



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 № 9

По дисциплине «Функциональное и логическое программирование»

Студент: Тимонин А. С.

Группа ИУ7-626

Преподаватель Толпинская Н. Б.

Москва.
2020 г.

Практическая часть

Ответить на вопросы (коротко):

1. Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?
2. Какое первое состояние резольвенты?
3. В каких пределах программы переменные уникальны?
4. В какой момент, и каким способом системе удастся получить доступ к голове списка?
5. Каково назначение использования алгоритма унификации?
6. Каков результат работы алгоритма унификации?
7. Как формируется новое состояние резольвенты?
8. Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?
9. В каких случаях запускается механизм отката?
10. Когда останавливается работа системы? Как это определяется на формальном уровне?

Используя хвостовую рекурсию, разработать эффективную программу, (комментируя назначение аргументов), позволяющую:

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка;
3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0) .

Убедиться в правильности результатов

Для одного из вариантов **ВОПРОСА** и одного из **заданий** составить таблицу, отражающую конкретный порядок работы системы: Т.к. резольвента хранится в виде стека, то состояние резольвенты требуется отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового

состояния резольвенты! Для каждого запуска алгоритма унификации, требуется указать № выбранного правила и дальнейшие действия – и почему.

Листинг 1. Реализация программы

```
domains
    mlist = integer*.

predicates
    nondeterm count(mlist, integer).
    nondeterm sum(mlist, integer).
    nondeterm sumMod2(integer, mlist, integer).
    nondeterm evenNum(integer).

clauses
    count([], Count) :- Count = 0, !.
    count([_ | Tail], Count) :- count(Tail, TmpCount),
                                Count = TmpCount+1.

    sum([], Sum) :- Sum = 0, !.
    sum([Head | Tail], Sum) :- sum(Tail, TmpSum),
                                Sum = TmpSum+Head.

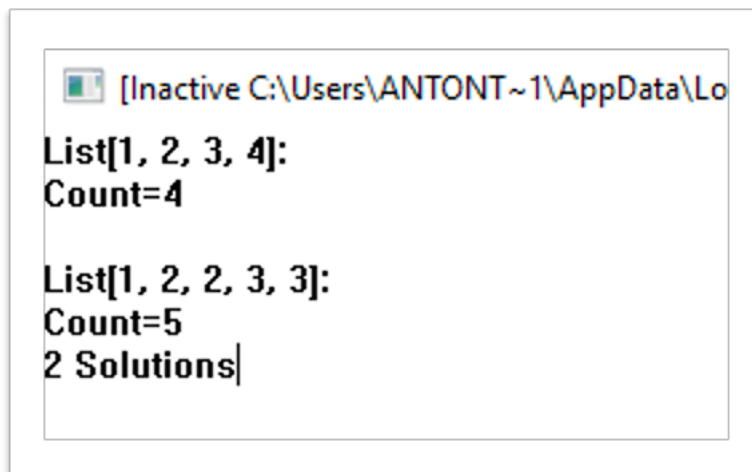
    evenNum(A) :-
        B = A mod 2,
        B = 0,
        !.

    sumMod2(_, [], Sum) :- Sum = 0, !.
    sumMod2(Index, [Head | Tail], Sum) :-
        NewIndex = Index+1,
        evenNum(Index),
        sumMod2(NewIndex, Tail, TmpSum),
        Sum = TmpSum+Head,
        !.
    sumMod2(Index, [_ | Tail], Sum) :- NewIndex = Index+1,
                                        sumMod2(NewIndex, Tail, Sum), !.

goal
    write("List[1, 2, 3, 4]: \n"),
    count([1, 2, 3, 4], Count);
    write("\nList[1, 2, 2, 3, 3]: \n"),
    count([1, 2, 2, 3, 3], Count).

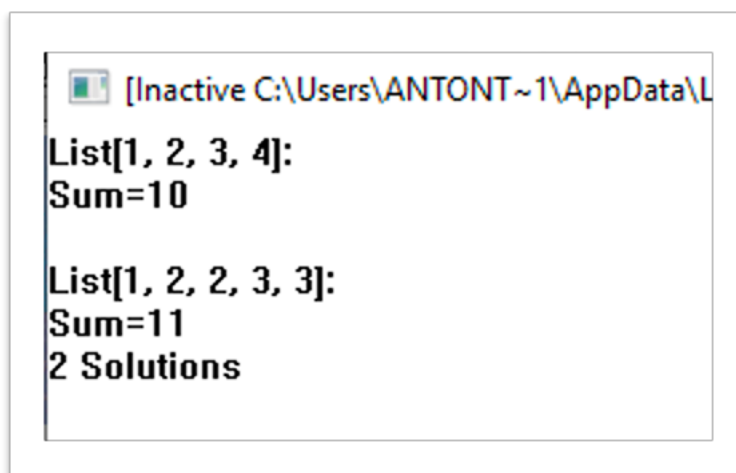
    write("List[1, 2, 3, 4]: \n"),
    sum([1, 2, 3, 4], Sum);
    write("\nList[1, 2, 2, 3, 3]: \n"),
    sum([1, 2, 2, 3, 3], Sum).

    write("Sum of even index numbers[1, 2, 3, 4, 5, 6, 7]: \n"),
    sumMod2(0, [1, 2, 3, 4, 5, 6, 7], SumMod2);
    write("\nSum of even index numbers[1, 0, 1, 0]: \n"),
    sumMod2(0, [1, 0, 1, 0], SumMod2).
```



