Федеральное государственное бюджетное образовательное учреждение высшего профессионального 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»

Отчет по лабораторной работе №19

по курсу:

«Функциональное и Логическое программирование»

Студент группы ИУ7-63Б: Фурдик Н. О.
(Фамилия И.О.)
Преподаватель: Толпинская Н. Б., Строганов Ю. В.
(Фамилия И.О.)

Москва, 2020

Оглавление

Постановка задачи

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

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.

Листинг программы

Ниже представлен листинг программы:

```
domains

list = integer*.

predicates

len(list, integer length).
sum(list, integer sum).
odd_sum(list, integer sum, integer index).

clauses

len([], 0) :-!.
len([_|Tail], Length) :-len(Tail, Tail_Length), Length = Tail_Length + 1.

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

odd_sum([], 0, _) :-!.
odd_sum([Head|Tail], Sum, Index) :-Index mod 2 = 1, Next_Index = Index + 1,
                          odd_sum(Tail, Sum_, Next_Index), Sum = Sum_ + Head.
odd_sum([_|Tail], Sum, Index) :-Index mod 2 = 0, Next_Index = Index + 1,
                          odd_sum(Tail, Sum, Next_Index).



goal

%len of list
len([1,21,3,4,5], Length).

%Length=5
%1 Solution


%sum of numbers
sum([1,21,3,4,5], Sum).

%Sum=34
%1 Solution


%sum of odd
odd_sum([1,21,3,4,5], Sum, 0).

%Sum=25
%1 Solution
```

Листинг 1: Код программы

Описание работы системы

Ниже представлен алгоритм поиска ответов на вопрос len([1,2,3], Length).

Таблица 1: Описание работы системы при поиске ответа на вопрос

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	**ТР: len([1,2,3], Length)**	ПРІ: [] = [1, 2, 3] ПРІІ: [_|Tail] = [1, 2, 3]	ПРІ: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРІІ: успех (подобрано знание) => {Tail = [2, 3]}, проверка тела ПРІІ
2	**ТР: len([2,3], Tail_Length)** Length = Tail_Length1 + 1	ПРІ: [] = [2, 3] ПРІІ: [_|Tail] = [2, 3]	ПРІ: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРІІ: успех (подобрано знание) => {Tail = [2, 3]}, проверка тела ПРІІ
3	**ТР: len([3], Tail_Length)** Tail_Length = Tail_Length + 1 Length = Tail_Length1 + 1	ПРІ: [] = [3] ПРІІ: [_|Tail] = [3]	ПРІ: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРІІ: успех (подобрано знание) => {Tail = [3]}, проверка тела ПРІІ
4	**ТР: len([], Tail_Length)** Tail_Length = Tail_Length + 1 Tail_Length = Tail_Length + 1 Length = Tail_Length1 + 1	ПРІ: [] = []	ПРІ: успех (подобрано знание) => пустое тело заменяет цель в резольвенте
5	**ТР: пусто** Tail_Length = 0 Tail_Length = Tail_Length + 1 Tail_Length = Tail_Length + 1 Length = Tail_Length1 + 1	Length = 3	успех – однократный ответ – «Да», метка – на ПРІ. Отказ от найденного значения (откат), возврат к предыдущему состоянию резольвенты
6	**ТР: len([])**	ПРІІ: [_|Tail] = []	поиск знания от метки ниже унификация невозможна =>неудача, надо включить откат, но метка (метки) в конце процедуры – других альтернатив нет => система завершает работу с единственным результатом – «Да».
7	**Вывод:**	Length = 3	

Ответы на вопросы

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

   Рекурсия – это один из способов организации повторных вычислений.

   Для осуществления хвостовой рекурсии 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?**

   Изначально в резольвенте находится вопрос.

3) **В каких пределах программы уникальны переменные?**

   Именованные переменные уникальны в рамках одного предложения. Анонимная переменная уникальна всегда. Переменные предназначены для передачи значений «во времени и в пространстве».

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

   В момент подстановки, если использовалась конструкция [Head|Tail], где Head - голова списка.

5) **Каково назначение использования алгоритма унификации?**

   Система запускает унификацию в том случае, если ей был задан вопрос. Унификация вопроса и первого предложения базы знаний происходит на первом шаге работы программы.

   Алгоритм унификации необходим для попытки "увидеть одинаковость"– сопоставимость двух термов, может завершаться успехом или тупиковой ситуацией.

6) **Каков результат работы алгоритма унификации?**

   Результат алгоритма унификации – ответ «да» или «нет».

7) **Как формируется новое состояние резольвенты?**

На каждом шаге имеется некоторая совокупность целей - утверждений, истинность (выводимость) которых надо доказать. Эта совокупность называется резольвентой - её состояние меняется в процессе доказательства (Для хранения резольвенты система использует стек). Новая резольвента образуется в два этапа:

- в текущей резольвенте выбирается одна из подцелей (по стековому принципу - верхняя) и для неё выполняется редукция - замена подцели на тело найденного (подобранного, если удалось) правила (а как подбирается правило?),
- затем, к полученной конъюнкции целей применяется подстановка, полученная как наибольший общий унификатор цели (выбранной) и заголовка сопоставленного с ней правила.

8) **Как применяется подстановка, полученная с помощью алгоритма унификации?**

Пока стек не пуст – **цикл**:

- считать из стека в рабочую область очередное равенство S=T
- обработать считанное по правилам:
  - если S и T несовпадающие константы, то неудача=1, и выход из цикла
  - если одинаковые константы то следующий шаг цикла
  - если S переменная и T терм содержащий S, то неудача=1, и выход из цикла
  - если S переменная и T терм НЕ содержащий S, то отыскать в стеке и в результирующей ячейке все вхождения S и заменить на T. Добавить в результирующую ячейку равенство S=T. Следующий шаг цикла
  - если S и T составные термы с разными функторами или разными арностями, то неудача=1, выход из цикла
  - если S и T составные термы с одинаковыми функторами и арностью: $S = f(s_1, s_2..., s_m)$; $T = f(t_1, t_2..., t_m)$, то занести в стек равенство

$S_1 = T_1, S_2 = T_2 ... S_m = T_m.$

- очистить рабочее поле

– **конец цикла**

9) **В каком случае запускается механизм отката?**

Механизм отката запускается в 2 случаях:

- Если алгоритм попал в тупиковую ситуацию.
- Если резольвента не пуста и решение найдено, но в базе знание остались не отмеченные предложения.

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

Работа системы останавливается в двух случаях:

- когда встретился символ отсечения (!);
- когда резольвента осталась пустой (формально не осталось подходящих фактов и правил).

Литература

1. Толпинская Н.Б. - Курс лекций по "Функциональному и Логическому программированию"[Текст], Москва 2020 год.

2. Анатолий Адаменко, Андрей Кучуков. Логическое программирование и Visual Prolog (с CD). — СПб.: БХВ-Петербург, 2003. — 990 с. — ISBN 5-94157-156-9.