|  |  |
| --- | --- |
| 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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Лабораторная работа №19*

*По предмету: «Функциональное и логическое программирование»*

Преподаватель: Строганов Ю.В.

Студент: Мирзоян С.А.,

Группа: ИУ7-65Б

Москва, 2020 г.

# Задание

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

* Найти длину списка (по верхнему уровню);
* Найти сумму элементов числового списка
* 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.

# Теоретические вопросы

**Что такое рекурсия?**

Рекурсия – это ссылка на описываемый объект при описании объекта.

**Как организуется хвостовая рекурсия в Prolog?**

* Рекурсивный вызов единственен и расположен в конце тела правила
* Не должно быть возможности сделать откат до вычисления рекурсивного вызова

**Как организовать выход из рекурсии в Prolog?**

С помощью отсечения («!»).

**Какое первое состояние резольвенты?**

Исходный вопрос

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

Получить голову или хвост списка можно при унификации списка с [HEAD|TAIL], HEAD – голова списка, TAIL – хвост списка.

**Каково назначение и результат использования алгоритма унификации?**

Унификация – механизм логического вывода.

Результат – подстановка.

**В каких пределах программы переменные уникальны?**

Именованные - в рамках использующего предложения;

Анонимные - всегда уникальны.

**Как формируется новое состояние резольвенты?**

С помощью редукции. Редукцией цели G с помощью программы P называется замена цели G телом того правила из P, заголовок которого унифицируется с целью. Новая резольвента образуется в два этапа:

1. в текущей резольвенте выбирается одна из подцелей и для неё выполняется редукция;
2. к полученной конъюнкции целей применяется подстановка, полученная как наибольший общий унификатор цели и заголовка сопоставленного с ней правила.

**Как применяется подстановка, полученная с помощью алгоритма унификации? Как глубоко?**

Подстановка применяется к целям в резольвенте путем замены текущей переменной на соответствующий терм. В результате применения подстановки некоторые переменные конкретизируются значениями, которые (значения) могут использованы при доказательстве истинности выбранного правила.

**В каких случаях запускается механизм отката?**

При неудаче алгоритма унификации.

**Когда останавливается работа системы?**

Когда найдены все возможные ответы на вопрос.

**Как это определяется на формальном уровне?**

Когда в резольвенте находится вершина дерева поиска (т.е исходный вопрос, для которого пройдена вся БЗ).

# Листинг

1. domains
2. lst = integer\*.
3. predicates
4. len(lst, integer).
5. len(lst, integer, integer).
7. sum(lst, integer).
8. sum(lst, integer, integer).
10. odd(lst, integer).
11. odd(lst, integer, integer).
13. clauses
14. len(Lst, Res) :- len(Lst, 0, Res).
15. len([], Res, Res) :- !. %empty list - stop
16. len([\_|Tail], Cur, Res) :-  CurLen = Cur + 1, len(Tail, CurLen, Res).
18. sum(Lst, Res) :- sum(Lst, 0, Res).
19. sum([], Res, Res) :- !. % empty list
20. sum([Head|Tail], Cur, Res) :- Sum = Cur + Head, sum(Tail, Sum, Res).
22. odd(Lst, Res) :- odd(Lst, 0, Res).
23. odd([], Res, Res) :- !. %empty list
24. odd([\_], Res, Res) :- !. %nification imposible
25. odd ([\_|[Head|Tail]], Cur, Res) :- NewS = Cur + Head, odd(Tail, NewS, Res).
27. goal
28. %len([0, 1, 2, 3], Res).
29. %sum([-1, -1, 10], Res).
30. %odd([1, 2, 1, 2,], Sum).

