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

Дисциплина Функциональное и логическое программирование

Тема Работа программы на Prolog

Студент Куприй А. А.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Толпинская Н. Б., Строганов Ю. В.

Москва, 2020 г.

Цель работы – получить навыки построения модели предметной области, разработки и оформления программы на Prolog, изучить принципы, логику формирования программы и отдельные шаги выполнения программы на Prolog.

Задачи работы: приобрести навыки декларативного описания предметной области с использованием фактов и правил.

Изучить способы использования термов, переменных, фактов и правил в программе на Prolog, принципы и правила сопоставления и отождествления, порядок унификации.

Задание:

Используя хвостовую рекурсию, разработать эффективную программу, 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.

Вопросы:

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

Рекурсия — это ссылка на определяемый объект во время его определения. В языке Prolog рекурсия организуется при помощи правила, в котором есть обращение к тому же правилу. Выход из рекурсии в Prolog организуется при помощи отсечения.

Какое первое состояние резольвенты?

Заданный вопрос (goal).

В каких пределах программы переменные уникальны?

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

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

Во время выполнения алгоритма унификации системе удается получить доступ к голове списка. Во время унификации система пытается разделить список на «начало» и «конец», чтобы унификация была успешна.

Каково назначение использования алгоритма унификации?

Для поиска ответа на вопрос системе необходимо найти подходящее знание в БЗ, для поиска такого знания используется алгоритм унификации. Формально, он помогает системе понять, что заголовок подошел: алгоритм попарно пытается сопоставить термы (текущую цель и термы из БЗ) и построить для них общий пример (для этого используется подстановка).

Каков результат работы алгоритма унификации?

Алгоритм унификации может завершиться «успехом» и «неудачей». В случае успеха результирующая ячейка будет содержать подстановку (наиболее общий унификатор).

Как формируется новое состояние резольвенты?

Преобразования резольвенты выполняются с помощью редукции – замены текущей цели на тело найденного в программе правила (с помощью унификации текущей цели и заголовка правила программы).

Преобразование резольвенты разделено на два этапа:

1. Берется верхняя из подцелей резольвенты (по стековому принципу) и заменяется на тело правила, найденного в программе.
2. Затем к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор цели и сопоставленного с ней правила).

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

Подстановкой называется множество пар, вида: { Xi= ti } , где Xi – переменная, а ti – терм. Т.е происходит конкретизация переменной термом. Применение подстановки заключается в замене каждого вхождения переменной Xi на соответствующий терм (ti). В результате применения подстановки переменные конкретизируются значениями, которые будут далее использованы при доказательстве истинности тела выбранного правила, то есть значения переменных переходят на следующих шаг доказательства.

В каких случаях запускается механизм отката?

Механизм отката запустится в случае неудачи алгоритма унификации.

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

Система завершает работу, в случае если метка расположена в конце процедуры(которая доказывается для ответа на поставленный вопрос) и не осталось альтернатив (для каждой подцели были найдены все возможные наборы значений и везде были проставлены метки), либо если ответ не был найден, но были просмотрены все возможные варианты.

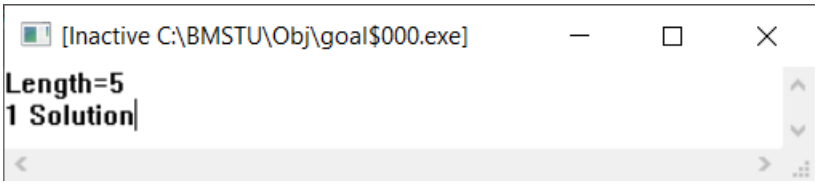
Текст программы

```
domains
        list = integer*.

predicates
        length(list, integer).
        sum(list, integer).
        sumOdd(list, integer).
        sumOdd(list, integer, integer).

clauses
        length([], 0) :- !.
        length([_|Tail], Length) :-
                length(Tail, TailLength),
                Length = TailLength + 1.

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

        sumOdd(List, Sum) :- sumOdd(List, Sum, 0).
        sumOdd([], 0, _) :- !.
        sumOdd([_|Tail], Sum, Index) :-
                Index mod 2 = 0,
                NextIndex = Index + 1,
                sumOdd(Tail, Sum, NextIndex).
        sumOdd([Head|Tail], Sum, Index) :-
                Index mod 2 = 1,
                NextIndex = Index + 1,
                sumOdd(Tail, TailSum, NextIndex),
                Sum = TailSum + Head.

goal
        %length([1, 2, 3, 4, 5], Length).
        %sum([1, 2, 3, 4, 5, 6], Sum).
        sumOdd([1, 2, 1, 2, 1, 2], SumOdd).
```

