношений, следующих типов.

4.1) "Находится ли субъект а в отношении R с субъектом b ?"

4.1.1) "Является ли субъект a родственником типа R для

субъекта b ?" [вопрос типа R(a,b)].

4.1.2) "Является ли субъект b родственником типа R для

субъекта a ?" [вопрос типа R(b,a)].

4.2) "Кто находится в отношении R с субъектом a ?"

4.2.1) "Найти всех субъектов, для которых a является

родственником типа R" [вопрос типа R(a,X)]

4.2.2) "Найти всех субъектов, которые являются родственни-

ками типа R для субъекта b" [вопрос типа R(Y,b)].

4.3) "Найти все пары субъектов, находящихся в отношении R"

[вопрос типа R(X,Y)].

5. Проверить вручную, соответсвуют ли полученные ответы ожи-

даемому результату.

6. Оформить отчет, в который включить Пролог-программу, воп-

росы, представленные средствами Пролога, и ответы на данные воп-

росы, выданные Пролог-системой.

ОСНОВНЫЕ СВЕДЕНИЯ

При использовании реляционного подхода программа рассматрива-

ется как множество взаимоопределенных, возможно, очень сложных,

отношений между объектами. В соответствии с реляционным подходом

основными задачами при составлении прикладной программы должны

быть: анализ значимых сущностей и отношений в предметной области,

выбор ограничений для представления этих сущностей и отношений, а

затем - определение каждого отношения при помощи предложений языка

Пролог.

Определяемые бинарные отношения должны соответствовать опре-

деленным свойствам и ограничениям, обеспечивающих их целостность.

Для аргументов любого бинарного отношения будет справедливо одно

из следующих ограничений: "один-к-одному", "один-к многим", "мно-

гие-к-одному", "многие-к многим". Некоторые отношения обладают

следующими свойствами: симметричности, рефлексивности, транзитив-

ности.

После того, как отношение будет точно определено с учетом

всех его свойств и ограничений, его можно использовать как неза-

висимый модуль при определении других отношений. Для анализа би-

нарные отношения часто оказывается полезным представоять в виде

графа.

В данной лабораторной работе используются бинарные отношения

родства, определенные на множестве людей. Базовыми отношениями при

этом являются: бинарное отношение РОДИТЕЛЬ(англ.: parent), унарное

отношение ЖЕНЩИНА (англ.:woman), унарное отношение МУЖЧИНА

(англ.:man), бинарное отношение СУПРУГИ англ.:married couple). Все

базовые отношения задаются в форме фактов.

Перечисленные ниже производные родственные отношения опреде-

ляются на основе базовых родственных отношений. Производные

родственные отношения разбиты на три группы: А) близкие кровные

родственники; Б) неблизкие кровные родственники; С) родственники

по закону (англ.: in law).

А. БЛИЗКИЕ КРОВНЫЕ РОДСТВЕННИКИ

1. отец (father)

2. мать (mother)

3. сын (son)

4. дочь (daughter)

5. брат (brother)

6. сестра (sister)

7. внук (grandson)

8. внучка (granddaughter)

9. дедушка (grandfather)

10. бабушка (grandmother)

11. тетя (aunt)

12. дядя (uncle)

13. племянник (nephew)

14. племянница (niece)

Б. НЕБЛИЗКИЕ КРОВНЫЕ РОДСТВЕННИКИ

1. Правнук (great\_grandson)

2. Правнучка (great\_granddaughter)

3. (N)-правнук (n\_grandson)

4. (N)-правнучка (n\_granddaughter)

5. двоюродный брат [кузен] (first\_coustin\_brother)

6. двоюродная сестра [кузина] (first\_coustin\_sister)

7. троюродный брат (second\_coustin\_brother)

8. троюродная сестра (second\_coustin\_sister)

9. (N)-юродный брат (n\_coustin\_brother)

10. (N)-юродная сестра (n\_coustin\_sister)