# Таблицы

**Вопрос: Res([0,], Res)**

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Текущая резольвента – ТР | ТЦ, выбираемые правила: сравниваемые термы,  подстановка | Дальнейшие действия с комментариями |
| 1 | len([0], Res) | len([0], Res) =  len(Lst, Res)  Успех  Lst = [0] Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 2 | len(Lst, 0, Res) | len([0], 0, Res) = len([], Res, Res)  Неудача | Переход к следующему предложению |
| 3 | len(Lst, 0, Res) | len([0], 0, Res) = len([\_|Tail], Cur, Res)  Успех  T = [] Cur = 0 Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 4 | CurLen = Cur +1 len(Tail, NewRes, Res) | NewRes = 0 +1 = 1 | Прямой ход. |
| 5 | len(Tail, NewRes, Res) | len([], 1, Res) = len([], Res, Res)  Успех  Res = Res = 1 | Прямой ход.  Тело правила заносится в резольвенту. |
| 6 | ! |  | Резольвента пуста.  Найдено решение:  Res = 1  Ввиду отсечения не будет попыток найти другие решения len([], 1, Res)  Откат к 1. Конец len арности 2.  Система завершает работу |

**Текст процедуры**

**Вопрос: sum([1], Res)**

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Текущая резольвента – ТР | ТЦ, выбираемые правила: сравниваемые термы,  подстановка | Дальнейшие действия с комментариями |
| 1 | sum([1], Res) | sum([1], Res) =  sum(Lst, Res)  Успех  Lst = [1] Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 2 | sum(Lst, 0, Res) | sum([1], 0, Res) = len([], Res, Res)  Неудача | Переход к следующему предложению |
| 3 | sum(Lst, 0, Res) | sum([1], 0, Res) = sum([HEAD|TAIL], Cur, Res)  Успех  Head = 1 Tail = [] Cur = 0 Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 4 | Sum = Cur + Head sum(Tail, Sum, Res) | Sum = 0 +1 = 1 | Прямой ход. |
| 5 | sum(Tail, Sum, Res) | sum([], 1, Res) = sum([], Res, Res)  Успех  Res = Res = 1 | Прямой ход.  Тело правила заносится в резольвенту. |
| 6 | ! |  | Резольвента пуста.  Найдено решение:  Res = 1  Ввиду отсечения не будет попыток найти другие решения sum([], 1, Res)  Откат к 1. Конец sum арности 2.  Система завершает работу |

**Текст процедуры**

**Вопрос: odd([1, 2], Res)**

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Текущая резольвента – ТР | ТЦ, выбираемые правила: сравниваемые термы,  подстановка | Дальнейшие действия с комментариями |
| 1 | Odd([1, 2], Res) | Odd([1, 2], Res) =  Odd (Lst, Res)  Успех  Lst = [1, 2] Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 2 | Odd(Lst, 0, Res) | Odd([1, 2], 0, Res) = Odd (Lst, Res)  Неудача | Переход к следующему предложению |
| 3 | Odd(Lst, 0, Res) | Odd([1, 2], 0, Res) = Odd ([], Res, Res)  Неудача | Переход к следующему предложению |
| 4 | Odd(Lst, 0, Res) | Odd([1, 2], 0, Res) = Odd ([\_], Res, Res)  Неудача | Переход к следующему предложению |
| 5 | Odd(Lst, 0, Res) | Odd([1, 2], 0, Res) = Odd ([\_|[HEAD|TAIL]], Cur, Res)  Успех  H = 2 T = [] Cur = 0 Res = Res | Прямой ход.  Тело правила заносится в резольвенту. |
| 6 | NewS = Cur +Head Odd(Tail, NewS, Res) | NewS = 0 + 2 = 2 | Прямой ход. |
| 7 | Odd(Tail, NewS, Res) | Odd([], 2, Res) = Odd (Lst, Res)  Неудача | Переход к следующему предложению |
| 8 | Odd(Tail, NewS, Res) | Odd([], 2, Res) = Odd ([], Res, Res)  Успех.  Res = Res = 2 | Прямой ход.  Тело правила заносится в резольвенту. |
| 9 | ! |  | Резольвента пуста.  Найдено решение:  Res = 2  Ввиду отсечения не будет попыток найти другие решения Odd([], 2, Res)  Откат к 5. Конец Odd арности 3.  Откат к 1. Конец Odd арности 2.  Система завершает работу |