



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 15

Структура программы на Prolog и ее реализация

Дисциплина	Функциональное и логическое программирование
Студент	Сиденко А.Г.
Группа	ИУ7-63Б
Преподаватель	Толпинская Н.Б., Строганов Ю.В.

Москва, 2020 г.

Задание

Создать базу знаний «Собственники», дополнив базу знаний, хранящую знания (лаб. 13):

- «Телефонный справочник»: Фамилия, №тел, Адрес – структура (Город, Улица, №дома, №кв),
- «Автомобили»: Фамилия_владельца, Марка, Цвет, Стоимость, и др.,
- «Вкладчики банков»: Фамилия, Банк, счет, сумма, др.,

знаниями о дополнительной собственности владельца. Преобразовать знания об автомобиле к форме знаний о собственности. Вид собственности (кроме автомобиля):

- Строение, стоимость и другие его характеристики;
- Участок, стоимость и другие его характеристики;
- Водный_транспорт, стоимость и другие его характеристики.

Описать и использовать вариантный домен: Собственность. Владелец может иметь, но только один объект каждого вида собственности (это касается и автомобиля), или не иметь некоторых видов собственности.

Используя конъюнктивное правило и разные формы задания одного вопроса, обеспечить возможность поиска:

1. Названий всех объектов собственности заданного субъекта,
2. Названий и стоимости всех объектов собственности заданного субъекта,
3. Разработать правило, позволяющее найти суммарную стоимость всех объектов собственности заданного субъекта.

Для 2-го пункт и одной фамилии составить таблицу, отражающую конкретный порядок работы системы, с объяснениями порядка работы и особенностей использования доменов (указать конкретные T1 и T2 и полную подстановку на каждом шаге)

Программа

```
1 domains
2  lastname , city , street , model , color , bank , name = symbol
3  telephone , house , flat , price , account , deposit ,
4                                     price = integer
```

```

5 latitude , longitude = real
6 adress = adress(city , street , house , flat)
7 ownership = house(adress , price);
8             ground(latitude , longitude , price);
9             auto(model , color , price);
10            boat(name , price)
11
12 predicates
13 abonent(lastname , telephone , adress).
14 depositor(lastname , city , bank , account , deposit).
15 own(lastname , city , ownership).
16 owns(lastname , city , name).
17 ownsPrice(lastname , city , name , price).
18 ownsSumPrice(lastname , city , price).
19 boatPrice(lastname , city , price).
20 housePrice(lastname , city , price).
21 groundPrice(lastname , city , price).
22 autoPrice(lastname , city , price).
23
24 clauses
25 abonent(ellen , 111111 , adress(moscow , tverskaya , 1 , 1)).
26 abonent(john , 222222 , adress(moscow , arbat , 11 , 112)).
27 abonent(tom , 333333 , adress(moscow , presnya , 10 , 11)).
28 abonent(eric , 444444 , adress(moscow , pokrovka , 21 , 55)).
29 abonent(mark , 555555 , adress(moscow , solyanka , 13 , 13)).
30 abonent(mark , 888888 , adress(kazan , pushkin , 22 , 130)).
31 abonent(bill , 666666 , adress(ekb , lenin , 12 , 88)).
32 abonent(bill , 777777 , adress(spb , sadovaya , 1 , 12)).
33 depositor(ellen , moscow , sberbank , 10000 , 2000).
34 depositor(john , moscow , sberbank , 10000 , 3000).
35 depositor(john , moscow , vtb , 20000 , 5000).
36 depositor(tom , moscow , gazprom , 5000 , 3000).
37 depositor(eric , moscow , sberbank , 100000 , 30000).
38 depositor(mark , moscow , sberbank , 10000 , 2000).
39 depositor(mark , kazan , vtb , 10000 , 2000).
40 depositor(mark , moscow , gazprom , 10000 , 2000).
41 own(ellen , moscow , auto(bmw , red , 10000)).
42 own(ellen , moscow , house(adress(moscow , tverskaya , 1 , 1) ,
43                               100000)).
44 own(tom , moscow , auto(mersedes , white , 15000)).
45 own(tom , moscow , ground(48.40338 , 52.17403 , 50000)).
46 own(eric , moscow , auto(mersedes , white , 20000)).
47 own(eric , moscow , house(adress(moscow , pokrovka , 21 , 55) ,

