



**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

О т ч е т

по лабораторной работе № 12

Дисциплина: «Функциональное и логическое программирование»

Выполнила: Овчинникова А. П.

Группа: ИУ7-65Б

Преподаватель: Толпинская Н. Б.

Строганов Ю. В.

Москва, 2020

Задание

Составить программу – базу знаний, с помощью которой можно определить, например, множество студентов, обучающихся в одном ВУЗе. Студент может одновременно обучаться в нескольких ВУЗах. Привести примеры возможных вариантов вопросов и варианты ответов (не менее 3-х). Описать порядок формирования вариантов ответа.

Исходную базу знаний сформировать с помощью только фактов.

*Исходную базу знаний сформировать, используя правила.

*Разработать свою базу знаний (содержание произвольно).

Теоретическая часть

Основным элементом языка является терм. Терм – это:

1. Константа:

- Число (целое, вещественное);
- Символьный атом (комбинация символов латинского алфавита, цифр и символа подчеркивания, начинающаяся со строчной буквы: `aA`, `ab_2`), используется для обозначения конкретного объекта предметной области или для обозначения конкретного отношения;
- Строка: последовательность символов, заключенных в кавычки.

2. Переменная:

- Именованная – обозначается комбинацией символов латинского алфавита, цифр и символа подчеркивания, начинающейся с прописной буквы или символа подчеркивания (`X`, `A21`, `_X`);
- Анонимная – обозначается символом подчеркивания (`_`).

3. Составной терм:

- Это средство организации группы отдельных элементов знаний в единый объект, синтаксически представляется: $f(t_1, t_2, \dots, t_m)$, где f - функтор (функциональный символ), t_1, t_2, \dots, t_m – термы, в том числе и составные (их называют аргументами). Аргументом или параметром составного терма может быть

константа, переменная или составной объект. Число аргументов предиката называется его арностью или местностью. Составные термы с одинаковыми функторами, но разной арности, обозначают разные отношения.

Программа на Prolog представляет собой базу знаний и вопрос. База знаний состоит из **предложений** - CLAUSES (отдельных знаний или утверждений): **фактов** и **правил**.

Предложение более общего вида – правило – имеет вид:

$$A :- B1, \dots, Bn.$$

A называется заголовком правила, а $B1, \dots, Bn$ – телом правила.

Правило – это предложение, истинность которого зависит от истинности одного или нескольких предложений. Обычно правило содержит несколько хвостовых целей, которые должны быть истинными для того, чтобы правило было истинным. Правило называют условной истиной, а факт, не содержащий тела – безусловной истиной.

Факт – это частный случай правила. **Факт** – это предложение, в котором отсутствует тело (т. е. тело пустое).

Причем, $A, B1, \dots, Bn$ – это термы; символ ":-" это специальный символ-разделитель.

Факт констатирует, что между объектами выполнено некоторое отношение. Факт представляет собой безусловно истинное утверждение.

Вопрос состоит только из тела – составного терма (или нескольких составных термов). Вопросы используются для выяснения выполнимости некоторого отношения между описанными в программе объектами.

В Prolog существует понятие процедуры. Процедурой называется совокупность правил, заголовки которых имеют одно и то же имя и одну и ту же арность (местность), т. е. это совокупность правил, описывающих одно определенное отношение. Отношение, определяемое процедурой, называется предикатом.

Программа на Прологе состоит из следующих семи разделов.

- Директивы компилятора.

В самом начале программы можно расположить одну или несколько директив компилятора, которые дают компилятору дополнительные инструкции по обработке программы (trace, nowarnings, include).

- CONSTANTS — раздел описания констант.

Объявление константы имеет вид:

<имя константы> = <значение>

В этом разделе прописные и строчные символы не различаются, поэтому в качестве первого символа имени константы можно использовать строчные символы. Однако при использовании констант в разделе описания предложений нужно задействовать в качестве первого символа имени константы только строчные символы.

- DOMAINS — раздел описания доменов.

Раздел описания доменов является аналогом раздела описания типов в обычных императивных языках программирования. Основные стандартные домены:

- integer — целое число.
- real — действительное число.
- char — символ, заключенный в одиночные апострофы.
- string — последовательность символов, заключенная в двойные кавычки.
- symbol — символическая константа.
- file — файл.

В разделе описания доменов объявляются любые нестандартные домены в формате:

$\langle \text{имя домена} \rangle = \langle \text{определение домена} \rangle$

или

$file = \langle \text{имя файлового домена} 1 \rangle; \dots; \langle \text{имя файлового домена} N \rangle$

Можно использовать описание доменов для сокращения имен стандартных доменов.

Из доменов можно конструировать составные или структурные домены (структуры):

$\langle \text{имя структуры} \rangle = \langle \text{имя функтора} \rangle (\langle \text{имя домена первой компоненты} \rangle, \dots, \langle \text{имя домена последней компоненты} \rangle) [; \langle \text{имя функтора} \rangle (\dots)]^*$

Каждая компонента структуры, в свою очередь, может быть структурой.

Списковый домен описывается следующим образом:

$\langle \text{имя спискового домена} \rangle = \langle \text{имя домена элемента списка} \rangle^*$

Можно описывать вариантные домены:

$\langle \text{имя домена} \rangle = \langle \text{имя домена 1-го варианта} \rangle; \dots; \langle \text{имя домена N-ого варианта} \rangle$

- DATABASE — раздел описания предикатов внутренней базы данных. Здесь описываются те предикаты, которые можно в процессе выполнения программы добавлять во внутреннюю базу данных или удалять оттуда.

- PREDICATES — раздел описания предикатов.

Содержит описания определяемых пользователем предикатов. Имя предиката должно быть идентификатором.

Домены аргументов должны быть либо стандартными, либо объявленными в разделе описания доменов.

В Прологе есть так называемые стандартные (встроенные) предикаты, которые не нужно описывать в

разделе описания предикатов PREDICATES.

- CLAUSES — раздел описания предложений.

Содержит факты и правила, реализующие пользовательские предикаты. Все предикаты, которые применяются в этом разделе и не являются стандартными предикатами, должны быть описаны в разделе описания предикатов или в разделе описания предикатов базы данных.

- GOAL — раздел описания внутренней цели.

Если этот раздел отсутствует, то после запуска программы Пролог-система выдает приглашение вводить вопросы в диалоговом режиме (внешняя цель).

Программа, компилируемая в исполняемый файл, который можно запускать независимо от среды разработки, обязательно должна иметь внутреннюю цель.

В программе не обязательно должны быть все эти разделы. В программе может быть несколько разделов описаний DOMAINS, PREDICATES, DATABASE и CLAUSES. Однако разделов GOAL не может быть в программе более одного. Порядок разделов может быть произвольным, но при этом константы, домены и предикаты должны быть определены до их использования. Однако в разделе DOMAINS можно ссылаться на домены, которые будут объявлены позже.