11. двоюродный дядя (first\_coustin\_uncle)

12. двоюродная тетя (first\_coustin\_aunt)

13. троюродный дядя (second\_coustin\_uncle)

14. троюродная тетя (second\_coustin\_aunt)

15. (N)-юродный дядя (n\_coustin\_uncle)

16. (N)-юродная тетя (n\_coustin\_aunt)

17. двоюродный племянник (first\_coustin\_nephew)

18. двоюродная племянница (first\_coustin\_niece)

19. троюродный племянник (second\_coustin\_nephew)

20. троюродная племянница (second\_coustin\_niece)

21. (N)-юродный племянник (n\_coustin\_nephew)

22. (N)-юродная племянница (n\_coustin\_niece)

23. отпрыск (любого уровня) (shoot)

24. (N)-юродный брат (любого уровня) (any\_coustin\_brother)

25. (N)-юродная сестра (любого уровня) (any\_coustin\_sister)

26. (N)-юродный дядя (любого уровня) (any\_coustin\_uncle)

27. (N)-юродная тетя (любого уровня) (any\_coustin\_aunt)

28. (N)-юродный племянник (любого уровня) (any\_coustin\_nephew)

29. (N)-юродная племянница (любого уровня) (any\_coustin\_niece)

С. РОДСТВЕННИКИ ПО ЗАКОНУ

1. муж (husband)

2. жена (wife)

3. тесть (wifes\_father)

4. теща (wifes\_mother)

5. свекор (husbands\_father)

6. свекровь (husbands\_mother)

7. деверь [брат мужа] (husbands\_brother)

8. свояченица [сестра жены] (wifes\_sister)

9. свояк [муж свояченицы] (husband\_of\_wifes\_sister)

10. зять

11. невестка

12. золовка

ВАРИАНТЫ ЛАБОРАТОРНЫХ ЗАДАНИЙ

-----------------------------------------

Номер Шифры

варианта родственных отношений

-----------------------------------------

1 А1,А11,Б7,Б17,Б24,С1

2 А2,А12,Б8,Б18,Б25,С2

3 А3,А13,Б9,Б19,Б23,С10

4 А4,А14,Б10,Б20,С9,С11

5 А5,Б1,Б11,Б21,С8,С12

6 А6,Б2,Б12,Б22,С7

7 А7,Б3,Б13,Б26,С6

8 А8,Б4,Б14,Б27,С5

9 А9,Б5,Б15,Б28,С4

10 А10,Б6,Б16,Б29,С3

-----------------------------------------

Верс. 1

**Лабораторная работа N 3**

**РАБОТА СО СПИСКАМИ НА ЯЗЫКЕ ПРОЛОГ**

Варианты заданий

----------------

Тема А. Работа с одноуровневыми списками

........................................

1) Предикат APPEND для соединения (конкатенации) двух списков

в один новый список.

2) Предикат REMOVE для удаления из списка заданного элемента.

3) Предикат REMOVEF для удаления из списка первых вхождений

данного элемента.

4) Предикат SUBSTITUTE для замены всех вхождений заданного

элемента в списке на новый элемент.

5) Предикат REVERSE, изменяющий порядок элементов в списке на

обратный.

6) Предикат ADDLAST для добавления заданного элемента в конец

списка.

7) Предикат PARITY для определения четности числа элементов в

списке.

8) Предикат SHIFT для циклического сдвига списка на один сим-

вол вправо.

9) Предикат LENGTH для определения длины списка.

Тема В. Работа с многоуровневыми списками

..........................................

1) Предикат LINEAR для линеаризации списков. При этом многоу-

ровневый список (т.е. список списков всех уровней) преобразуется в

одноуровневый. Пример: исходный спискок

[[a,b],c,[e,[f,[[g,e,p],v],y]],x], результирующий список

[a,b,c,e,f,g,e,v,y,x].