```

```

48                                                                 200000)).
49 own(eric , moscow, ground(28.40338, 32.17403, 500000)).
50 own(eric , moscow, boat(ericary , 500000)).
51 own(mark, moscow, auto(hyundai, silver , 7000)).
52 own(mark, moscow, ground(41.40338, 2.17403, 100000)).
53 own(mark, moscow, boat(mercury , 5000)).
54 own(bill , ekb , auto(hyundai, black , 10000)).
55 own(bill , ekb , house(adress(ekb, lenin , 12, 88), 50000)).
56 own(bill , spb , auto(volvo , black , 13000)).
57 own(bill , spb , house(adress(spb, sadovaya , 1, 12), 100000)).
58
59 % Task1
60 owns(Name, City , Types) :-
61     own(Name, City , house(_, _)), Types = house.
62 owns(Name, City , Types) :-
63     own(Name, City , auto(_, _, _)), Types = auto.
64 owns(Name, City , Types) :-
65     own(Name, City , ground(_, _ ,_)), Types = ground.
66 owns(Name, City , Types) :-
67     own(Name, City , boat(_, _)), Types = boat.
68
69 % Task2
70 ownsPrice(Name, City , Types, Prices) :-
71     own(Name, City , house(_, Prices)), Types = house.
72 ownsPrice(Name, City , Types, Prices) :-
73     own(Name, City , auto(_, _, Prices)), Types = auto.
74 ownsPrice(Name, City , Types, Prices) :-
75     own(Name, City , ground(_, _ , Prices)), Types = ground.
76 ownsPrice(Name, City , Types, Prices) :-
77     own(Name, City , boat(_, Prices)), Types = boat.
78
79 % Task3
80 housePrice(Name, City , Prices) :-
81     own(Name, City , house(_, Prices)) ,!.
82 housePrice(_, _ , Prices) :-
83     Prices = 0.
84
85 groundPrice(Name, City , Prices) :-
86     own(Name, City , ground(_, _ , Prices)) ,!.
87 groundPrice(_, _ , Prices) :-
88     Prices = 0.
89
90 autoPrice(Name, City , Prices) :-

```

```

91     own(Name, City, auto(_ ,_, Prices)),!.
92 autoPrice(_ ,_, Prices) :-
93     Prices = 0.
94
95 boatPrice(Name, City, Prices) :-
96     own(Name, City, boat(_ ,_, Prices)),!.
97 boatPrice(_ ,_, Prices) :-
98     Prices = 0.
99
100 ownsSumPrice(Name, City, Prices) :-
101     housePrice(Name, City, A),
102     groundPrice(Name, City, B),
103     autoPrice(Name, City, C),
104     boatPrice(Name, City, D),
105     Prices = A + B + C + D,
106     Prices > 0.

```


Примеры работы:

1. Названий всех объектов собственности заданного субъекта,

```

1 goal
2   Name = eric ,
3   City = moscow ,
4   owns(Name, City, Types).

```

 [Inactive Y:\Desktop\university\3_course\sem6\Logical_programming\lab5\Obj\goal\$000.exe]

```

Name=eric, City=moscow, Types=house
Name=eric, City=moscow, Types=auto
Name=eric, City=moscow, Types=ground
Name=eric, City=moscow, Types=boat
4 Solutions|


```

2. Названий и стоимости всех объектов собственности заданного субъекта,

```

1 goal
2   Name = bill ,
3   City = spb ,
4   ownsPrice(Name, City, Types, Price).

```

 [Inactive Y:\Desktop\university\3_course\sem6\Logical_programming\lab5\Obj\goal\$000.exe]

```

Name=bill, City=spb, Types=house, Price=100000
Name=bill, City=spb, Types=auto, Price=13000
2 Solutions|


```

3. Разработать правило, позволяющее найти суммарную стоимость всех объектов собственности заданного субъекта.

```

1 goal
2   Name = bill ,
3   City = spb ,
4   ownsSumPrice(Name, City , Price ).

```

 [Inactive Y:\Desktop\university\3_course\sem6\Logical_programming\lab5\Obj\goal\$000.exe]

Price=113000

1 Solution

Приведем таблицу для задания 2.

