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
по лабораторной работе № 9

Дисциплина: Функциональное и логическое программирование

Студент: Платонова Ольга

Группа: ИУ7-65Б

Преподаватели: Толпинская Н. Б.
Строганов Ю. В.

Москва, 2021 г.

Задание

Используя хвостовую рекурсию, разработать программу, позволяющую найти

1. Длину списка;
2. Сумму элементов числового списка;
3. Сумму элементов числового списка, стоящих на нечетных позициях исходного списка.

Листинг

```
domains
        list = integer*.

predicates
        getLen(list, integer, integer).

clauses
        getLen([], Res, Res) :- !.
        getLen([_ | T], Cntr, Res) :- NCntr = Cntr + 1, getLen(T, NCntr, Res).

goal
        getLen([1, 2, 3], 0, Res).
```

```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=3
1 Solution
```

```
domains
        list = integer*.

predicates
        getSum(list, integer, integer).

clauses
        getSum([], Res, Res) :- !.
        getSum([H | T], Sum, Res) :- NSum = Sum + H, getSum(T, NSum, Res).

goal
        getSum([1, 2, 3, 4], 0, Res).
```

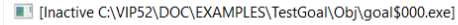
```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=10
1 Solution
```

```
domains
        list = integer*.

predicates
        getSumOdd(list, integer, integer).

clauses
        getSumOdd([], Res, Res) :- !.
        getSumOdd([_ | []], Sum, Res) :- getSumOdd([], Sum, Res).
        getSumOdd([_, B | T], Sum, Res) :- NSum = Sum + B, getSumOdd(T, NSum, Res).

goal
        getSumOdd([1, 2, 3, 4, 5], 0, Res).
```



Res=6
1 Solution

1. getSum([1, 2, 3, 4], 0, Res).

№ шага	Состояние резольвенты и вывод	Для каких термов запускается алгоритм унификации и каков результат	Дальнейшие действия
0	getSum([1, 2, 3, 4], 0, Res).		
1	getSum([1, 2, 3, 4], 0, Res).	T1 = getSum([1, 2, 3, 4], 0, Res). T2 = getSum([], Res, Res). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
2	getSum([1, 2, 3, 4], 0, Res).	T1 = getSum([1, 2, 3, 4], 0, Res). T2 = getSum([H | T], Sum, Res). Успех. Унифицируемые. Подстановка: {H = 1, T = [2, 3, 4], Sum = 0, Res = Res}	Удаляется из стека: getSum([1, 2, 3, 4], 0, Res). Связываются переменные: H = 1, T = [2, 3, 4], Sum = 0
3	NSum = 0 + 1, getSum([2, 3, 4], NSum, Res).	NSum = 0 + 1. NSum = 1	Удаляется из стека: NSum = 0 + 1. Связываются переменные: NSum = 1
4	getSum([2, 3, 4], 1, Res).	T1 = getSum([2, 3, 4], 1, Res). T2 = getSum([], Res, Res). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ

