|  |  |
| --- | --- |
| 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»

**ОТЧЕТ**

*к лабораторной работе №16-17*

*По курсу: «Функциональное и логическое программирование»*

**Темы:** «Использование правил в программе на Prolog», «Формирование эффективных программ на Prolog»

Студент: Якуба Д.В.

Группа: ИУ7-63Б

Преподаватели: Толпинская Н. Б.,

Строганов Ю. В.

Москва, 2021 г.

# Практическая часть

# Лабораторная работа 16.

Задание. Создать базу знаний: «ПРЕДКИ», позволяющую наиболее эффективным способом (за меньшее количество шагов, что обеспечивается меньшим количеством предложений БЗ – правил), и 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.

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

|  |
| --- |
| domains  sex = symbol.  name = string.  parent = parent(name, sex).  predicates  isParentOf(parent, name).  ancestors(name, name, name, name, name).  clauses  isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  isParentOf(parent("Alexandra Romanova", w), "Morgan Romanov").    isParentOf(parent("Helen Romanova", w), "Alexey Romanov").  isParentOf(parent("Vasiliy Romanov", m), "Alexey Romanov").  isParentOf(parent("Klavdiya Portak", w), "Helen Romanova").  isParentOf(parent("Evheny Portak", m), "Helen Romanova").  isParentOf(parent("Valentina Romanova", w), "Vasiliy Romanov").  isParentOf(parent("Elisey Romanov", m), "Vasiliy Romanov").  isParentOf(parent("Anastasia Grishkovec", w), "Alexandra Romanova").  isParentOf(parent("Gennadiy Sehenswurdigkeiten", m), "Alexandra Romanova").  isParentOf(parent("Irina Grishkovec", w), "Anastasia Grishkovec").  isParentOf(parent("Sergey Grishkovec", m), "Anastasia Grishkovec").  isParentOf(parent("Klavdiya Sehenswurdigkeiten", w), "Gennadiy Sehenswurdigkeiten").  isParentOf(parent("Dmitry Sehenswurdigkeiten", m), "Gennadiy Sehenswurdigkeiten").  ancestors(ChildName, MothersMother, MothersFather, FathersMother, FathersFather) :-  isParentOf(parent(MothersName, w), ChildName), isParentOf(parent(FathersName, m), ChildName),  isParentOf(parent(MothersMother, w), MothersName), isParentOf(parent(MothersFather, m), MothersName),  isParentOf(parent(FathersMother, w), FathersName), isParentOf(parent(FathersFather, m), FathersName).  goal  % ancestors("Morgan Romanov", MothersMotherName, \_, FathersMotherName, \_)  % Point 1  % MothersMotherName=Anastasia Grishkovec,  % FathersMotherName=Helen Romanova    % ancestors("Morgan Romanov", \_, MothersFatherName, \_, FathersFatherName)  % Point 2  % MothersFatherName=Gennadiy Sehenswurdigkeiten,  % FathersFatherName=Vasiliy Romanov    % ancestors("Alexey Romanov", MothersMotherName, MothersFatherName, FathersMotherName, FathersFatherName)  % Point 3  % MothersMotherName=Klavdiya Portak,  % MothersFatherName=Evheny Portak,  % FathersMotherName=Valentina Romanova,  % FathersFatherName=Elisey Romanov    % ancestors("Alexey Romanov", MothersMotherName, \_, \_, \_)  % Point 4  % MothersMotherName=Klavdiya Portak    ancestors("Alexandra Romanova", MothersMotherName, MothersFatherName, \_, \_)  % Point 5  % MothersMotherName=Irina Grishkovec,  % MothersFatherName=Sergey Grishkovec  . |

ancestors("Alexey Romanov", MothersMotherName, \_, \_, \_)

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 (к чему приводит?) |
| 1 | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_)  Резольвента непуста, запуск алгоритма унификации для терма, извлекаемого из стека | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_) = isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  Унификация неуспешна (несовпадение функторов) | Откат, переход к следующему предложению |

# Лабораторная работа 17.

Задание. **В одной программе написать правила, позволяющие найти**

1. Максимум из двух чисел:
   1. Без использования отсечения;
   2. С использованием отсечения;
2. Максимум из трех чисел:
   1. Без использования отсечения;
   2. С использованием отсечения.

Убедиться в правильности результатов. Для каждого случая из пункта 2 обосновать необходимость всех условий тела. Для одного из вариантов ВОПРОСА и каждого варианта задания 2 составить таблицу, отражающую конкретный порядок работы системы.

