|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 16**

|  |  |
| --- | --- |
| **Дисциплина** Функциональное и логическое программирование  **Тема** \_Использование правил в программе на Prolog\_  **Студент** \_Ильясов И. М.\_  **Группа** \_ИУ7-63Б\_  **Оценка (баллы)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Преподаватель** \_Толпинская Н. Б., Строганов Ю. В.\_ |  |

Москва, 2020 г.

**Цель работы** –изучить использование правил в программе: структуру, особенности оформления, а также, способ и принципы выполнения таких программ на Prolog.

**Задачи работы**:

* приобрести навыки эффективного декларативного описания предметной области с использованием фактов и правил;
* изучить порядок использования фактов и правил в программе на Prolog, принципы и особенности сопоставления и отождествления термов, на основе механизма унификации. Способ формирования и изменения резольвенты. Порядок формирования ответа

**Задание лабораторной работы**

**Создать базу знаний: «ПРЕДКИ»**, позволяющую **наиболее эффективным** способом (за меньшее количество шагов, что обеспечивается меньшим количеством предложений БЗ - правил), используя разные варианты (примеры) **одного вопроса**, определить (указать: какой вопрос для какого варианта):

1. по имени субъекта определить всех его бабушек (предки 2-го колена),
2. по имени субъекта определить всех его дедушек (предки 2-го колена),
3. по имени субъекта определить всех его бабушек и дедушек (предки 2-го колена),
4. по имени субъекта определить его бабушку по материнской линии (предки 2-го колена),
5. по имени субъекта определить его бабушку и дедушку по материнской линии (предки 2-го колена).

Минимизировать количество правил и количество вариантов вопросов. Использовать **конъюнктивные правила и простой вопрос.**

**Для одного** из вариантов **ВОПРОСА** и конкретной БЗ **составить таблицу**, отражающую конкретный порядок работы системы, с объяснениями:

очередная проблема на каждом шаге и метод ее решения;

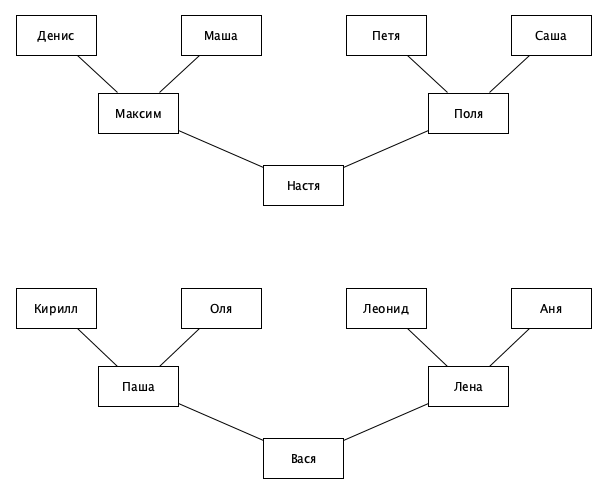
каково новое текущее состояние резольвенты, как получено;

какие дальнейшие действия? (Запускается ли алгоритм унификации? Каких термов? Почему этих?);

вывод по результатам очередного шага и дальнейшие действия.

Т.к. резольвента хранится в виде стека, то состояние резольвенты требуется отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового состояния резольвенты!

Генеалогическое представление семьи:



**Текст программы**

domains

father, mother, child = symbol.

predicates

create\_family(father, mother, symbol)

get\_grandpars(symbol P, symbol GP1, symbol GM1, symbol GP2, symbol GM2)

clauses

create\_family(denis, masha, max).

create\_family(petya, sasha, polya).

create\_family(max, polya, nastya).

create\_family(kirill, olya, pasha).

create\_family(leonid, anya, lena).

create\_family(pasha, lena, vasya).

get\_grandpars(P, GP1, GM1, GP2, GM2) :-

create\_family(F, M, P),

create\_family(GP1, GM1, F),

create\_family(GP2, GM2, M).