Результаты работы программы:

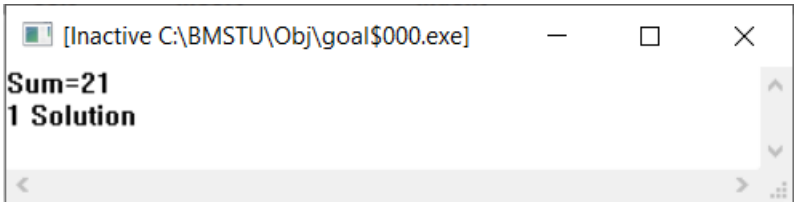
goal

length([1, 2, 3, 4, 5], Length).



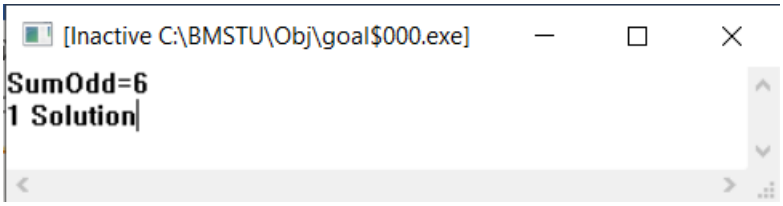
goal

sum([1, 2, 3, 4, 5, 6], Sum).



goal

sum_odd([1, 2, 1, 2, 1, 2], SumOdd).



Таблицы

Цель:
goal
length([1, 2, 3, 4, 5], Length).

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	length([1, 2, 3, 4, 5], Length)	Подстановка: Tail = [2, 3, 4, 5], Length = Length length([1, 2, 3, 4, 5], Length) length([_|Tail], Length)	Прямой ход
2	length(Tail, TailLength) Length = TailLength + 1	Подстановка: Tail = [3, 4, 5], Length = TailLength length([2, 3, 4, 5], TailLength)	Прямой ход
3	length(Tail, TailLength) Length = TailLength + 1 Length = TailLength + 1	Подстановка: Tail = [4, 5], Length = TailLength length([3, 4, 5], TailLength) length([_|Tail], TailLength)	Прямой ход
4	length(Tail, TailLength) Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Tail = [5], Length = TailLength length([4, 5], TailLength) length([_|Tail], Length)	Прямой ход
5	length(Tail, TailLength) Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Tail = [], Length = TailLength length([5], TailLength) length([_|Tail, Length], 0)	Прямой ход
6	length(Tail, TailLength) Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Сравнение: [] и [] length([], TailLength) length([], 0)	Прямой ход

	Length = TailLength + 1		
7	length([], 0) Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Taillength = 0	Прямой ход
8	Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Length = 1	Прямой ход
9	Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Length = 2	Прямой ход
10	Length = TailLength + 1 Length = TailLength + 1 Length = TailLength + 1	Подстановка: Length = 3	Прямой ход
11	Length = TailLength + 1 Length = TailLength + 1	Подстановка: Length = 4	Прямой ход
12	Length = TailLength + 1	Подстановка: Length = 5	Прямой ход
13	Пусто	Результат: Length = 5	Обратный ход