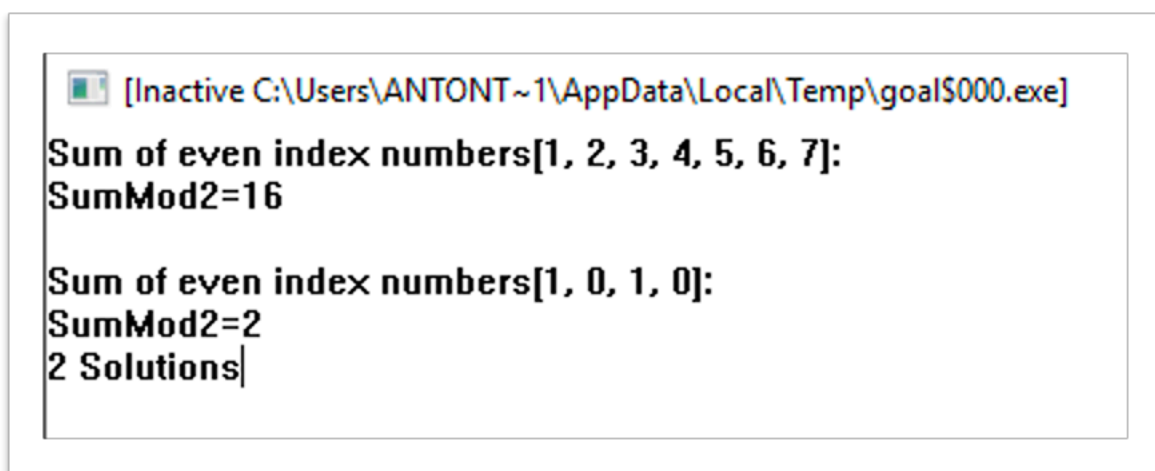
```
[Inactive C:\Users\ANTONT~1\AppData\Lo  
List[1, 2, 3, 4]:  
Count=4  
  
List[1, 2, 2, 3, 3]:  
Count=5  
2 Solutions|
```

Рисунок 1. Тестирование count(mlist, integer)



```
[Inactive C:\Users\ANTONT~1\AppData\L  
List[1, 2, 3, 4]:  
Sum=10  
  
List[1, 2, 2, 3, 3]:  
Sum=11  
2 Solutions
```