№ шага	Сравниваемые термы; результат; подстановка, если есть	Дальнейшие действия: прямой ход или откат
1	По <code>ownsPrice(Name, City, Types, Price) = ownsPrice(bill, spb, Types, Price)</code> ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек <code>ownsPrice(bill, spb, Types, Price)</code> , прямой ход
2	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо <code>own(Name, City, house(_, Prices)), Types = house</code> .	Заносится в стек <code>own(bill, spb, house(_, Prices))</code>
3	По <code>own(bill, spb, house(_, Prices))</code> ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено
4	Унификация <code>own(bill, spb, house(_, Prices))</code> с <code>own(ellen, moscow, house(adress(moscow, tverskaya, 1, 1), 100000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
5	Унификация <code>own(bill, spb, house(_, Prices))</code> с <code>own(eric, moscow, house(adress(moscow, pokrovka, 21, 55), 200000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
6	Унификация <code>own(bill, spb, house(_, Prices))</code> с <code>own(bill, ekb, house(adress(ekb, lenin, 12, 88), 50000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)

7	Унификация <code>own(bill, spb, house(_, Prices))</code> с <code>own(bill, spb, house(adress(spb, sadovaya, 1, 12), 100000))</code>	Результат сравнения термов: <code>true</code> , Prices примет значение 100000. Установим маркер. Анонимные переменные не связываются со значением. Переход к следующему целевому утверждению в теле правила (прямой ход)
8	Следующее целевое утверждение <code>Types = house.</code>	Types принимает значение house, вывод результата
9		В базе знаний больше ни одного утверждения с заданным именем, возврат, достаём из стека <code>own(bill, spb, house(_, Prices))</code>
10	Переход к следующему правилу с заданным именем, начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо <code>own(Name, City, auto(_,_, Prices))</code> , <code>Types = auto</code>	Заносится в стек <code>own(bill, spb, auto(_,_, Prices))</code>
11	По <code>own(bill, spb, auto(_, Prices))</code> ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено
12	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(ellen, moscow, auto(bmw, red, 10000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
13	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(tom, moscow, auto(mercedes, white, 15000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
14	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(eric, moscow, auto(mercedes, white, 20000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
15	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(mark, moscow, auto(hyundai, silver, 7000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
16	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(bill, ekb, auto(hyundai, black, 10000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)

17	Унификация <code>own(bill, spb, auto(_, Prices))</code> с <code>own(bill, spb, auto(volvo, black, 13000))</code>	Результат сравнения термов: <code>true</code> , <code>Prices</code> примет значение <code>13000</code> . Установим маркер. Анонимные переменные не связываются со значением. Переход к следующему целевому утверждению в теле правила (прямой ход)
18	Следующее целевое утверждение <code>Types = auto.</code>	<code>Types</code> принимает значение <code>auto</code> , вывод результата
19		В базе знаний больше ни одного утверждения с заданным именем, возврат, достаём из стека <code>own(bill, spb, auto(_, Prices))</code>
20	Переход к следующему правилу с заданным именем, начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо <code>own(Name, City, ground(_, _, Prices))</code> , <code>Types = ground</code>	Заносится в стек <code>own(bill, spb, ground(_, _, Prices))</code>
21	По <code>own(bill, spb, ground(_, Prices))</code> ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено
22	Унификация <code>own(bill, spb, ground(_, Prices))</code> с <code>own(tom, moscow, ground(48.40338, 52.17403, 50000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
23	Унификация <code>own(bill, spb, ground(_, Prices))</code> с <code>own(eric, moscow, ground(28.40338, 32.17403, 500000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
24	Унификация <code>own(bill, spb, ground(_, Prices))</code> с <code>own(mark, moscow, ground(41.40338, 2.17403, 100000))</code>	Результат сравнения термов: <code>false</code> , в базе знаний больше ни одного утверждения с заданным именем, возврат, достаём из стека <code>own(bill, spb, ground(_, Prices))</code>

25	Переход к следующему правилу с заданным именем, начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо <code>own(Name, City, boat(_, Prices)), Types = boat</code>	Заносится в стек <code>own(bill, spb, boat(_,_, Prices))</code>
26	По <code>own(bill, spb, boat(_, Prices))</code> ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено
27	Унификация <code>own(bill, spb, boat(_, Prices))</code> с <code>own(eric, moscow, boat(ericary, 500000))</code>	Результат сравнения термов: <code>false</code> , переход к следующей строке (прямой ход)
28	Унификация <code>own(bill, spb, boat(_, Prices))</code> с <code>own(mark, moscow, boat(mercury, 5000))</code>	Результат сравнения термов: <code>false</code> , в базе знаний больше ни одного утверждения с заданным именем, возврат, достаем из стека <code>own(bill, spb, boat(_, Prices))</code>
29	Больше нет правил с заданным именем, достаем из стека <code>ownsPrice(bill, spb, Types, Price)</code>	Стек пуст, завершение программы

Ответы на вопросы

1. В какой части правила сформулировано знание? Это знание о чем, с формальной точки зрения?