Программа на Prolog может содержать вопрос в программе (так называемая внутренняя цель GOAL). Если программа содержит внутреннюю цель, то после запуска программы на выполнение система проверяет достижимость заданной цели, исходя из базы знаний.

Ответ на поставленный вопрос система дает в логической форме – «Да» или «нет». Цель системы состоит в том, чтобы на поставленный вопрос найти возможность, исходя из базы знаний, ответить «Да». Вариантов ответить «Да» на поставленный вопрос может быть несколько. Система может быть

настроена в режим получения всех возможных вариантов ответа «Да» на поставленный вопрос.

Поиск содержательного ответа на поставленный вопрос, с помощью имеющейся базы знаний, фактически заключается в поиске нужного знания, но какое знание понадобится – заранее неизвестно. Этот поиск осуществляется формально с помощью механизма унификации, встроенного в систему и не доступного программисту.

Упрощенно, процесс унификации можно представить как формальный процесс сравнения (сопоставления) термина вопроса с очередным термом знания. При этом знания по умолчанию просматриваются сверху вниз, хотя такой порядок и не очевиден. В процессе сравнения для переменных «подбираются», исходя из базы знаний, значения (для именованных переменных). И эти подобранные для переменных значения возвращаются в качестве побочного эффекта ответа на поставленный вопрос.

Цель работы программы – определить является ли вопрос логическим следствием программы или нет, что выполняется с применением правил вывода. Правила вывода – это утверждения о взаимосвязи между допущениями и заключениями, которые с позиции исчисления предикатов верны всегда.

Существенным недостатком в использовании этих правил вывода является необходимость «угадать» подстановку или пример термина. Кроме этого, переменные в факте и в вопросе могут стоять на одной позиции. Поэтому для выполнения логического вывода используется механизм (алгоритм) унификации, встроенный в систему.

Унификация – операция, которая позволяет формализовать процесс логического вывода (наряду с правилом резолюции). С практической точки зрения — это основной вычислительный шаг, с помощью которого происходит:

- Двухнаправленная передача параметров процедурам,
- Неразрушающее присваивание,

- Проверка условий (доказательство).

В процессе работы система выполняет большое число унификаций. Процесс унификации запускается автоматически, но пользователь имеет право запустить его принудительно с помощью утверждения (немного нарушает форму записей): $T1 = T2$. Унификация – попытка "увидеть одинаковость" – сопоставимость двух термов, может завершаться успехом или тупиковой ситуацией (неудачей). В последнем случае включается механизм отката к предыдущему шагу.

Факты, правила, и вопросы могут содержать переменные. Имя переменной может состоять из букв латинского алфавита, цифр, знаков подчеркивания и должно начинаться с прописной буквы или знака подчеркивания. Переменные в факты и правила входят только с квантором всеобщности. А в вопросы переменные входят только с квантором существования. Переменная в Прологе обозначает объект, а не некоторую область памяти.

Переменные предназначены для передачи значений «во времени и в пространстве». В логическом программировании все переменные рассматриваются как безтиповые.

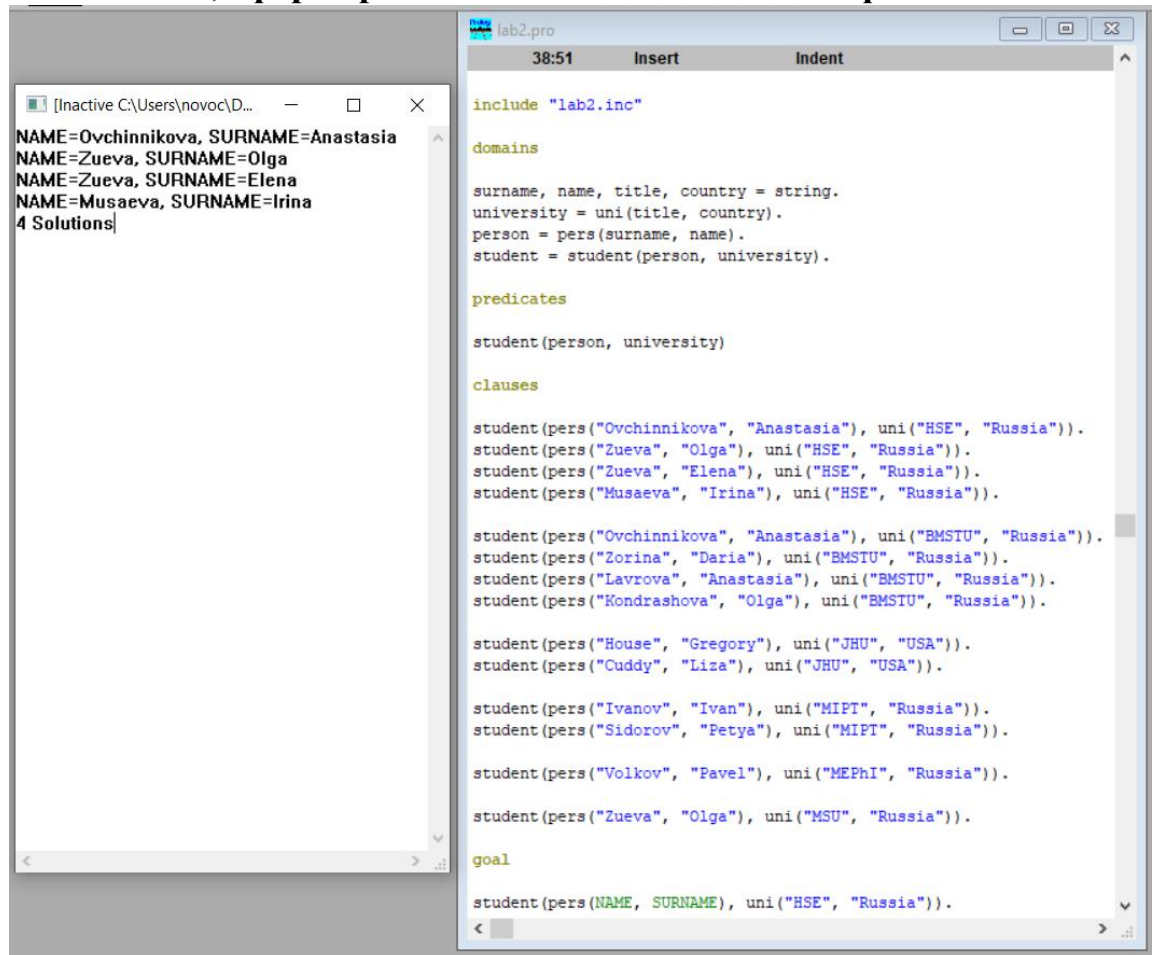
Переменные могут быть свободными или связанными. Свободная (неконкретизированная) переменная – это переменная, которая еще не получила значения. Переменная, которая получила какое-то значение и оказалась связанной с определенным объектом, называется связанной. Если переменная была конкретизирована каким-то значением и ей сопоставлен некоторый объект, то эта переменная уже не может быть изменена. В логическом программировании поддерживается механизм деструктивной конкретизации переменной. Т. е. используется идея реконкретизации переменной путем «отката» вычислительного процесса и отказа от выполненной ранее конкретизации. Это реализовано для возможности поиска нового значения для именованной переменной.

