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

О т ч е т

по лабораторной работе № 20

Дисциплина: «Функциональное и логическое программирование»

Выполнила: Овчинникова А. П.

Группа: ИУ7-65Б

Преподаватель: Толпинская Н. Б.

Строганов Ю. В.

Москва, 2020

Задание

Ответить на вопросы (коротко):

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 1-ого задания 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 соответствующий вывод: успех или нет –и почему.

Теоретическая часть

1. Как организуется хвостовая рекурсия в Prolog?

Для осуществления хвостовой рекурсии в Prolog рекурсивный вызов определяемого предиката должен быть последней подцелью в теле рекурсивного правила и к моменту рекурсивного вызова не должно остаться точек возврата. То есть у подцелей, расположенных левее рекурсивного вызова определяемого предиката, не должно оставаться каких-то непроверенных вариантов и у процедуры не должно быть предложений, расположенных ниже рекурсивного правила. Пролог распознает хвостовую

рекурсию и устраняет связанные с ней дополнительные расходы. Этот процесс называется оптимизацией хвостовой рекурсии или оптимизацией последнего вызова.

2. Какое первое состояние резольвенты?

Первое состояние резольвенты – заданный вопрос.

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

Для доступа к элементам списка используется метод разбиения списка 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 терм. Если начало состоит из одного элемента, то получим: голову и хвост.

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

Первые два подряд идущих элемента:

[H1, H2|T]

1-й и 3-й:

[H1, _, H3|T]

5. Как формируется новое состояние резольвенты?

Преобразования резольвенты выполняются с помощью редукции. Новая резольвента образуется в два этапа:

- В текущей резольвенте выбирается одна из подцелей (по стековому принципу – верхняя) и для нее выполняется редукция – замена подцели на тело найденного правила (если удалось найти правило).
- Затем к полученной конъюнкции целей применяется подстановка, полученная как наибольший общий унификатор выбранной цели и заголовка сопоставленного с ней правила.

Если для редукции цели из резольвенты был выбран факт из БЗ, то новая резольвента будет содержать в конъюнкции на одну цель меньше. Если задан простой вопрос и подобран для редукции факт, то произойдет немедленное его согласование. А если для простого вопроса подобрано правило, то число целей в резольвенте не уменьшится, т. к. цель будет заменена телом подобранного правила.

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

Работа системы останавливается, когда найдены все возможные варианты ответа на вопрос, то есть резольвента содержит исходный вопрос, а все предложения в БЗ помечены как пройденные.

Успешное завершение работы программы достигается тогда, когда резольвента пуста.

Если резольвента не пуста и нет утверждений в базе знаний, которые удается сопоставить с выделенной в этой вершине подцелью, то работа системы завершается неудачей.

Код программы

Программа 1. Список из элементов числового списка, больших заданного значения.

```
domains

list = integer*.

predicates

bigger_than_list(list, list, integer). % original list, result list, number to compare with

clauses
```

```
bigger_than_list([], [], _). % base
bigger_than_list([H|T1], [H|T2], Min) :- % if current head is more than min
        H > Min,
        bigger_than_list(T1, T2, Min),
        !.
bigger_than_list([_|T1], T2, Min) :- % if current head is not more than min
        bigger_than_list(T1, T2,  Min).

goal

bigger_than_list([], RES, 2).
%bigger_than_list([1, 2, 3, 4], RES, 4).
%bigger_than_list([1, 2, 3, 4], RES, 0).
%bigger_than_list([1, 2, 3, 4], RES, 2).
```

Программа 2. Список из элементов, стоящих на нечетных позициях исходного списка.

```
domains

list = integer*.

predicates

list_odd(list, list). % original list, result list

clauses

list_odd([], []). % base
list_odd([_, H|T1], [H|T2]) :- list_odd(T1, T2), !.
```

```
list_odd([_|T1], T2) :- list_odd(T1, T2).

goal

list_odd([], RES).
%list_odd([1], RES).
%list_odd([1, 2], RES).
%list_odd([1, 2, 3], RES).
%list_odd([1, 2, 3, 4], RES).
```

Программа 3. Удаление заданного элемента из списка.

```
domains

list = integer*.

predicates

delete(list, list, integer). % original list, result list, element to delete

clauses

delete([], [], _).
delete([Elem|T1], T2, Elem) :- delete(T1, T2, Elem), !.
delete([H|T1], [H|T2], Elem) :- delete(T1, T2, Elem).

goal

delete([1, 3, 2, 4, 3, 3], RES, 3).
```

```
%delete([3, 1], RES, 3).
%delete([1, 3], RES, 3).
%delete([1, 2 , 3], RES, 3).
%delete([3], RES, 3).
%delete([1, 2], RES, 3).
%delete([1], RES, 3).
%delete([], RES, 3).
```

