Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Лабораторная работа №4 (Язык искусственного интеллекта PROLOG)**

**Дисциплина**: Интеллектуальные системы

Выполнил студент гр. 13541/1 Смирнов М.И.

(подпись)

Руководитель Сазанов А.М.

(подпись)

“ ” 2017 г.

Санкт – Петербург

2017

**Содержание**

[Задания и решение 3](#_Toc498283147)

[1. Постройте генеалогическое дерево, описание онтологии и концептуальную карту для демонстрационного проекта «family1» на основе результатов выполнения программы и исходного кода программы. 3](#_Toc498283148)

[2. Выполнить одно из индивидуальных заданий (из пособия Бураков С. В. «Язык логического программирования PROLOG», СПбГУАП, 2003). 6](#_Toc498283149)

[3. Изучить 1-2 лабы по методичке Седана С.Н. (доп литература). Согласно своему варианту решить задачу с помощью PROLOG. 9](#_Toc498283150)

[Выводы 12](#_Toc498283151)

[Список литературы 14](#_Toc498283152)

# **Задания и решение**

## **Постройте генеалогическое дерево, описание онтологии и концептуальную карту для демонстрационного проекта «family1» на основе результатов выполнения программы и исходного кода программы.**

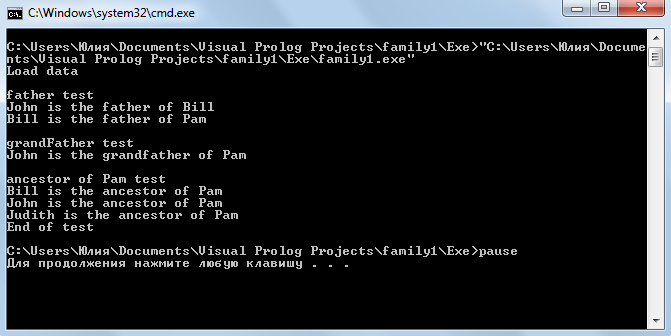
Текст программы «family1»:

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*                          Copyright (c) Sabu Francis Associates  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ implement main     open core      constants         className = "main".         classVersion = "$JustDate: 2007-09-24 $$Revision: 1 $".      clauses         classInfo(className, classVersion).      domains         gender = female(); male().      class facts - familyDB         person : (string Name, gender Gender).         parent : (string Person, string Parent).      class predicates         father : (string Person, string Father) nondeterm anyflow.     clauses         father(Person, Father) :-             parent(Person, Father),             person(Father, male()).      class predicates         grandFather : (string Person, string GrandFather) nondeterm (o,o).     clauses         grandFather(Person, GrandFather) :-             parent(Person, Parent),             father(Parent, GrandFather).      class predicates         ancestor : (string Person, string Ancestor) nondeterm (i,o).     clauses         ancestor(Person, Ancestor) :-             parent(Person, Ancestor).         ancestor(Person, Ancestor) :-             parent(Person, P1),             ancestor(P1, Ancestor).      class predicates         reconsult : (string FileName).     clauses         reconsult(FileName) :-             retractFactDB(familyDB),             file::consult(FileName, familyDB).      clauses         run():-             console::init(),             stdIO::write("Load data\n"),             reconsult("..\\fa.txt"),             stdIO::write("\nfather test\n"),             father(X, Y),                 stdIO::writef("% is the father of %\n", Y, X),             fail.         run():-             stdIO::write("\ngrandFather test\n"),             grandFather(X, Y),                 stdIO::writef("% is the grandfather of %\n", Y, X),             fail.         run():-             stdIO::write("\nancestor of Pam test\n"),             X = "Pam",             ancestor(X, Y),                 stdIO::writef("% is the ancestor of %\n", Y, X),             fail.         run():-             stdIO::write("End of test\n"). end implement main  goal     mainExe::run(main::run). |