Областью действия переменной в Прологе является одно предложение. Исключением из правила определения области действия является анонимная переменная, которая обозначается символом подчеркивания «_». Анонимная переменная применяется в случае, когда значение переменной не важно. Каждая анонимная переменная – это отдельный объект.

Именованные переменные уникальны в рамках предложения, а анонимная переменная – любая уникальна. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов.

Практическая часть

1. База знаний, сформированная с помощью только фактов



The screenshot shows a Prolog IDE with two windows. The left window displays the solutions to a query, and the right window displays the source code of the knowledge base.

Left Window (Solutions):

```
[Inactive C:\Users\novoc\D...]
```

```
NAME=Ovchinnikova, SURNAME=Anastasia  
NAME=Zueva, SURNAME=Olga  
NAME=Zueva, SURNAME=Elena  
NAME=Musaeva, SURNAME=Irina  
4 Solutions|
```

Right Window (lab2.pro):

```
38:51 Insert Indent  
  
include "lab2.inc"  
  
domains  
  
surname, name, title, country = string.  
university = uni(title, country).  
person = pers(surname, name).  
student = student(person, university).  
  
predicates  
  
student(person, university)  
  
clauses  
  
student(pers("Ovchinnikova", "Anastasia"), uni("HSE", "Russia")).  
student(pers("Zueva", "Olga"), uni("HSE", "Russia")).  
student(pers("Zueva", "Elena"), uni("HSE", "Russia")).  
student(pers("Musaeva", "Irina"), uni("HSE", "Russia")).  
  
student(pers("Ovchinnikova", "Anastasia"), uni("BMSTU", "Russia")).  
student(pers("Zorina", "Daria"), uni("BMSTU", "Russia")).  
student(pers("Lavrova", "Anastasia"), uni("BMSTU", "Russia")).  
student(pers("Kondrashova", "Olga"), uni("BMSTU", "Russia")).  
  
student(pers("House", "Gregory"), uni("JHU", "USA")).  
student(pers("Cuddy", "Liza"), uni("JHU", "USA")).  
  
student(pers("Ivanov", "Ivan"), uni("MIPT", "Russia")).  
student(pers("Sidorov", "Petya"), uni("MIPT", "Russia")).  
  
student(pers("Volkov", "Pavel"), uni("MEPhI", "Russia")).  
  
student(pers("Zueva", "Olga"), uni("MSU", "Russia")).  
  
goal  
  
student(pers(NAME, SURNAME), uni("HSE", "Russia")).
```

Рисунок 1. База знаний, сформированная с помощью только фактов (часть 1).

Факты в базе знаний не содержат переменных, поэтому являются основными. Вопрос содержит переменные, поэтому является неосновным. В этом случае используется простое правило: обобщение (факта) – если есть

такая подстановка, что вопрос $Q\Theta$ логически следует из программы, то и вопрос Q следует из программы.

В примере на рисунке 1 система ищет имена и фамилии всех студентов, которые учатся в «HSE» в стране «Russia». Знания (по умолчанию) просматриваются сверху вниз. В процессе сравнения для переменных «подбираются», исходя из базы знаний, значения (для именованных переменных). И эти подобранные для переменных значения возвращаются в качестве побочного эффекта ответа на поставленный вопрос.

Первый факт в базе знаний – *«student(pers("Ovchinnikova", "Anastasia"), uni("HSE", "Russia")).»* (T1), цель – *«student(pers(NAME, SURNAME), uni("HSE", "Russia")).»* (T2). Для выполнения логического вывода используется механизм (алгоритм) унификации, встроенный в систему. Первый факт в базе знаний – составной терм. Для унификации составных термов проверяются три условия:

- T1 и T2 имеют одинаковые главные функторы – да;
- T1 и T2 имеют одинаковые аргументы – да;
- успешно унифицируется каждая пара их соответствующих компонент (аргументов) – да:
 - первая пара компонент – составные термы. Необходимо их унифицировать:
 - первая пара компонент – константа и неконкретизированная переменная – унификация успешна;
 - вторая пара компонент – константа и неконкретизированная переменная – унификация успешна;

Унификация первой пары компонент (первых составных термов) успешна.

- вторая пара компонент – составные термы. Необходимо их унифицировать:

- первая пара – две одинаковые константы – унификация успешна;
 - вторая пара – две одинаковые константы – унификация успешна.
- Унификация второй пары компонент (первых составных термов) успешна.

Таким образом первый факт в базе знаний и цель унифицированы успешно. Значения неконкретизированным переменным «подбираются», исходя из базы знаний. Т. е. переменные NAME и SURNAME конкретизируются в данном случае значениями «Ovchinnikova» и «Anastasia» соответственно. И эти подобранные для переменных значения возвращаются в качестве побочного эффекта ответа на поставленный вопрос. Найденное решение выводится на экран.

Аналогичным образом анализируются все остальные факты. Если цель и текущий факт не унифицируемы, значит, это решение не подходит, на экран ничего не выводится.