Рисунок 2. Тестирование sum(mlist, integer)



```
[Inactive C:\Users\ANTONT~1\AppData\Local\Temp\goal$000.exe]  
Sum of even index numbers[1, 2, 3, 4, 5, 6, 7]:  
SumMod2=16  
  
Sum of even index numbers[1, 0, 1, 0]:  
SumMod2=2  
2 Solutions|
```

Рисунок 3. Тестирование sumMod2(integer, mlist, integer)

Формирование ответа

Таблица 1. count([1, 2, 3, 4], Count)

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат	Дальнейшие действия: прямой ход или откат
1	count([1, 2, 3, 4], Count)	Подстановка mlist = [1, 2, 3, 4], Count = Count	Прямой ход
2	count(Tail, TmpCount) Count = TmpCount+1	Подстановка mlist = [2, 3, 4], Count = TmpCount	Прямой ход
3	count(Tail, TmpCount) Count = TmpCount+1 Count = TmpCount+1	Подстановка mlist = [3, 4], Count = TmpCount	Прямой ход
4	count(Tail, TmpCount) Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1	Подстановка mlist = [4], Count = TmpCount	Прямой ход
5	count(Tail, TmpCount) Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1	Подстановка mlist = [], Count = TmpCount	Прямой ход
6	Count = 0 Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1	Подстановка Count = 0	Прямой ход
7	Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1	Подстановка Count = 1	Прямой ход
8	Count = TmpCount+1 Count = TmpCount+1 Count = TmpCount+1	Подстановка Count = 2	Прямой ход
9	Count = TmpCount+1 Count = TmpCount+1	Подстановка Count = 3	Прямой ход
10	Count = TmpCount+1	Подстановка Count = 4	Прямой ход
11	Пусто	Результат Count = 4	Откат

Таблица 2. sumMod2(0, [1, 0, 1, 0], SumMod2)

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат	Дальнейшие действия: прямой ход или откат
1	sumMod2(0, [1, 0, 1, 0], SumMod2)	Подстановка Index = 0, mlist = [1, 2, 3, 4], Count = Count	Прямой ход
2	NewIndex = Index+1 evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка NewIndex = 1	Прямой ход

3	evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка A = 0	Прямой ход
4	B = A mod 2 B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка B = 0	Прямой ход
5	B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Сравнение B = 0 и 0 Успех	Прямой ход
6	sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка Index = 1, Tail = [0, 1, 0], Sum = TmpSum	Прямой ход
7	NewIndex = Index+1 evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка NewIndex = 2	Прямой ход
8	evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка A = 1	Прямой ход
9	B = A mod 2 B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка B = 1	Прямой ход
10	B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Сравнение B=1 и 0 Неудача	Откат
11	sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка Index = 1, Tail = [0, 1, 0], Sum = TmpSum	Прямой ход
12	NewIndex = Index+1 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка NewIndex = 2	Прямой ход
13	sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head	Подстановка Index = 2, Tail = [1, 0], Sum = Sum	Прямой ход
14	NewIndex = Index+1 evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка NewIndex = 3	Прямой ход
15	evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка A = 2	Прямой ход
16	B = A mod 2 B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка B = 0	Прямой ход
17	B = 0 sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Сравнение B=0 и 0 Успех	Прямой ход
18	sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка Index = 3, Tail = [0], Sum = TmpSum	Прямой ход
19	NewIndex = Index+1 evenNum(Index) sumMod2(NewIndex, Tail, TmpSum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка NewIndex = 4	Прямой ход

20	evenNum(Index) sumMod2(NewIndex, Tail, TmpSun) Sum = TmpSumHead+Head Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка A = 3	Прямой ход
21	B = A mod 2 B = 0 sumMod2(NewIndex, Tail, TmpSun) Sum = TmpSumHead+Head Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка B = 1	Прямой ход
22	B = 0 sumMod2(NewIndex, Tail, TmpSun) Sum = TmpSumHead+Head Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Сравнение B=1 и 0 Неудача	Откат
23	NewIndex = Index+1 sumMod2(NewIndex, Tail, Sum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка NewIndex = 4	Прямой ход
24	sumMod2(Index, Tail, Sum) Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка Index = 4, Tail = [], Sum = Sum	Прямой ход
25	Sum = 0 Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка Sum = 0	Прямой ход
26	Sum = TmpSumHead+Head Sum = TmpSumHead+Head	Подстановка Sum = 1	Прямой ход
27	Sum = TmpSumHead+Head	Подстановка Sum = 2	Прямой ход
28	Пусто		Откат

Теоретическая часть

Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?