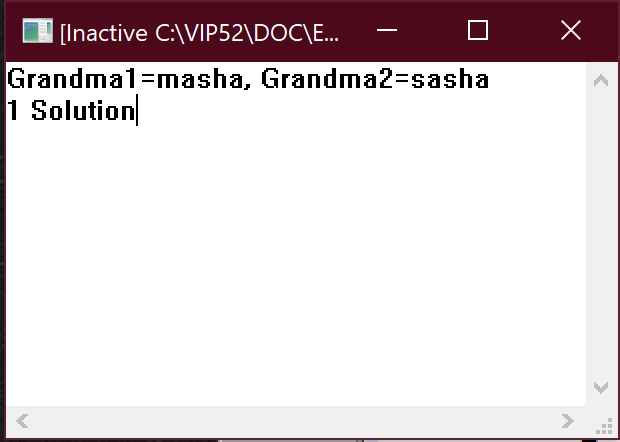
goal

get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).

**Примеры работы программы**

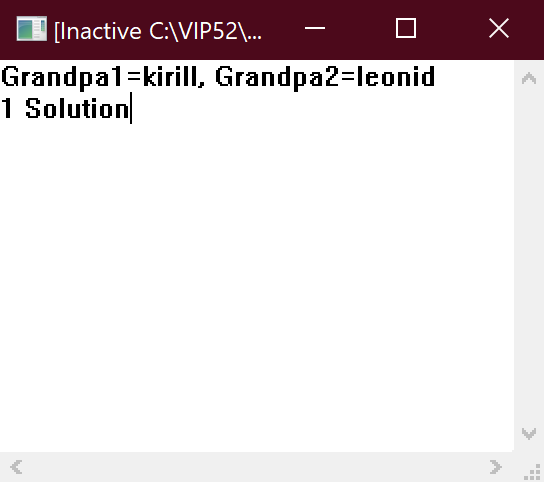
1. По имени субъекта определить всех его бабушек (предки 2-го колена).

Ниже приведен ответ на вопрос о всех бабушках Насти.



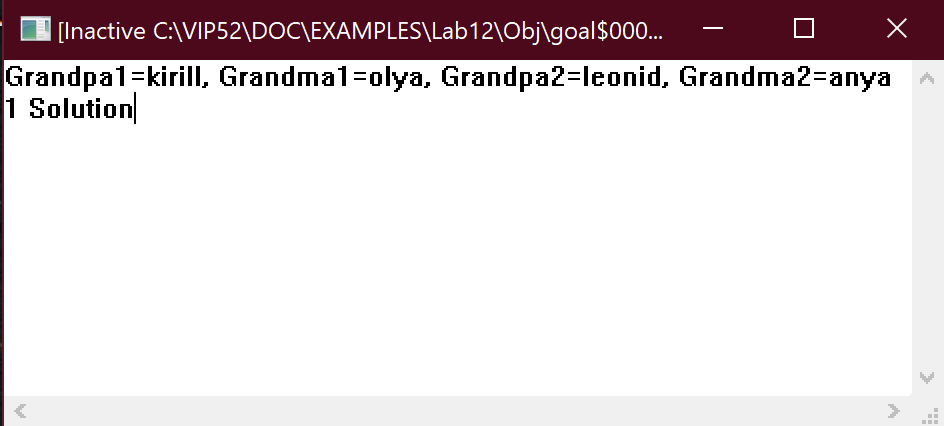
1. По имени субъекта определить всех его дедушек (предки 2-го колена).

Ниже приведен ответ на вопрос о всех дедушках Васи.



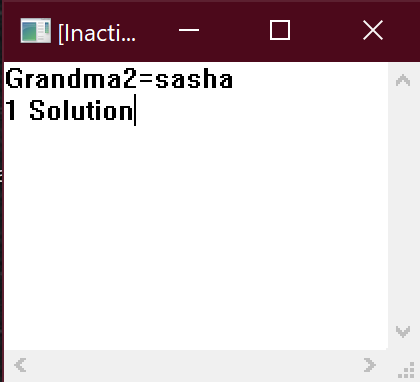
1. По имени субъекта определить всех его бабушек и дедушек (предки 2-го колена).