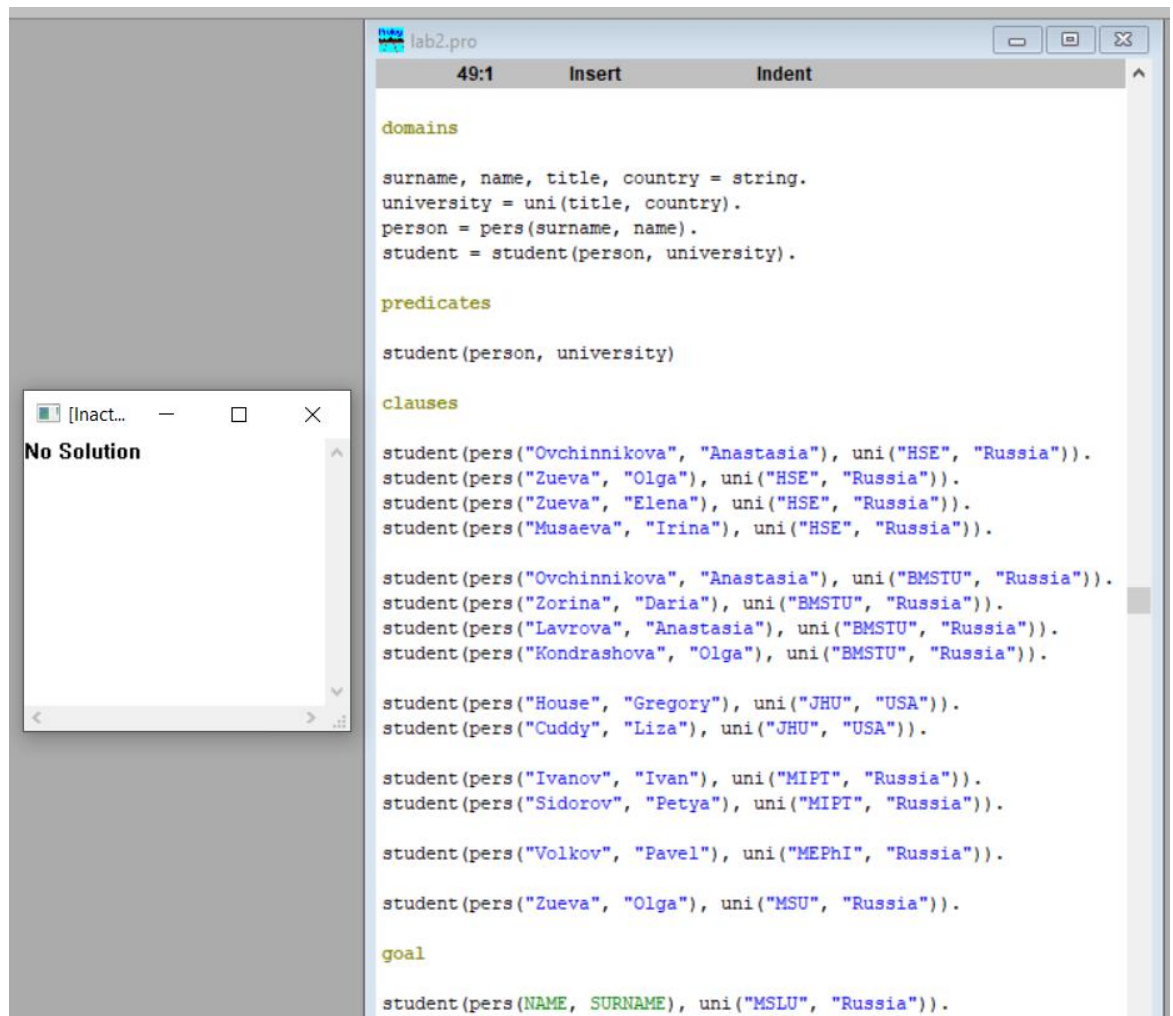
После обработки всех фактов система выводит на экран количество найденных значений.

В примере 2 на рисунке 2 система работает так же, как в примере 1. Однако ни одна унификация не будет успешной, поэтому система выведет сообщение «No solutions».

В примере 3 на рисунке 3 система ищет названия всех ВУЗов России, в которых учится студент с именем «Olga» и фамилией «Zueva». Рассуждения аналогичны примеру 1, однако в данном примере неконкретизированная переменная только одна.

В примере 4 на рисунке 4 выполняется поиск имен всех студентов из ВУЗа «HSE» страны «Russia» с фамилией «Zueva». Рассуждения аналогичны

примеру 1, однако в данном примере неконкретизированная переменная только одна.



The screenshot shows a Prolog IDE window titled 'lab2.pro'. The main editor displays a knowledge base with the following structure:

```
domains
surname, name, title, country = string.
university = uni(title, country).
person = pers(surname, name).
student = student(person, university).

predicates
student(person, university)

clauses
student(pers("Ovchinnikova", "Anastasia"), uni("HSE", "Russia")).
student(pers("Zueva", "Olga"), uni("HSE", "Russia")).
student(pers("Zueva", "Elena"), uni("HSE", "Russia")).
student(pers("Musaeva", "Irina"), uni("HSE", "Russia")).

student(pers("Ovchinnikova", "Anastasia"), uni("BMSTU", "Russia")).
student(pers("Zorina", "Daria"), uni("BMSTU", "Russia")).
student(pers("Lavrova", "Anastasia"), uni("BMSTU", "Russia")).
student(pers("Kondrashova", "Olga"), uni("BMSTU", "Russia")).

student(pers("House", "Gregory"), uni("JHU", "USA")).
student(pers("Cuddy", "Liza"), uni("JHU", "USA")).

student(pers("Ivanov", "Ivan"), uni("MIPT", "Russia")).
student(pers("Sidorov", "Petya"), uni("MIPT", "Russia")).

student(pers("Volkov", "Pavel"), uni("MEPhI", "Russia")).

student(pers("Zueva", "Olga"), uni("MSU", "Russia")).

goal
student(pers(NAME, SURNAME), uni("MSLU", "Russia")).
```

On the left side, a small window titled '[Inact...]' displays the message 'No Solution'.

Рисунок 2. База знаний, сформированная с помощью только фактов (часть 2).