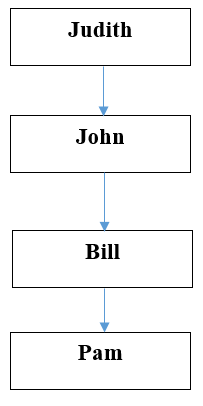
Содержимое файла «fa.txt»:

|  |
| --- |
| clauses  person("Judith",female()).  person("Bill",male()).  person("John",male()).  person("Pam",female()).  parent("John","Judith").  parent("Bill","John").  parent("Pam","Bill"). |

Результат запуска программы:



Генеалогическое дерево:



Онтология:

1. Определяем степень родства между людьми и, таким образом, построим генеалогическое дерево.
2. Входными данными будут являться люди с указанными именами и полом, а также с указанными родственными связями:

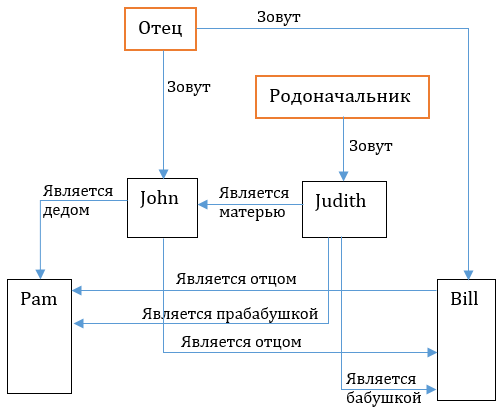
А) Определяем кто чей отец – должен быть родителем и мужчиной.

Б) Определяем кто чей дедушка – должен быть родителем родителя и мужчиной.

В) Определяем предка по восходящей линии для конкретного человека.

По итогу получим: результаты отцовских тестов, тестов на дедушку, и тестов на старших родственников для указанного человека.

Концептуальная карта (семантическая сеть), описывающая данный пример:



1. **Выполнить одно из индивидуальных заданий (из пособия Бураков С. В. «Язык логического программирования PROLOG», СПбГУАП, 2003).**

Вариант 12.

12. Четыре человека играют в домино.

Их фамилии Кузнецов, Токарев, Слесарев и Резчиков.

Профессия каждого игрока соответствует фамилии одного из других игроков.

Напротив Кузнецова сидит слесарь.

Напротив Резчикова сидит резчик.

Справа от Слесарева сидит токарь.

Кто сидит слева от кузнеца?

Программная реализация:

domains  
man=m(string,string)  
men=man\*  
slist=string\*  
   
predicates  
naprotiv(string,string,slist)  
sprava(string,string,slist)  
sleva(string,string,slist)  
find(string)  
member(man,men)  
   
clauses  
member(H,[H|\_]).  
member(H,[\_|Tail]):-member(H,Tail).  
   
naprotiv(A,B,[\_,A,\_,B]).  
naprotiv(B,A,[A,\_,B,\_]).  
naprotiv(B,A,[\_,A,\_,B]).  
   
sprava(A,B,[A,B,\_,\_]).  
sprava(A,B,[\_,\_,A,B]).  
   
sleva(A,B,L):-sprava(B,A,L).  
   
find(Ans):-L=[m(A,\_),m(B,\_),m(C,\_),m(D,\_)],Surnames=[A,B,C,D],  
    member(m(kuznecov,\_),L),member(m(tokarev,\_),L),  
    member(m(slesarev,\_),L),member(m(rezchikov,\_),L),  
    member(m(S1,kuznec),L),not(S1=kuznecov),  
    member(m(S2,tokar),L),not(S2=tokarev),  
    member(m(S3,slesar),L),not(S3=slesarev),  
    member(m(S4,rezchik),L),  
    naprotiv(kuznecov,S3,Surnames),  
    sprava(slesarev,S2,Surnames),  
        naprotiv(rezchikov,S4,Surnames),  
    sleva(Ans,S1,Surnames).  
   
goal  
find(Ans).

В задаче предполагается, что игроки сидят на прямоугольном столе друг напротив друга, по 2 человек с каждой стороны. Вполне возможно, что фамилия и профессия игрока совпадает.

