|  |  |
| --- | --- |
| 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 №18*

*По предмету: «Функциональное и логическое программирование»*

Преподаватель: Строганов Ю.В.

Студент: Гасанзаде М.А.,

Группа: ИУ7-66Б

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.

# 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?**

Заданный вопрос (goal).

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

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

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

Унификация – механизм логического вывода. Результат – подстановка.

**В каких пределах программы переменные уникальны?**

Именованная переменная уникальна в рамках предложения, в котором она используется. Анонимные переменные всегда уникальны.

**Как формируется новое состояние резольвенты?**

Преобразования резольвенты выполняются 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 и будут далее использованы при доказательстве истинности тела выбранного правила.

**В каких случаях запускается механизм отката?**

Механизм отката запустится в случае неудачи алгоритма унификации.

**Когда останавливается работа системы?**

Работа системы останавливается, когда найдены все возможные ответы на вопрос.

**Как это определяется на формальном уровне?**

Когда в резольвенте находится исходный вопрос, для которого пройдена вся БЗ.

# ЛИСТИНГ

domains

list = **integer**\*.

predicates

listLen(list, **integer**).

listLen(list, **integer**, **integer**).

listSum(list, **integer**).

listSum(list, **integer**, **integer**).

sumOdd(list, **integer**).

sumOdd(list, **integer**, **integer**).

clauses

listLen(List, Len) :- listLen(List, 0, Len).

listLen([], Len, Len) :- !.

listLen([\_|T], CurLen, Len) :-

NewLen = CurLen + 1,

listLen(T, NewLen, Len).

listSum(List, Sum) :- listSum(List, 0, Sum).

listSum([], Sum, Sum) :- !.

listSum([H|T], CurSum, Sum) :-

NewSum = CurSum + H,

listSum(T, NewSum, Sum).

sumOdd(List, Sum) :- sumOdd(List, 0, Sum).

sumOdd([], Sum, Sum) :- !.

sumOdd([\_], Sum, Sum) :- !.

sumOdd([\_|[H|T]], CurSum, Sum) :-

NewSum = CurSum + H,

sumOdd(T, NewSum, Sum).

goal

%listLen([0, 1, -2, 10], Len).

%listSum([0, -2, 10], Sum).

%sumOdd([1, 2, 1, 2, 1], Sum).

%sumOdd([1, 2, 1, 2], Sum).

Эффективность достигнута за счет использования хвостовой рекурсии и использования отсечения.

# ТАБЛИЦА

**Текст процедуры**

listLen(List, Len) :- listLen(List, 0, Len).

listLen([], Len, Len) :- !.

listLen([\_|T], CurLen, Len) :-

NewLen = CurLen + 1,

listLen(T, NewLen, Len).

**Вопрос: listLen([1,], Len)**

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

**Текст процедуры**

listSum(List, Sum) :- listSum(List, 0, Sum).

listSum([], Sum, Sum) :- !.

listSum([H|T], CurSum, Sum) :-

NewSum = CurSum + H,

listSum(T, NewSum, Sum).

**Вопрос: listSum([1], Sum)**

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Текущая резольвента – ТР | ТЦ, выбираемые правила: сравниваемые термы,  подстановка | Дальнейшие действия с комментариями |
| 1 | listSum([1], Sum) | listSum([1], Sum) =  listSum(List, Sum)  Успех  List = [1] Sum = Sum | Прямой ход.  Тело правила заносится в резольвенту. |
| 2 | listSum(List, 0, Sum) | listSum([1], 0, Sum) = listLen([], Len, Len)  Неудача | Переход к следующему предложению |
| 3 | listSum(List, 0, Sum) | listSum([1], 0, Sum) = listSum([H|T], CurSum, Sum)  Успех  H = 1 T = [] CurSum = 0 Sum = Sum | Прямой ход.  Тело правила заносится в резольвенту. |
| 4 | NewSum = CurSum +H listSum(T, NewSum, Sum) | NewSum = 0 +1 = 1 | Прямой ход. |
| 5 | listSum(T, NewSum, Sum) | listSum([], 1, Sum) = listSum([], Sum, Sum)  Успех  Sum = Sum = 1 | Прямой ход.  Тело правила заносится в резольвенту. |
| 6 | ! |  | Резольвента пуста.  Найдено решение:  Sum = 1  Ввиду отсечения не будет попыток найти другие решения listSum([], 1, Len)  Откат к 1. Конец listSum арности 2.  Система завершает работу |

**Текст процедуры**

sumOdd(List, Sum) :- sumOdd(List, 0, Sum).

sumOdd([], Sum, Sum) :- !.

sumOdd([\_], Sum, Sum) :- !.

sumOdd([\_|[H|T]], CurSum, Sum) :-

NewSum = CurSum + H,

sumOdd(T, NewSum, Sum).

**Вопрос: sumOdd([1, 2], Sum)**

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