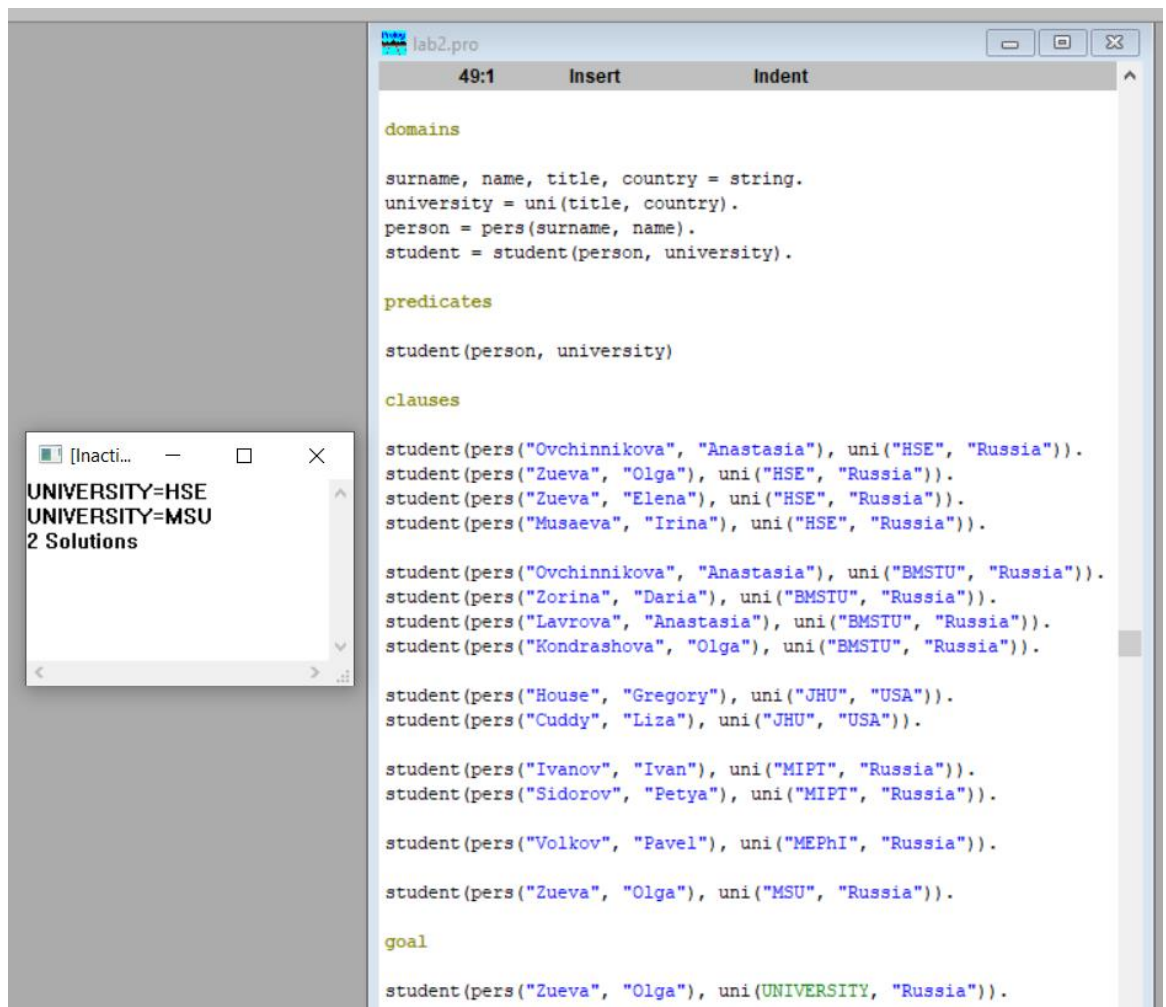


Рисунок 3. База знаний, сформированная с помощью только фактов (часть 3).

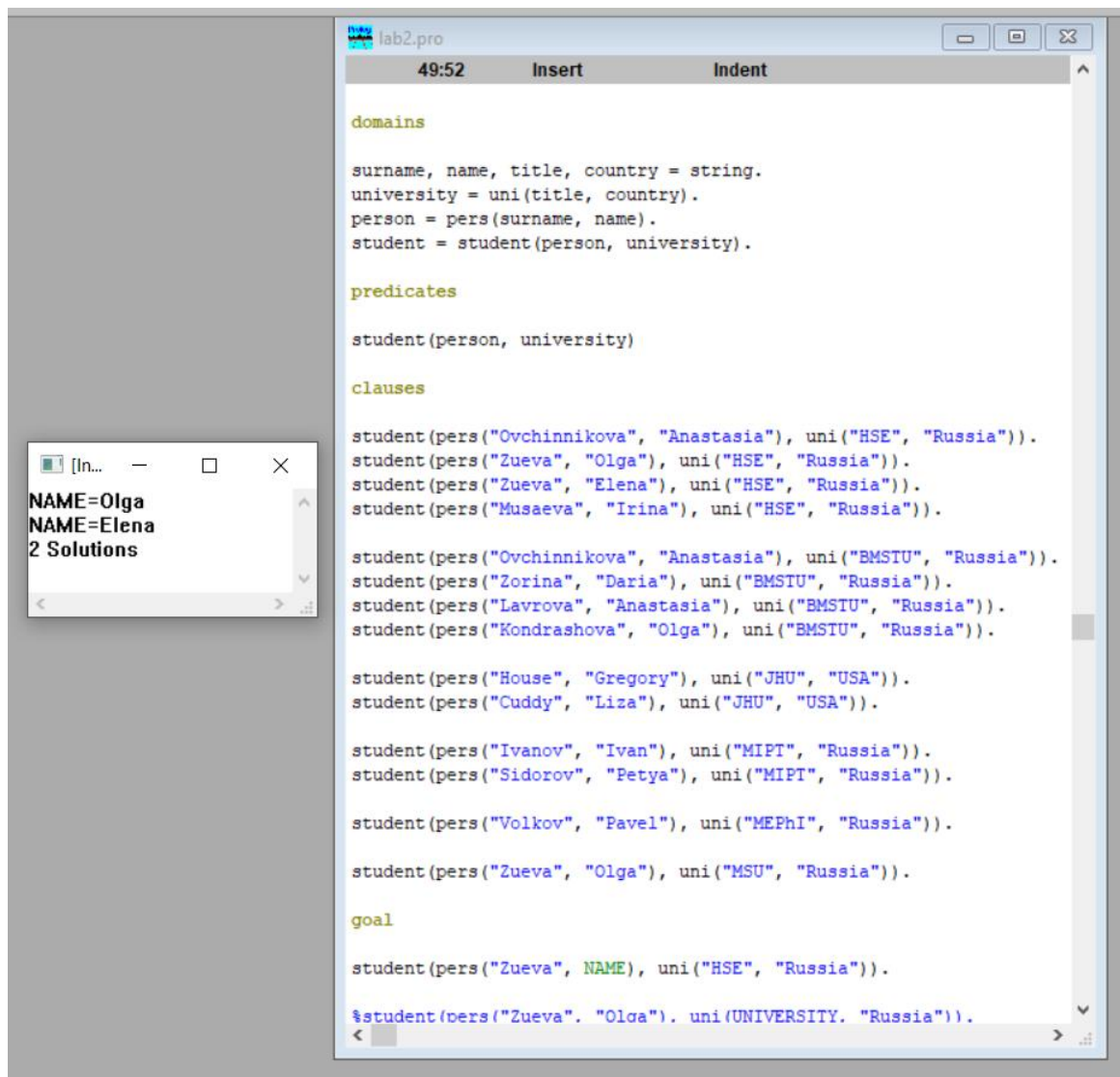


Рисунок 4. База знаний, сформированная с помощью только фактов (часть 4).

2. База знаний, сформированная с использованием правил

Эта база знаний отличается от первой наличием правил.