Были написаны следующие предикаты:

* naprotiv(string,string,slist);
* sprava(string,string,slist);
* sleva(string,string,slist);
* find(string);
* member(man,men).

Есть список игроков, в которых игроки хранятся в соответствии со своим положением. Проходя по столу, проверяется соответствие заданным в определении задачи правилам. По итогу с помощью find мы формируем окончательное расположение игроков.

1. **Изучить 1-2 лабы по методичке Седана С.Н. (доп литература). Согласно своему варианту решить задачу с помощью PROLOG.**

Вариант 1.

Как-то раз случай свёл в купе астронома, поэта, прозаика и драматурга. Это были Алешин,  
Бобров, Константинов и Дмитриев. Оказалось, что каждый из них взял с собой книгу,  
написанную одним из пассажиров этого купе. Алешин и Бобров углубились в чтение  
предварительно обменявшись книгами. Поэт читал пьесу, прозаик — очень молодой  
человек, выпустивший свою книгу, говорил, что он никогда и ничего не читал по астрономии.  
Бобров купил одно из произведений Дмитриева. Никто из пассажиров не читал свои книги.  
Что читал каждый из них, кто кем был?

Главную роль в этой задаче играет предикат person, который получает следующие значения:

* имя человека;
* профессия;
* тему книги, которую читает человек;
* список профессий
* список книг, которые уже используются в текущем решении.

implement main  
open core  
constants  
className = "main".  
classVersion = "".  
  
clauses  
%classInfo(className, classVersion).  
domains  
professions\_list = string\*.  
reading\_list = string\*.  
class predicates  
name : (string) nondeterm anyflow.  
profession : (string) nondeterm anyflow.  
person : (string Name, string Prof, string Book, string\*,string\*) nondeterm anyflow.  
check\_reading : (string Prof, string Book) nondeterm anyflow.  
check\_aleks\_astr : (string AProf, string BRead) nondeterm anyflow.  
in\_list : (string, string\*) nondeterm anyflow.  
  
clauses  
name("Alekseev"). name("Borisov"). name("Konstantinov"). name("Dmitriev").  
profession("Astronomy"). profession("Poet"). profession("Proze"). profession("Drama").  
person(Name, Prof, Book, Professions, Readings):-  
name(Name),  
profession(Prof),not(in\_list(Prof,Professions)),  
profession(Book),not(in\_list(Book,Readings)),  
check\_reading(Prof,Book).  
check\_reading("Poet","Drama").  
check\_reading("Proze",Book):-  
profession(Book),Book<>"Proze",Book<>"Astronomy".  
check\_reading(Prof, Book):-  
profession(Prof),  
profession(Book),  
Prof<>"Poet",Prof<>"Proze",  
Prof<>Book.  
check\_aleks\_astr(AProf, BRead):-  
AProf<>"Proze",!;  
BRead<>"Astronomy",!.  
in\_list(Item,[Item|Tail]).  
in\_list(Item,[Head|Tail]):-  
in\_list(Item, Tail).  
  
run():-  
console::init(),  
person("Dmitriev",DProf,DRead,[],[]),  
DProf<>"Proze",   
ARead=DProf,    
person("Alekseev",AProf,ARead,[DProf],[DRead]),  
person("Borisov",BProf,BRead,[DProf, AProf],[DRead, ARead]),  
BRead <> AProf,    
check\_aleks\_astr(AProf, BRead),    
person("Konstantinov",KProf,KRead,[DProf, AProf, BProf],[DRead, ARead, BRead]),  
  
stdIO::writef("% is % and reading % book\n","Alekseev",AProf,ARead),  
stdIO::writef("% is % and reading % book\n","Dmitriev",DProf,DRead),  
stdIO::writef("% is % and reading % book\n","Borisov",BProf,BRead),  
stdIO::writef("% is % and reading % book\n","Konstantinov",KProf,KRead),  
stdIO::write("\_\_\_\_\_\_\_\_\_\n\n"),  
fail.  
  