Рекурсия – это один из способов организации повторных вычислений. Т.к. логическое программирование – не операторное, то рекурсия – это способ заставить систему использовать многократно одну и ту же процедуру. Но этот процесс рано или поздно надо остановить. Поэтому в рекурсивных процедурах должна быть предусмотрена возможность выхода из рекурсии – специальное предложение процедуры. Эффективный способ организации рекурсии – это хвостовая рекурсия. Хвостовая рекурсия в Пролог организуется при помощи правила, которое обращается к тому же правилу. Кроме этого, повысить эффективность рекурсивной процедуры можно отсекая неперспективные

пути поиска решения. В целях выхода из рекурсии используется предикат отсечения («!»), который, при необходимости, включают в тело некоторых привил.

Какое первое состояние резольвенты?

Простой вопрос

В каких пределах программы переменные уникальны?

Именованные переменные уникальны в рамках предложения, а анонимная переменная – любая уникальна.

В какой момент, и каким способом системе удастся получить доступ к голове списка?

В Prolog существует более общий способ доступа к элементам списка. Для этого используется метод разбиения списка на начало и остаток. Начало списка – это группа первых элементов, не менее одного. Остаток списка – обязательно список (может быть пустой). Для разделения списка на начало, и остаток используется вертикальная черта (|) за последним элементом начала.

Каково назначение использования алгоритма унификации?

Алгоритм унификации связывает переменные из вопроса со значениями, с параметрами, которые содержатся в правилах и фактах.

Каков результат работы алгоритма унификации?

Унификация – операция, которая позволяет формализовать процесс логического вывода. Алгоритм унификации сопоставляет подцель с заданной переменной.

```
1 goal
2     a = man("Anton", Surname).
3     a = man(Name, "Timonin")
```


Например, если в строке 2, в переменной a Name сопоставится с «Anton», а в строке 3, переменная Surname сопоставится с «Timonin». Если бы в строке 2 мы поставили бы фамилию, например, «Konin», тогда строка 3 выдала бы ошибку, так как «Konin» не равно «Timonin». Это связано с тем, что когда параметр в какой-либо переменной занят, он перестает сопоставлять переменные, а начинает их сравнивать.

Как формируется новое состояние резольвенты?

При возврате отменяется последняя уже выполненная редукция (восстанавливается предыдущее состояние резольвенты) и система выполняет ре- конкретизацию переменных, которые были конкретизированы на предыдущем шаге.

Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?

В ходе алгоритма унификации может быть два варианта: унифицируемые термы успешны или нет. Если они успешны:

1. T1 и T2 - одинаковые константы;
2. T1 – не конкретизированная переменная, а T2 - константа или составной терм, не содержащий в качестве аргумента T1. Тогда унификация успешна, причем T1 конкретизируется значением T2.

Тогда применяется подстановка и переменные связываются со значениями.

В каких случаях запускается механизм отката?

Когда в программе возможен выбор нескольких вариантов, Пролог заносит в стек точку возврата, для последующего отката по этой точке возврата.

Пролог унифицирует выбранный вариант, если унификация прошла успешно, тогда пролог подготавливает ответ, и далее по точке возврата происходит унификация с другими вариантами. Если пролог не видит дальнейшие

варианты, которые он мог бы проунифицировать, тогда по точке возврата программа возвращается на еще более раннюю стадию.

Когда останавливается работа системы? Как это определяется на формальном уровне?

Работа интерпретатора завершается либо когда список инструкций опустеет, либо когда произойдет какая-либо ошибка во время выполнения инструкции.