Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 19

Обработка списков на Prolog

Дисциплина	Функциональное и логическое программирование
Студент	Сиденко А.Г.
Группа	ИУ7-63Б
Преподаватель	Толпинская Н.Б., Строганов Ю.В.

Москва, 2020 г.

Задание

Используя хвостовую рекурсию, разработать эффективную программу, 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.

Программа

```
domains
  list = integer*
  length, summa = integer
predicates
  length(list, length).
  sum(list, summa).
  odd_sum(list, summa).
clauses
  length([], 0):-!.
  length([_|Tail], Length):-
    length(Tail, TailLength),
    Length = TailLength + 1.

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

  odd_sum([], 0):-!.
  odd_sum([_], 0):-!.
  odd_sum([_, Second|Tail], Sum):-
    odd_sum(Tail, TailSum),
    Sum = TailSum + Second.
```

Приведем таблицу для нахождения длины.

```
goal
  length([1,2,4], Length).
```

№ шага	Состояние резольвенты	Сравниваемые термы; результат; подстановка, если есть	Дальнейшие действия: прямой ход или откат
1	length([1,2,4], Length)	По length([1,2,4], Length) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек length([1,2,4], Length), прямой ход
2	length([2,4], TailLength), Length = TailLength + 1	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо length([2,4], TailLength), Length = TailLength + 1	Прямой ход
3	length([2,4], TailLength), Length = TailLength + 1	По length([2,4], Length) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек length([2,4], Length), прямой ход
4	length([4], TailLength), Length = TailLength + 1, Length = TailLength + 1	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо length([4], TailLength), Length = TailLength + 1	Прямой ход
5	length([4], TailLength), Length = TailLength + 1, Length = TailLength + 1	По length([4], Length) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек length([4], Length), прямой ход
6	length([], TailLength), Length = TailLength + 1, Length = TailLength + 1, Length = TailLength + 1	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо length([], TailLength), Length = TailLength + 1	Прямой ход

7	length([], TailLength), Length = TailLength + 1, Length = TailLength + 1, Length = TailLength + 1	По length([], Length) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек length([], Length), прямой ход
8	Length = 0 + 1, Length = TailLength + 1, Length = TailLength + 1	TailLength=0	Успех, достаем из стека length([], Length). Отсечение, больше определения отношения не ищется, переход к следующему целевому утверждению
9	Length = 1 + 1, Length = TailLength + 1	TailLength=1	Успех, достаем из стека length([4], Length). Больше определений с таким именем нет, переход к следующему целевому утверждению
10	Length = 2 + 1	TailLength=2	Успех, достаем из стека length([2,4], Length). Больше определений с таким именем нет, переход к следующему целевому утверждению
11	Резольента пуста	Length=3	Успех, достаем из стека length([1,2,4], Length). Больше определений с таким именем нет, резольента пуста. Вывод результата. Стек пуст, завершение программы

Вывод

Эффективный способ организации рекурсии – хвостовая рекурсия. Эффективность рекурсивной процедуры повышается благодаря отсечению неперспективных путей поиска решения. Используя «!» – отсечение. Которое сократит количество выполняемых унификаций для достижения максимальной эффективности работы системы.

Ответы на вопросы

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

Рекурсия позволяет использовать в процессе определения предиката его самого.

Хвостовая рекурсия: Для ее осуществления рекурсивный вызов определяемого предиката должен быть последней подцелью в теле рекурсивного правила и к моменту рекурсивного вызова не должно остаться точек возврата (непроверенных альтернатив).

Параметры должны изменяться на каждом шаге так, чтобы в итоге либо сработал базис рекурсии, либо условие выхода из рекурсии, размещенное в самом правиле.

2. Какое первое состояние резольвенты?

   Вопрос.

3. В каких пределах программы уникальны переменные?

   Областью действия переменной в Прологе является одно предложение. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов. Исключением является анонимная переменная. Каждая анонимная переменная – это отдельный объект.

4. В какой момент, и каким способом системе удается 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. Для этого используется вертикальная черта (|) за последним элементом начала.

   Если начало состоит из одного элемента, то получим: голову и хвост

5. Каково назначение использования алгоритма унификации? Каков результат работы алгоритма унификации?

   Пролог выполняет унификацию в двух случаях: когда цель сопоставляется с заголовком предложения или когда используется знак равенства, который является инфиксным предикатом (предикатом, который расположен между своими аргументами, а не перед ними).

   Унификация двух термов – это основной шаг доказательства. В процессе работы система выполняет большое число унификаций. **Унификация** – операция, которая позволяет формализовать процесс логического вывода.

6. Каков результат работы алгоритма унификации?

   Унификация представляет собой процесс сопоставления цели с фактами и правилами базы знаний. Цель может быть согласована, если она может быть сопоставлена с заголовком какого-либо предложения базы.

   Результатом его работы является последняя из построенных подстановок.

7. Как формируется новое состояние резольвенты?

   Резольвента - текущая цель, существующая на любой стадии вычислений. Резольвенты порождаются целью и каким-либо правилом или фактом, которые просматриваются последовательно сверху вниз. Если резольвента существует при наиболее общей унификации, она вычисляется. Если пустая резольвента с помощью такой стратегии не найдена, то ответ на вопрос отрицателен.

8. Как применяется подстановка, полученная с помощью алгоритма унификации?

   При согласовании переменные получают значения, указанные с другой стороны от знака «=», если переменные еще не были связаны. Переменные становятся связанными и после успешного согласования всех целевых утверждений, будет напечатано значение связанных переменных.

9. В каких случаях запускается механизм отката?

   Откат дает возможность получить много решений в одном вопросе к программе.

   Во всех точках программы, где существуют альтернативы, в стек заносятся точки возврата.

   Если впоследствии окажется, что выбранный вариант не приводит к успеху, то осуществляется откат к последней из имеющихся в стеке точек программы, где был выбран один из альтернативных вариантов.

   Выбирается очередной вариант, программа продолжает свою работу. Если все варианты в точке уже были использованы, то регистрируется неудачное завершение и осуществляется переход на предыдущую точку возврата, если такая есть.

   При откате все связанные переменные, которые были означены после этой точки, опять освобождаются.

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

    Когда стек пуст.