run():-  
stdIO::write("End of test\n").  
end implement main  
goal  
mainExe::run(main::run).

|  |  |
| --- | --- |
| **Результаты работы программы** | |
| 1. Dmitriev is Astronomy and reading Proze book. 2. Alekseev is Drama and reading Astronomy book. 3. Konstantinov is Poet and reading Drama book. 4. Borisov is Proze and reading Poet book. | 1. Dmitriev is Drama and reading Astronomy book. 2. Alekseev is Poet and reading Drama book. 3. Konstantinov is Proze and reading Poet book. 4. Borisov is Astronomy and reading Proze book. |

# **Выводы**

В работе я познакомился с особенностями языка и системы логического программирования PROLOG. С помощью него выполнил несколько задача проектирования интеллектуальных систем, узнал семантику и синтаксис языка, научился использовать основные базовые операции в системе.

1. В чем плюсы и минусы языка Prolog?

+ Просто чтения языка, так как язык является логическим;

+ Возможность использования в качестве невычислительного языка программирования (т.е. применение только выражений и фактов).

+ Огромная вариация путей решения одной и той же задачи;

- Язык неразвитый;

- Невозможность обособленного использования языка (практически всегда используется как дополнение к процедурным языкам);

- неудобство использования данного языка при решении задач с вычислительными операциями.

1. Какие еще языки используются для разработки ИИ, приведите примеры (НЕ МЕНЕЕ 2-х) проектов, языков и краткое описание проектов. (Альтернативы PROLOG)

**LISP**. Первый компьютерный язык, применяемый для создания искусственного интеллекта — ЛИСП. Этот язык является довольно таки гибким и расширяемым. Такие особенности, как быстрое прототипирование и макросы очень полезны в создании ИИ. Мощная система объектно-ориентированности делает LISP одним из самых популярных языков программирования для искусственного интеллекта.

**Java**. Основные преимущества этого многофункционального языка являются: прозрачность, переносимость и удобство сопровождения, а также легкая отладка, хорошее взаимодействие с пользователем, простота работы с большими проектами. Для новичков существуют сотни видеоуроков в Интернете, что упрощается изучение данного языка.

1. Решаема ли проблема комбинаторного взрыва, пути решения?

Пока не удается найти иного способа, кроме полного перебора всех возможных вариантов. Для обхода проблемы комбинаторного взрыва ищут специальные методы решения, в частности, применяют эвристические алгоритмы.

1. Корректно ли по-вашему в принципе разработка языка ИИ? Что он должен из себя представлять?

Сейчас в мире всё развивается невероятно быстрыми темпами и область ИИ тому не исключение. Уже сделаны множественные попытки разработки языков ИИ. Я считаю, что прогресс должен идти и человечество вскоре увидит настоящий искусственный интеллект.

Такой язык будет как минимум заключать в себе уже реализованные базовые библиотеки, функции для работы со средой искусственного интеллекта. И иметь огромную книгу документации по эксплуатации в нескольких томах.

1. Можно ли разработать ИИ не понимая, как он работает, должны ли мы понимать, как он работает, думает, рассуждает?

Однозначно должны. Такую сложную систему просто невозможно разработать, не имея глубочайших познаний в данной сфере. А управление и контроль за правильностью и безопасностью работы уже с готовым искусственным интеллектом будет заставлять правительство вводить новые учебные заведения и новые необходимые профессии.

# **Список литературы**

1. Бураков М.В. Язык логического программирования Prolog. СПб.: СПбГУАП, 2003.
2. Коста Э. Visual Prolog 7.1 для начинающих под ред. Е. А. Ефимовой. СПб., 2008.
3. Visual Prolog [Электронный ресурс]. – Режим доступа: http://www.visual-prolog.com/ (Дата обращения: 12.11.17).
4. Энциклопедия языков программирования [Электронный ресурс]. – Режим доступа:

http://progopedia.ru/language/prolog/ (Дата обращение: 12.11.17).