2) Предикат DEVLEV1 для разбивки списка на уровни. Пример:

исходный список [a,b,c], результирующий список [a,[b,[c]]].

3) Предикат DEVLEV2 для разбивки списка на уровни. Пример:

исходный список [a,b,c], результирующий список [[[a],b],c].

4) Предикат DESTLEV1 для уничтожения уровней в списке. При-

мер: исходный список [a,[b,[c]]], результирующий список [a,b,c].

5) Предикат DESTLEV2 для уничтожения уровней в списке. При-

мер: исходный список [[[a],b],c], результирующий список [a,b,c].

6) Предикат DEPTH для вычисления глубины списка (самой глубо-

кой ветви). Пример: исходный спискок

[[a,b],c,[e,[f,[[g,e,p],v],y]],x], глубина списка равна 5.

7) Предикат DEVPAIR для разбивки списка на пары. Пример:

исходный список [a,b,c,d], результирующий список [[a,b],[c,d]].

8) Предикат FIRST для определения первого элемента списка (в

учет принимаются списки всех уровней). Пример: исходный список:

[[[s,a],b],c,[e,[f,[[g,e,p],v],y]],x]. Первый элемент: s.

9) Предикат LENGTHFULL для определения числа неделимых эле-

ментов в многоуровневом списке. Пример: исходный список:

[[[s,a],b],c,[e,[f,[[g,e,p],v],y]],x]. Число атомов: 12.

Тема C. Операции над множествами

.................................

Множества заданы в виде списков.

1) Предикат INTERSECT для пересечения двух множеств.

2) Предикат UNION для объединения двух множеств.

3) Предикат DIFF для получения разности двух множеств.

4) Предикат SIMDIFF для получения симметрической разности

двух множеств.

5) Предикат SUBSET для определения того, является ли одно

множество подмножеством другого.

6) Предикат EQU для проверки равентсва двух множеств.

7) Предикат SETOFF для нормализации множества (исключения из

списка повторяющихся элементов).

8) Предикат CART для получения декартова произведения двух

множеств.

9) Предикат SETP для проверки, является ли список множеством,

т.е. входит ли каждый элемент в список лишь один раз.

**Лабораторная работа N 4**

**Работа с базами данных языка Пролог**

***Цель работы:*** изучение встроенных предикатов assert, asserta, assertz, retract, consult, reconsult для выполнения операций над базами данных языка Пролог, изучение способов организации циклов (для вывода на экран),  получение практических навыков работы с базами данных языка Пролог.

**Порядок выполнения работы**

1.     Включить правила, определяющие одно из производных родственных отношений, полученные в лабораторной работе N 2, в Пролог-программу с использованием предиката consult или reconsult.

2.     Включить правила, определяющие другое из производных родственных отношений, полученные в лабораторной работе N 2, в Пролог-программу с использованием предиката asserta или assertz.

3.     Создать базы фактов (БФ) для производных родственных отношений, определенных в лабораторной работе N 2.

4.     Вывести полученные БФ на экран.

5.     Удалить из БФ parent один или несколько фактов в соответствии с заданными условиями (например, удалить из БФ parent все сведения о родителях X).

6.     Вывести измененную БФ на экран.

7.     Удалить все факты из БФ, полученных в п.3.

8.     Удалить из Пролог-программы правила, включенные в нее в п.п.1-2.