5	getSum([2, 3, 4], 1, Res).	T1 = getSum([2, 3, 4], 1, Res). T2 = getSum([H | T], Sum, Res). Успех. Унифицируемые. Подстановка: {H = 2, T = [3, 4], Sum = 1, Res = Res}	Удаляется из стека: getSum([2, 3, 4], 1, Res). Связываются переменные: H1 = 2, T1 = [3, 4], Sum1 = 1
6	NSum1 = 1 + 2, getSum([3, 4], NSum1, Res1).	NSum1 = 1 + 2. NSum1 = 3	Удаляется из стека: NSum1 = 1 + 2. Связываются переменные: NSum1 = 3
7	getSum([3, 4], 3, Res1).	T1 = getSum([3, 4], 3, Res1). T2 = getSum([], Res, Res). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
8	getSum([3, 4], 3, Res1).	T1 = getSum([3, 4], 3, Res1). T2 = getSum([H | T], Sum, Res). Успех. Унифицируемые. Подстановка: {H = 3, T = [4], Sum = 3, Res = Res1}	Удаляется из стека: getSum([3, 4], 3, Res1). Связываются переменные: H2 = 3, T2 = [4], Sum2 = 3
9	NSum2 = 3 + 3, getSum([4], NSum2, Res2).	NSum2 = 3 + 3. NSum2 = 6	Удаляется из стека: NSum2 = 3 + 3. Связываются переменные: NSum2 = 6
10	getSum([4], 6, Res2).	T1 = getSum([4], 6, Res2). T2 = getSum([], Res, Res). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
11	getSum([4], 6, Res2).	T1 = getSum([4], 6, Res2). T2 = getSum([H | T], Sum, Res). Успех. Унифицируемые. Подстановка: {H = 4, T = [], Sum = 6, Res = Res2}	Удаляется из стека: getSum([4], 6, Res2). Связываются переменные: H3 = 4, T3 = [], Sum3 = 6
12	NSum3 = 6 + 4, getSum([], NSum3, Res3).	NSum3 = 6 + 4. NSum3 = 10	Удаляется из стека: NSum3 = 6 + 4. Связываются переменные: NSum3 = 10
13	getSum([], 10, Res3).	T1 = getSum([], 10, Res3). T2 = getSum([], Res, Res). Успех. Унифицируемые.	Удаляется из стека: getSum([], 10, Res3). Связываются переменные: Res = 10, Res3 = 10

		Подстановка: {[] = [], Res = 10, Res = Res3}	
10	!.	!. Истина.	Удаляется из стека: !.
11	Резольвента пуста.		Выводится Res = 10 Развязываются переменные: Res, Res3 Откат.
12	!.	!. Завершение процедуры.	Удаляется из стека: !. Развязываются переменные: NSum3, H3, T3, Sum3, NSum2, H2, T2, Sum2, NSum1, H1, T1, Sum1, NSum, H, T, Sum
13	Резольвента пуста.		Завершение работы программы.

Вывод

Эффективность работы системы может быть достигнута за счет хвостовой рекурсии и использования отсечения (уменьшения количества унификаций) в тех случаях, когда заведомо известна единственность ответа на вопрос.

1. *Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии в Prolog?*

Рекурсия – определение объекта через ссылку на самого себя. Один из способов организации повторных вычислений. Для организации хвостовой рекурсии необходимо, чтобы рекурсивный вызов был последним в теле рекурсивного правила, и не оставалось других точек выбора. Выход из рекурсии

осуществляется либо достижением базиса рекурсии, либо условием в теле правила.

2. *Какое первое состояние резольвенты?*

Исходная резольвента содержит вопрос.

3. *В каких пределах программы уникальны переменные?*

Именованные переменные уникальны в рамках предложения, анонимные – любые уникальны.

4. *В какой момент, и каким способом системе удается получить доступ к голове списка?*

Получить доступ к голове списка можно при его унификации со списком вида [H | T], где H - голова, T - хвост.

5. *Каково назначение использования алгоритма унификации?*

Алгоритм унификации необходим для того, чтобы подобрать знание, чтобы ответить на поставленный вопрос.

6. *Каков результат работы алгоритма унификации?*

Результатом работы алгоритма является значение переменной «неудача». Если неудача = 1, то унификация невозможна; если неудача = 0, то унификация прошла успешно, а побочным действием работы алгоритма является содержимое результирующей ячейки – результирующая подстановка.

7. *Как формируется новое состояние резольвенты?*

Резольвента меняется в 2 этапа:

   a. Редукция (замена вопроса на тело правила, заголовок которого был успешно унифицирован);
   b. Применение подстановки.

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

В результате подстановки связываются переменные, которые еще не были связаны. После связывания всех утверждений, будет напечатано значение связанных переменных.

9. *В каких случаях запускается механизм отката?*

В случае, когда унификация на текущем шаге завершается тупиковой ситуацией, или был получен ответ «да».

10. *Когда останавливается работа системы? Как это определяется на формальном уровне?*

Когда резольвента пуста и все указатели находятся в конце БЗ.