В общем случае в заголовке правила записывается предикат, а в теле – составная цель: $P :- G_1, G_2, \dots, G_n$. Для всех переменных правила, фигурирующий в его левой и правой части, мыслится квантор общности. Для тех же переменных, которые встречаются только в правой части, подразумевается квантор существования. В то время как факт есть безусловно истинное утверждение, правило описывает утверждение, которое истинно, только если выполнены все цели в правой части правила.

На рисунке 5 показан процесс доказательства предложения 2 («student(pers(NAME, SURNAME), uni("HSE", "Russia")).»). Предложение

доказано, и переменным «NAME» и «SURNAME» сопоставляются значения «Anastasia» и «Ovchinnikova» соответственно.

Процесс доказательства остальных предложений в базе знаний происходит аналогично. Унификация фактов происходит так, как показано в примере 1.

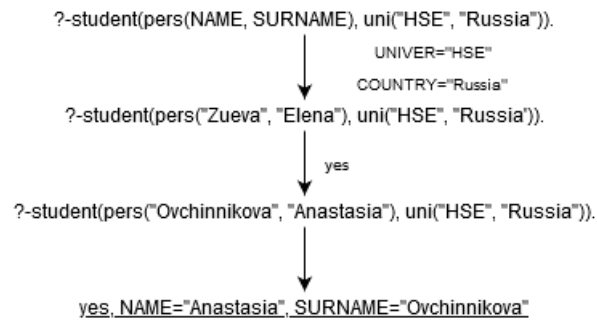


Рисунок 5. Доказательство предложения 2.

В примере 5 на рисунке 6 выполняется поиск имен и фамилий всех студентов, обучающихся в «HSE» в России.

В примере 6 на рисунке 7 поиск студентов, обучающихся в «MSLU» в России. Так как ни одно правило и ни один факт в базе знаний не являются унифицируемыми, система выводит сообщение «No solutions».

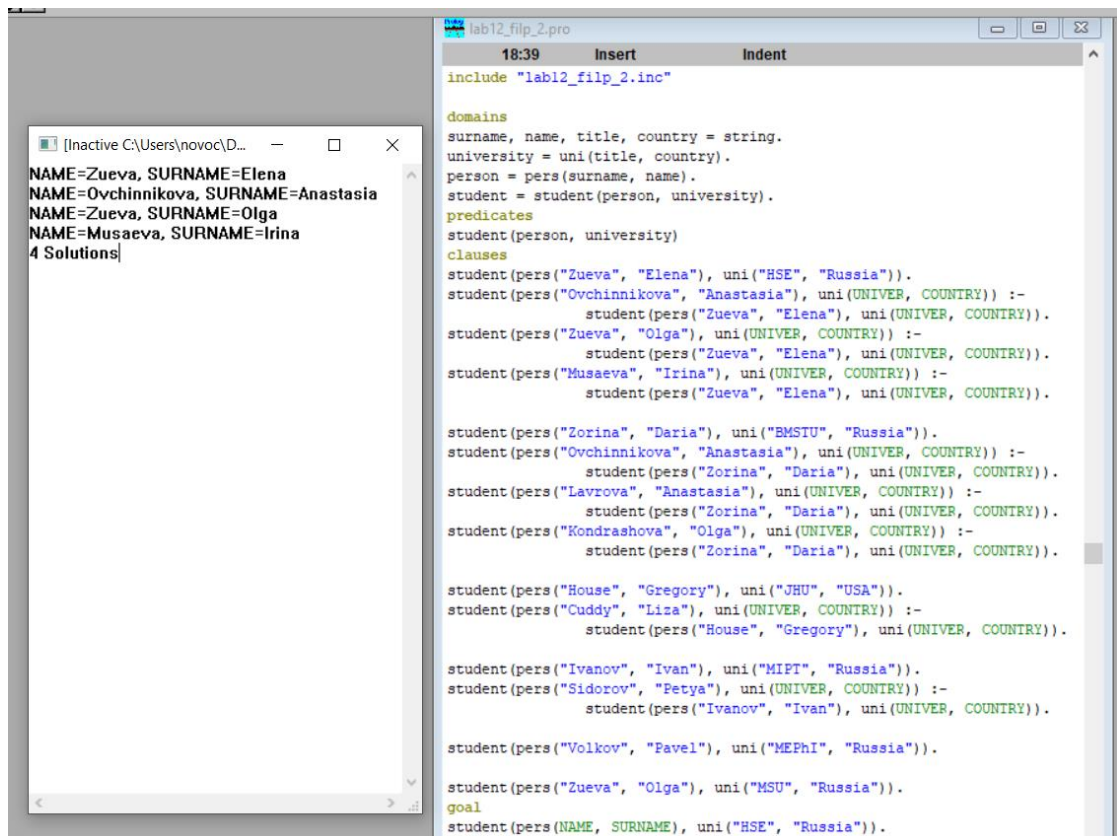


Рисунок 6. База знаний, сформированная с помощью правил (часть 1).