Программа 4. Преобразование списка в множество.

```
domains

list = integer*.

predicates

create_set(list, list). % original list, result set
delete(list, list, integer). % original list, result list, element to delete

clauses

delete([], [], _).
delete([Elem|T1], T2, Elem) :- delete(T1, T2, Elem), !.
delete([H|T1], [H|T2], Elem) :- delete(T1, T2, Elem).

create_set([], []).
create_set([H|T1], [H|T2]) :- delete(T1, T3, H), create_set(T3, T2).

goal
```

```
create_set([1, 2, 1, 3, 4, 5, 5, 6, 4], RES).
%create_set([1, 1], RES).
%create_set([1, 2], RES).
%create_set([1], RES).
%create_set([], RES).
```

Примеры работы программы

Программа 1. Список из элементов числового списка, больших заданного значения.

Вопрос	Ответ
bigger_than_list([], RES, 2).	RES=[] 1 Solution
bigger_than_list([1, 2, 3, 4], RES, 4).	RES=[] 1 Solution
bigger_than_list([1, 2, 3, 4], RES, 0).	RES=[1,2,3,4] 1 Solution
bigger_than_list([1, 2, 3, 4], RES, 2).	RES=[3,4] 1 Solution
bigger_than_list([2], RES, 1).	RES=[2] 1 Solution

Порядок работы системы для вопроса *bigger_than_list([2], RES, 1).* приведен в таблице 1.

Таблица 1. Порядок работы системы.

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	bigger_than_list([2], RES, 1).	Попытка унификации: T1= bigger_than_list([2], RES, 1). T2= bigger_than_list([], [], _). Результат: неудача, [2] != []	Откат, переход к следующему предложению
2	bigger_than_list([2], RES, 1).	Попытка унификации: T1= bigger_than_list([2], RES, 1). T2= bigger_than_list([H|T1], [H|T2], Min) :- H > Min, bigger_than_list(T1, T2, Min), !. Результат: успех. Подстановка: {H=2,	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.

		T1=[], RES=[2|T2], Min=1}	
3	2 > 1, bigger_than_list([], T2, 1), !.	ТЦ: 2 > 1, Результат: успех.	Прямой ход. К резольвенте применяется подстановка.
4	bigger_than_list([], T2, 1), !.	Попытка унификации: T1= bigger_than_list([], T2, 1), T2= bigger_than_list([], [], _). Результат: успех. Подстановка: {[]=[], T2=[], _=1}	Прямой ход. К резольвенте применяется подстановка.
5	!.	Выполнение отсечения. Результат: успех.	Резольвента становится пустой.
6	Резольвента пуста	Все переменные связанны.	Попытка отката завершает использование процедуры. Завершение работы программы. RES=[2|[]]. Вывод результата на экран.

Программа 2. Список из элементов, стоящих на нечетных позициях исходного списка.

Вопрос	Ответ
list_odd([], RES).	RES=[] 1 Solution
list_odd([1], RES).	RES=[] 1 Solution
list_odd([1, 2], RES).	RES=[2] 1 Solution
list_odd([1, 2, 3], RES).	RES=[2] 1 Solution
list_odd([1, 2, 3, 4], RES).	RES=[2,4] 1 Solution

Программа 3. Удаление заданного элемента из списка.

Вопрос	Ответ
delete([1, 3, 2, 4, 3, 3], RES, 3).	RES=[1,2,4] 1 Solution
delete([3, 1], RES, 3).	RES=[1]

	1 Solution
delete([1, 3], RES, 3).	RES=[1] 1 Solution
delete([1, 2 , 3], RES, 3).	RES=[1,2] 1 Solution
delete([3], RES, 3).	RES=[] 1 Solution
delete([1, 2], RES, 3).	RES=[1,2] 1 Solution
delete([1], RES, 3).	RES=[1] 1 Solution
delete([], RES, 3).	RES=[] 1 Solution

Программа 4. Преобразование списка в множество.

Вопрос	Ответ
create_set([], RES).	RES=[] 1 Solution
create_set([1], RES).	RES=[1] 1 Solution
create_set([1, 2], RES).	RES=[1,2] 1 Solution
create_set([1, 1], RES).	RES=[1] 1 Solution
create_set([1, 2, 1], RES).	RES=[1,2] 1 Solution
create_set([1, 2, 1, 3, 4, 5, 5, 6, 4], RES).	RES=[1,2,3,4,5,6] 1 Solution

Вывод

Таким образом, в программах эффективность достигается за счет использования хвостовой рекурсии и отсечения.