**Основные сведения**

Встроенный предикат *asserta(X)* добавляет утверждение Х в начало базы данных, а *assertz(X)* - в ее конец. Результат добавления в базу данных (БД) утверждения не устраняется при выполнении возврата Утверждение Х может быть либо фактом, либо правилом. Примеры:

*% Включение факта*

*assert(parent(tom,bob)).*

*% Включение правила*

*assert((mother(X,Y):- parent(X,Y), woman(X)).*

Встроенный предикат *retract(X)* позволяет удалять утверждения из БД. При попытке выполнить целевое утверждение retract(X) находится первое утверждение в БД, с которым может быть сопоставима X, и это утверждение удаляется. Пример:

*retract(parent(ann,tom)).*

Встроенный предикат *consult* используется для добавления утверждений из некоторого файла к утверждениям, уже имеющимися в БД. Аргумент предиката должен быть атомом, указывающим имя файла, из которого должны браться утверждения. Примеры:

*consult(myfile).*

*consult(‘u:dvn/91sii/test1’).*

Предикат *reconsult*аналогичен предикату *consult* за исключением того, что вводимые утверждения заменяют все имеющиеся утвержения для того же самого предиката.

**Лабораторная работа N 5**

**Трансформация формул**

***Цель работы:*** ознакомление с основами компьютерной алгебры и символьных вычислений, изучение способов представления формул на языке Пролог, изучение встроенных предикатов для создания и декомпозиции термов (предикаты =.., functor, arg).

**Порядок выполнения работы**

1.     Представить заданные формулы (исходную, заменяемую и замещающую) в префиксной форме.

2.     Закодировать формулы в виде термов языка Пролог.

3.     Составить программу на языке Пролог для замены одного терма на другой в исходном терме.

4.     Осуществить трансформацию исходной формулы с помощью разработанной Пролог-программы.

5.     Представить полученную в п.4 формулу в инфиксной форме.

**Основные сведения**

Целевое утверждение X=..L истинно, если L есть список, состоящий из функтора структуры (терма) X, за которым следуют аргументы X. Примеры:

?-f(a,b,c)=..Z.

Z=[f,a,b,c]

?-Y=..[a,b,c,d].

Y=a(b,c,d)

**Варианты заданий**

В исходной формуле заменить подформулу на подформулу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Исходная формула F** | **Заменя-емая подфор-мула P** | **Замещаю-щая подфор-мула R** | **Комментарии** |
| 1 | (x\*y/5-z)+x\*y\*w | x\*y | x-a\*y+1 |  |
| 2 | (x~(x&y))((~y~(x&y)(xyx&y)) | ~(x&y) | ~x(~y&z) |  |
| 3 | (xz)((yz)(xyz))(xy->z)) | AB | ~AB | Исключение импликации |
| 4 | ~((x~(pq)) t&s) | ~(AB) | ~A&~B | Правило де Моргана |
| 5 | (xx&y)->(xz (xx&y)) | xx&y | xz |  |
| 6 | ~((xz)&(xy))&(~(x&z)t&u) | ~(A&B) | ~A~B | Правило де Моргана |
| 7 | (a>ba=1)&(d100f=q&(a>ba=1) | a>ba=1 | p-1=2\*q |  |
| 8 | (x&yzx)(z(y&zx)) | AB | (AB)&(BA) | Исключение эквивалентности |
| 9 | x|zyx|(zx|yy) | A|B | ~A~B | Исключение функции Шеффера |
| 10 | a\*x­4+b\*x­3+c\*x­2+d\*x+e | A­B | B\*A­(B-1) |  |
| 11 | (xy)&(ab&(zw)) | A&(BC) | A&BA&C | Закон дистрибутивности конъюнкции относительно дизъюнкции |
| 12 | p&qa&b&(cx&y) | AB&C | (AB)&(AC) | Закон дистрибутивности дизъюнкции относительно конъюнкции |

**Примечания:** В данной таблице приняты следующие обозначения. 1) Большими буквами A и B обозначены подформулы; 2) Знак  обозначает логическую функцию  «эквивалентность»; 3) знак | обозначает логическую функцию Шеффера; 4) Знак ~ обозначает логическую функцию отрицания; 5) Знак  означает логическую функцию «импликация»; 6) Знак ­ означает арифметическую операцию возведения в степень. 7) Старшинство операций (в порядке убывания): а) отрицание; б) конъюнкция, функция Шеффера; с) дизъюнкция; д) импликация.