Знания о предметной области выражаются на языке Пролог в виде предложений, называемых утверждениями (clauses).

2. Что содержит тело правила?

Правило содержит несколько целей, которые должны быть истинными для того, чтобы правило было истинным.

`<предикат>:-<предикат>[,<предикат>]*`.

Символ «:-» означает «если». Символ «,>» – это логическая связка «и» или конъюнкция.

3. Что дает использование переменных при формулировании знаний? В чем отличие формулировки знания с помощью термов с одинаковой арностью

при использовании одной переменной и при использовании нескольких переменных?

Предикат факта может содержать переменные в качестве аргументов. Такие факты называются универсальными: они истинны для любых значений переменных.

Например, любит(Х, яблоко)

означает, что любой объект программы "любит яблоко". Универсальные факты сокращают запись программы.

Термы не содержащие переменных называются базовыми термами. Каждый такой терм именуется вполне определенным объектом. Если же терм содержит переменные, то он выступает в качестве представителя целого класса объектов, получающихся при подстановке различных термов вместо переменных.

4. С каким квантором переменные входят в правило, в каких пределах переменная уникальна?

На все переменные в имени предиката наложен квантор всеобщности \forall , на переменные в теле предиката, которые отсутствуют в имени, наложен квантор существования \exists .

Областью действия переменной в Прологе является одно предложение. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов. Исключением является анонимная переменная. Каждая анонимная переменная – это отдельный объект.

5. Какова семантика (смысл) предложений раздела DOMAINS? Когда, где и с какой целью используется это описание?

Домены должны быть определены до их использования.

Раздел описания доменов является аналогом раздела описания типов в императивных языках программирования.

Удобно использовать описание доменов для сокращения имен стандартных доменов.

Например: `i=integer` и далее использовать вместо ключевого слова `integer` односимвольное обозначение `i`.

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

Структура описывается следующим образом:

`<имя структуры>=<имя функтора>(<имя домена первой компоненты>, ... , <имя домена последней компоненты>).`

6. Какова семантика (смысл) предложений раздела PREDICATES? Когда, и где используется это описание? С какой целью?

Предикаты должны быть определены до их использования.

В разделе, озаглавленном зарезервированным словом PREDICATES, содержатся описания определяемых пользователем предикатов. В императивных языках программирования подобными разделами являются разделы описания заголовков процедур и функций.

Описание n-местного предиката имеет следующий вид:

<имя предиката>(<имя домена первого аргумента>, ... , <имя домена n-го аргумента>).

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

Один предикат может иметь несколько описаний.

7. Унификация каких термов запускается на самом первом шаге работы системы? Каковы назначение и результат использования алгоритма унификации?

Пролог берет вопрос и начинает последовательно сверху-вниз сравнивать его с фактами и правилами базы знаний. То есть самыми первыми термами для унификации будет вопрос и первый факт/правило.

Унификация двух термов – это основной шаг доказательства. В процессе работы система выполняет большое число унификаций. **Унификация** – операция, которая позволяет формализовать процесс логического вывода.

Унификация представляет собой процесс сопоставления цели с фактами и правилами базы знаний. Цель может быть согласована, если она может быть сопоставлена с заголовком какого-либо предложения базы.

8. В каком случае запускается механизм отката?

Откат дает возможность получить много решений в одном вопросе к программе.

Во всех точках программы, где существуют альтернативы, в стек заносятся точки возврата.

Если впоследствии окажется, что выбранный вариант не приводит к успеху, то осуществляется откат к последней из имеющихся в стеке точек программы, где был выбран один из альтернативных вариантов.

Выбирается очередной вариант, программа продолжает свою работу. Если все варианты в точке уже были использованы, то регистрируется неудачное завершение и осуществляется переход на предыдущую точку возврата, если такая есть.

При откате все связанные переменные, которые были означены после этой точки, опять освобождаются.