Ниже приведен ответ на вопрос о всех бабушках и дедушках Васи.



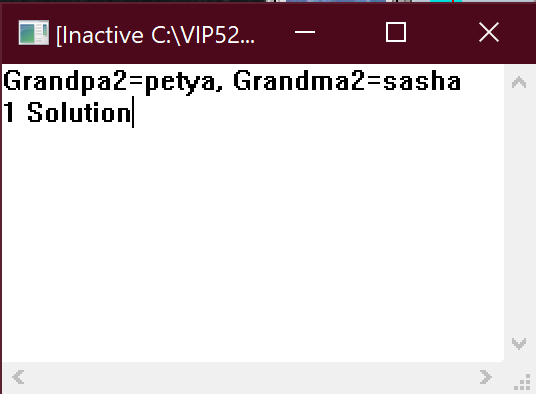
1. По имени субъекта определить его бабушку по материнской линии (предки 2-го колена).

Ниже приведен ответ на вопрос о бабушке Насти по материнской линии.



1. По имени субъекта определить его бабушку и дедушку по материнской линии (предки 2-го колена).

Ниже приведен ответ на вопрос о бабушке и дедушке Насти по материнской линии.



**Задание с таблицей**

get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) – по имени субъекта имена всех бабушек.

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты, и вывод: дальнейшие действия (почему?) | Для каких термов запускается алгоритм унификации: Т1=Т2 и каков **результат** (и подстановка) | Дальнейшие действия: прямой ход или откат (почему и к чему приводит?) |
| 1 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2)) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(denis, masha, max).  Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 2 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(petya, sasha, polya).Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 3 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(max, polya, nastya).  Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 4 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(kirill, olya, pasha).  Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 5 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(leonid, anya, lena).  Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 6 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2).  T2 = create\_family(pasha, lena, vasya).  Неудача, разные функторы. | Прямой ход к следующему предложению. |
| 7 | get\_grandpars(nastya, \_, Grandma1, \_, Grandma2) | T1 = get\_grandpars(nastya, \_, Grandma1, \_, Grandma2)  T2 = get\_grandpars(P, GP1, GM1, GP2, GM2).  Успех, подстановка {nastya=X, Grandma1=GM1, Grandma2=GM2}. | Прямой ход к сопоставлению family(F, M, nastya), поиск с начала предложений. |
| 8 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(denis, masha, max).  Неудача, nastya != max. | Прямой ход к следующему предложению. |
| 9 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(petya, sasha, polya).  Неудача, nastya != polya. | Прямой ход к следующему предложению. |
| 10 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(max, polya, nastya).  Успех, подстановка {F=max, M=polya, nastya=nastya}. | Прямой ход к сопоставлению create\_family(\_, GM1, max), поиск с начала предложений. |
| 11 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(denis, masha, max).  Успех, подстановка {GM1=masha, max=max}. | Прямой ход к сопоставлению create\_family(\_, GM2, polya), поиск с начала предложений. |
| 12 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(denis, masha, max).  Неудача, polya != max. | Прямой ход к следующему предложению. |
| 13 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(petya, sasha, polya).  Успех, подстановка {GM2=sasha, polya=polya}. | **Вывод: GM1=masha, GM2=sasha.**  Прямой ход к следующему предложению, реконкретизация GM2. |
| 14 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(max, polya, nastya).  Неудача, polya != nastya. | Прямой ход к следующему предложению. |
| 15 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(kirill, olya, pasha).  Неудача, polya != pasha. | Прямой ход к следующему предложению. |
| 16 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(leonid, anya, lena).  Неудача, polya != lena. | Прямой ход к следующему предложению. |
| 17 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = create\_family(pasha, lena, vasya).  Неудача, polya != vasya. | Прямой ход к следующему предложению. |
| 18 | create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM2, polya),  T2 = get\_grandpars(P, GP1, GM1, GP2, GM2).  Неудача, разные функторы. | Откат, переход к предыдущему состоянию резольвенты (шаг 11), реконкретизация GM1. |
| 19 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(petya, sasha, polya).  Неудача, max != polya. | Прямой ход к следующему предложению. |
| 20 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(max, polya, nastya).  Неудача, max != nastya. | Прямой ход к следующему предложению. |
| 21 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(kirill, olya, pasha).  Неудача, max != pasha. | Прямой ход к следующему предложению. |
| 22 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(leonid, anya, lena).  Неудача, max != lena. | Прямой ход к следующему предложению. |
| 23 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = create\_family(pasha, lena, vasya).  Неудача, max != vasya. | Прямой ход к следующему предложению. |
| 24 | create\_family(\_, GM1, max),  create\_family(\_, GM2, polya) | T1 = create\_family(\_, GM1, max),  T2 = get\_grandpars(P, GP1, GM1, GP2, GM2).  Неудача, разные функторы. | Откат, переход к предыдущему состоянию резольвенты (шаг 10), реконкретизация F и M. |
| 25 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(kirill, olya, pasha).  Неудача, nastya != pasha. | Прямой ход к следующему предложению. |
| 26 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(leonid, anya, lena).  Неудача, nastya != lena. | Прямой ход к следующему предложению. |
| 27 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = create\_family(pasha, lena, vasya).  Неудача, nastya != vasya. | Прямой ход к следующему предложению. |
| 28 | create\_family(F, M, nastya),  create\_family(\_, GM1, F),  create\_family(\_, GM2, M) | T1 = create\_family(F, M, nastya),  T2 = get\_grandpars(P, GP1, GM1, GP2, GM2)  Неудача, разные функторы. | Откат, переход к предыдущему состоянию резольвенты (шаг 7). |
| 29 | get\_grandpars(nastya, \_, GM1, \_, GM2);  конец clauses;  опустошение резольвенты;  завершение работы. |  |  |