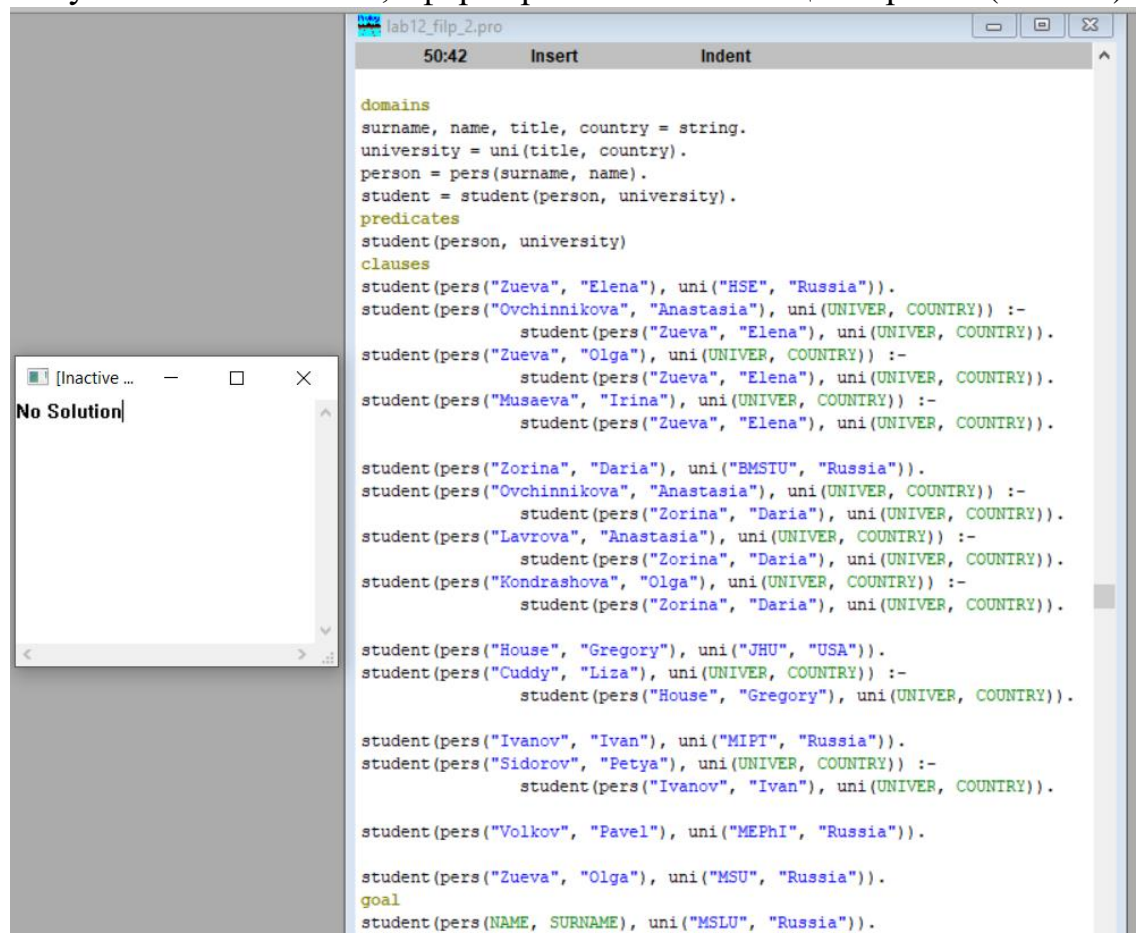


Рисунок 7. База знаний, сформированная с помощью правил (часть 2).

В примере 7 на рисунке 8 выводятся все данные обо всех студентах в базе знаний. Любое правило и любой факт в базе знаний является унифицируемым, поэтому система выводит на экран все, что есть в базе знаний.

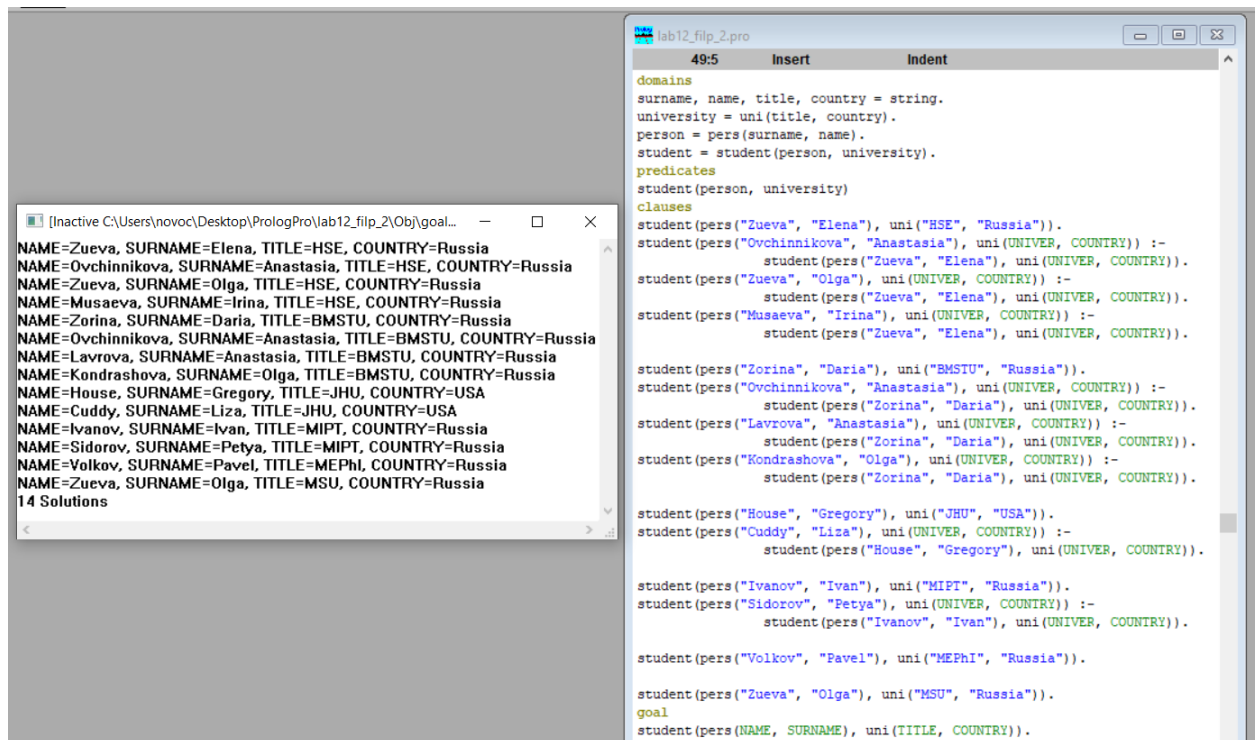


Рисунок 8. База знаний, сформированная с помощью правил (часть 3).

3. База знаний с произвольным содержанием

База знаний определяет спектральный класс звезды и ее видимый цвет в зависимости от ее температуры. Если все утверждения какого-либо правила выполняются, то правило доказано. В данной базе знаний может быть только одно решение, или решений нет вообще.

В примере 8 и 9 на рисунках 9 и 10 соответственно система определяет класс звезды по заданной температуре и получает одно решение. В примере 10 на рисунке 11 решений нет, так как ни одно правило не может быть доказано. В примере 12 выводится только класс звезды. Анонимная переменная унифицируема с любым термом и значение, которым она конкретизируется, не выводится на экран.

