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 №20

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, комментируя аргументы, эффективную программу, позволяющую:

1) Сформировать список из элементов числового списка, больших заданного значения;

2) Сформировать список из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0);

3) Удалить заданный элемент из списка (один или все вхождения);

4) Преобразовать список в множество (можно использовать ранее разработанные процедуры).

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

more_than(list, integer length, list new_list).
odd_elems(list, integer index, list new_list).
delete_elem(list, integer elem, list new_list).
get_set(list, list).
clauses

more_than([], _, []) :-!.
more_than([Head|Tail], Number, New_List) :-Head > Number,
                        more_than(Tail, Number, List_), New_List = [Head|List_].
more_than([Head|Tail], Number, New_List) :-Head <= Number,
                        more_than(Tail, Number, New_List).

odd_elems([], _, []) :-!.
odd_elems([Head|Tail], Index, New_List) :-Index mod 2 = 1, Next_Index = Index + 1,
                        odd_elems(Tail, Next_Index, List_), New_list = [Head|List_].
odd_elems([_|Tail], Index, New_List) :-Index mod 2 = 0, Next_Index = Index + 1,
                        odd_elems(Tail, Next_Index, New_List).

delete_elem([], _, []) :-!.
delete_elem([Head|Tail], Elem, New_List) :-Elem <> Head,
                        delete_elem(Tail, Elem, List_), New_List = [Head|List_].
delete_elem([Head|Tail], Elem, New_List) :-Elem = Head,
                        delete_elem(Tail, Elem, New_List).

get_set([], []) :-!.
get_set([Head|Tail], Set) :-delete_elem(Tail, Head, New_List),
                        get_set(New_List, Set_),Set = [Head|Set_].
goal

%list with elems that are more than
more_than([5,312,1,2,3,2,1,1,6], 1, Result).

%Result=[5,312,2,3,2,6]
%1 Solution


%list with odd elems
odd_elems([1,2,3,4,5,6,7,8,9,10],0, Result).

%Result=[2,4,6,8,10]
%1 SolutionNew_ListNew_List
```

```
%delete all entries
delete_elem([1,2,3,4,1,1],1, Result).

%Result=[2,3,4]
%1 Solution


%get a set from list
get_set([1,1,1,1,2,3,3,3,3,2,2,2,2,1], Result).

%Result=[1,2,3]
%1 Solution
```

Листинг 1: Код программы

Описание работы системы

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

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: more_than([1,2,3], 1, Result)**	ПРI: [] = [1, 2, 3] ПРII: [Head|Tail] = [1, 2, 3] ПРIII: [Head|Tail] = [1, 2, 3]	ПРI: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРII: успех (подобрано знание) => {Head = [1] , Tail = [2, 3]}, проверка тела ПРII, неудача => возврат к ТЦ, метка переносится ниже ПРIII: успех (подобрано знание) => {Head = [1] , Tail = [2, 3]}, проверка тела ПРII
2	**ТР: more_than([2,3], 1, New_List)** New_List = New_List	ПРI: [] = [2, 3] ПРII: [Head|Tail] = [2, 3]	ПРI: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРII: успех (подобрано знание) => {Head = [2], Tail = [3]}, проверка тела ПРII
3	**ТР: more_than([3], 1, New_List)** New_List = [2|[New_List]] New_List = New_List	ПРI: [] = [3] ПРII: [Head|Tail] = [3]	ПРI: унификация невозможна => возврат к ТЦ, метка переносится ниже ПРII: успех (подобрано знание) => {Head = [3]}, проверка тела ПРII
4	**ТР: more_than([], 1, New_List)** New_List = [3|[New_List]] New_List = [2|[New_List]] New_List = New_List	ПРI: [] = []	ПРI: успех (подобрано знание) => пустое тело заменяет цель в резольвенте
5	**ТР: пусто** New_List = [] New_List = [3|[New_List]] New_List = [2|[New_List]] New_List = New_List	New_List = [2 |[3|[|[]]]] = [2, 3]	успех – однократный ответ – «Да», метка – на ПРI. Отказ от найденного значения (откат), возврат к предыдущему состоянию резольвенты
6	**ТР: New_List([])**	ПРII: [Head|Head] = [] ПРIII: [Head|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:**	Result = [2, 3]	

Ответы на вопросы

1) **Как организуется хвостовая рекурсия в Prolog?**

   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).

2) **Какое первое состояние резольвенты?**

   Изначально в резольвенте находится вопрос.

3) **Каким способом можно разделить список на части, какие, требования к частям?**

   Список можно разбить на начало и остаток. Начало списка – это группа первых элементов, не менее одного. Остаток списка – обязательно список (может быть пустой). Для разделения списка на начало, и остаток используется вертикальная черта (|) за последним элементом начала. Остаток это всегда один (простой или составной) терм.

4) **Как выделить за один шаг первые два подряд идущих элемента списка? Как выделить 1-й и 3-й элемент за один шаг?**

   Для этого нужно связать их с переменными: [First, Second|_], тогда First будет обозначать первый элемент списка, а Second – второй).
   Соответственно, конструкция [First, _, Third|_] используется, чтобы выделить 1-й и 3-й элемент за один шаг (First – первый элемент списка, Third – третий)

5) **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.

6) **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.