**Ответы на вопросы**

1. **В каком случае система запускает алгоритм унификации? (Как эту необходимость на формальном уровне распознает система?)**

Если есть что доказывать (цель), то процесс унификации запускается автоматически. Формально: если резольвента не пуста – запускается алгоритм унификации.

1. **Каковы назначение и результат использования алгоритма унификации?**

Назначение алгоритма унификации заключается в попарном сопоставлении термов и попытке построить для них общий пример. Унификация может завершаться успехом или тупиковой ситуацией (неудачей).

1. **Какое первое состояние резольвенты?**

Если задан простой вопрос, то сначала он попадает в резольвенту.

1. **Как меняется резольвента?**

Изменение резольвенты происходит в 2 этапа:

1) из стека выбирается подцель (верхняя, т.к. стек) и для нее выполняется редукция, т.е. замена подцели на тело найденного правила;

2) к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор выбранной цели и заголовка сопоставленного с этой целью правила).

1. **В каких пределах программы уникальны переменные?**

Переменные уникальны в пределах предложения, т.е. в рамках предложения одно и то же имя принадлежит одной и той же переменной. Исключение – анонимные переменные (обозначаются символом нижнего подчеркивания «\_») – каждая такая переменная является отдельной сущностью и применяется, когда ее значение неважно для данного предложения.

1. **Как применяется подстановка, полученная с помощью алгоритма унификации?**

Применение подстановки {*X1=T1, …, Xn=Tn*} заключается в замене каждого вхождения переменной *Xi* на соответствующий терм *Ti*.

1. **В каких случаях запускается механизм отката?**

Механизм отката запускается в 2 случаях:

1. Если алгоритм попал в тупиковую ситуацию.

2. Если резольвента не пуста и решение найдено, но в базе знание остались не отмеченные предложения.