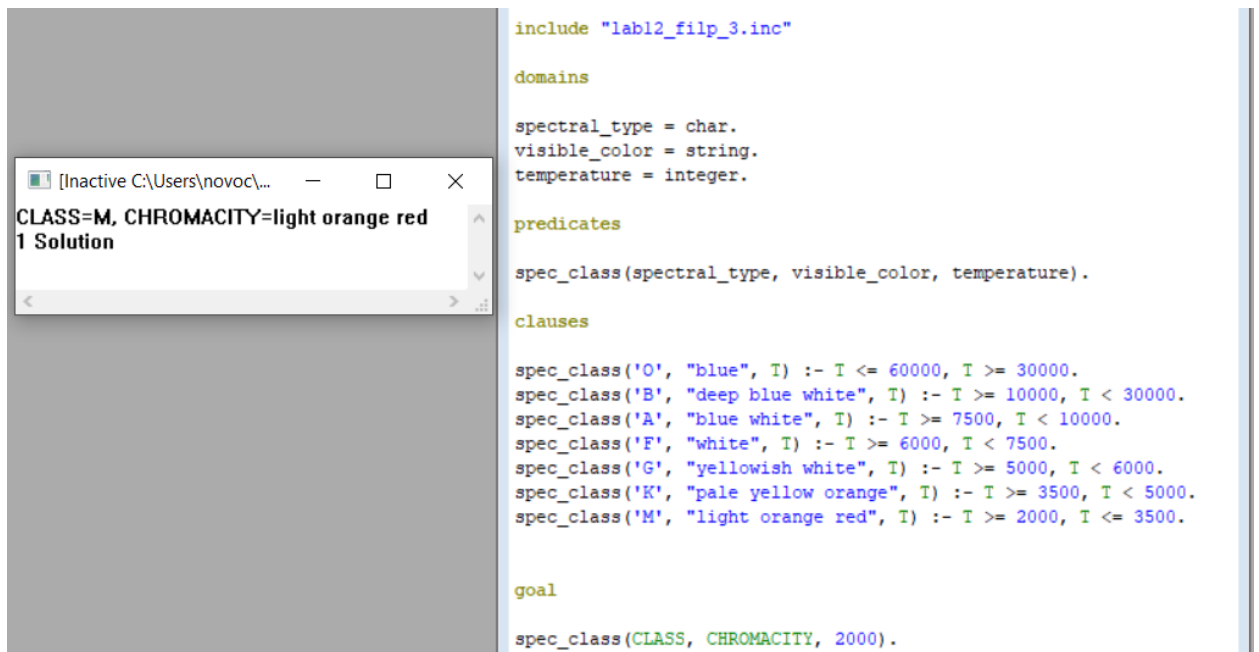


Рисунок 9. База знаний с произвольным содержанием (часть 1).

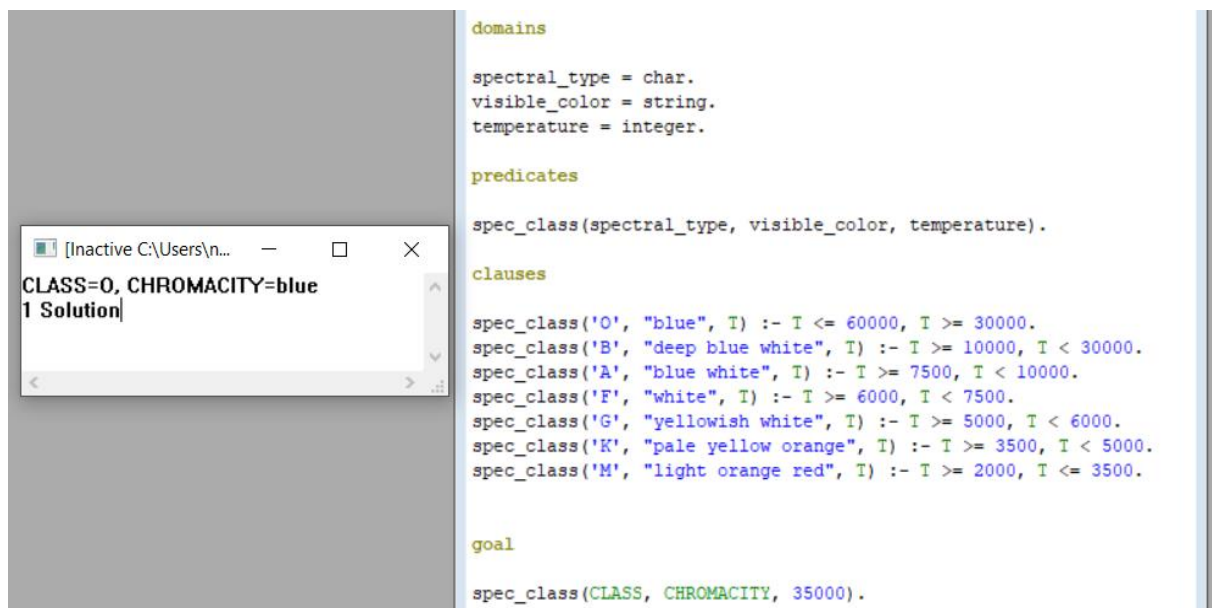


Рисунок 10. База знаний с произвольным содержанием (часть 2).

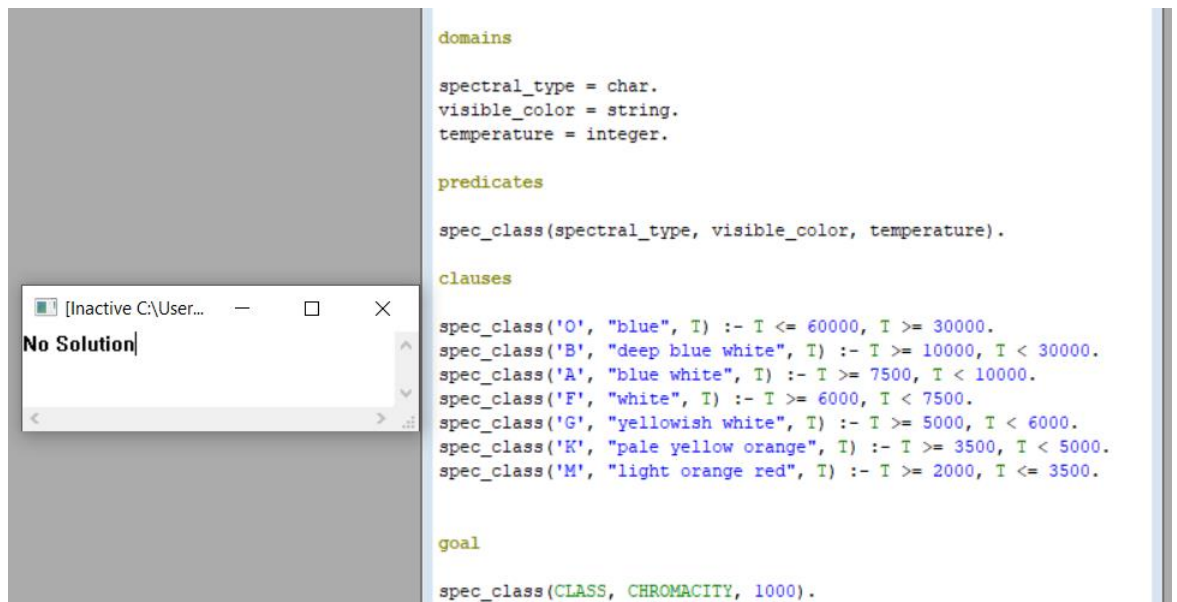


Рисунок 11. База знаний с произвольным содержанием (часть 3).



Рисунок 12. База знаний с произвольным содержанием (часть 4).