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

Требуется ответить на вопрос: «За счет чего может быть достигнута эффективность работы системы?»

|  |
| --- |
| predicates  maxOfTwo(real, real, real).  maxOfTwoCut(real, real, real).  maxOfThree(real, real, real, real).  maxOfThreeCut(real, real, real, real).  clauses  maxOfTwo(FVar, SVar, FVar) :- FVar >= SVar.  maxOfTwo(FVar, SVar, SVar) :- FVar < SVar.  maxOfTwoCut(FVar, SVar, FVar) :- FVar >= SVar, !.  maxOfTwoCut(\_, SVar, SVar).  maxOfThree(FVar, SVar, TVar, FVar) :- FVar >= SVar, FVar >= TVar.  maxOfThree(FVar, SVar, TVar, SVar) :- SVar >= FVar, SVar >= TVar.  maxOfThree(FVar, SVar, TVar, TVar) :- TVar >= SVar, TVar >= FVar.  maxOfThreeCut(FVar, SVar, TVar, FVar) :- FVar >= SVar, FVar >= TVar, !.  maxOfThreeCut(\_, SVar, TVar, SVar) :- SVar >= TVar, !.  maxOfThreeCut(\_, \_, TVar, TVar).  goal  maxOfTwo(1.2, 1.333, Max); % 1.333  maxOfTwo(1.999, -2, Max); % 1.999  maxOfTwo(1.999, 1.999, Max); % 1.999, 1.999  maxOfTwoCut(1.2, 1.333, Max); % 1.333  maxOfTwoCut(1.999, -2, Max); % 1.999  maxOfTwoCut(1.999, 1.999, Max); % 1.999  maxOfThree(1.1, 1.2, 1.3, Max); % 1.3  maxOfThree(1.3, 1.2, 1.1, Max); % 1.3  maxOfThree(1.3, 1.1, 1.2, Max); % 1.3  maxOfThree(1.2, 1.1, 1.3, Max); % 1.3  maxOfThree(1.2, 1.2, 1.3, Max); % 1.3  maxOfThree(1.2, 1.3, 1.2, Max); % 1.3  maxOfThree(1.3, 1.2, 1.2, Max); % 1.3  maxOfThree(111, 111, 111, Max); % 111 111 111  maxOfThreeCut(1.1, 1.2, 1.3, Max); % 1.3  maxOfThreeCut(1.3, 1.2, 1.1, Max); % 1.3  maxOfThreeCut(1.3, 1.1, 1.2, Max); % 1.3  maxOfThreeCut(1.2, 1.1, 1.3, Max); % 1.3  maxOfThreeCut(1.2, 1.2, 1.3, Max); % 1.3  maxOfThreeCut(1.2, 1.3, 1.2, Max); % 1.3  maxOfThreeCut(1.3, 1.2, 1.2, Max); % 1.3  maxOfThreeCut(111, 111, 111, Max) % 111  . |

Обоснование.

|  |
| --- |
| maxOfThree(FVar, SVar, TVar, FVar) :- FVar >= SVar, FVar >= TVar.  maxOfThree(FVar, SVar, TVar, SVar) :- SVar >= FVar, SVar >= TVar.  maxOfThree(FVar, SVar, TVar, TVar) :- TVar >= SVar, TVar >= FVar. |

Условия в теле первого правила изложены для определения факта того, что первое переданное число больше двух других чисел или равно им. Условия в теле второго правила изложены из факта того, что второе переданное число больше двух других чисел или равно им. Условия в теле третьего правила изложены из факта того, что третье переданное число больше двух других чисел или равно им.

|  |
| --- |
| maxOfThreeCut(FVar, SVar, TVar, FVar) :- FVar >= SVar, FVar >= TVar, !.  maxOfThreeCut(\_, SVar, TVar, SVar) :- SVar >= TVar, !.  maxOfThreeCut(\_, \_, TVar, TVar). |

Условия в теле первого правила изложены для определения факта того, что первое переданное число больше или равно двух других чисел. В случае, если условие не выполняется, FVar – точно меньше второго переданного значения SVar, поэтому в теле второго правила определяется, больше или равно значение SVar значения TVar. В случае, если условие не выполняется, TVar – наибольшее из трёх чисел. В случае выполнения условия первого или второго правила смысла далее идти по предложениям процедуры нет, так как искомое значение уже было найдено, поэтому применяется системный предикат отсечения.

maxOfTwo(1.2, 1.333, Max)

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 (к чему приводит?) |
| 1 | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_)  Резольвента непуста, запуск алгоритма унификации для терма, извлекаемого из стека | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_) = isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  Унификация неуспешна (несовпадение функторов) | Откат, переход к следующему предложению |

maxOfTwoCut(1.2, 1.333, Max)

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 (к чему приводит?) |
| 1 | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_)  Резольвента непуста, запуск алгоритма унификации для терма, извлекаемого из стека | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_) = isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  Унификация неуспешна (несовпадение функторов) | Откат, переход к следующему предложению |

maxOfThree(1.1, 1.2, 1.3, Max)

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 (к чему приводит?) |
| 1 | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_)  Резольвента непуста, запуск алгоритма унификации для терма, извлекаемого из стека | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_) = isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  Унификация неуспешна (несовпадение функторов) | Откат, переход к следующему предложению |

maxOfThreeCut(1.2, 1.3, 1.2, Max)

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 (к чему приводит?) |
| 1 | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_)  Резольвента непуста, запуск алгоритма унификации для терма, извлекаемого из стека | Ancestors(“Alexey Romanov”, MothersMotherName, \_, \_, \_) = isParentOf(parent("Alexey Romanov", m), "Morgan Romanov").  Унификация неуспешна (несовпадение функторов) | Откат, переход к следующему предложению |