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 № 20

Тема Формирование и модификация списков на Prolog

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Толпинская Н.Б.

Москва.
2020 г

Цель работы – изучить способы формирования и модификации списков в Prolog, эффективные методы обработки списков и порядок реализации рекурсивных программ.

Задачи работы: приобрести навыки формирования и модификации списков на Prolog, эффективного способа их обработки, организации и прядка работы соответствующих программ.
Изучить особенность использования переменных при обработке списков. Способ формирования и изменения резольвенты в этом случае и порядок формирования ответа.

Задание

Ответить на вопросы (коротко**):**

1. Как организуется хвостовая рекурсия в Prolog?
2. Какое первое состояние резольвенты?
3. Каким способом можно разделить список на части, какие, требования к частям?
4. Как выделить за один шаг первые два подряд идущих элемента списка? Как выделить 1-й и 3-й элемент за один шаг?
5. Как формируется новое состояние резольвенты?
6. Когда останавливается работа системы? Как это определяется на формальном уровне?

Используя хвостовую рекурсию, разработать, <u>комментируя аргументы</u>, 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: <u>вершина – сверху!</u> 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.

Текст процедуры …; Вопрос:…..

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
шаг1	…	…	…
…	…	…	…

Ответы на вопросы (коротко**):**

Как организуется хвостовая рекурсия в Prolog?

Рекурсивный вызов функции должен быть расположен в конце правила. Не должно быть возможности выполнить откат до выхода из рекурсивного правила.

Какое первое состояние резольвенты?

Первое состояние резольвенты - заданный вопрос.

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

Можно получить голову и хвост списка. Это можно сделать при унификации с [H|T]. Хвост обязательно должен являться списком.

Как выделить за один шаг первые два подряд идущих элемента списка?

[H1|[H2|_]]

Как выделить 1-й и 3-й элемент за один шаг?

[H1|[_|[H3|_]]]

Как формируется новое состояние резольвенты?

При изменении строится новая резольвента. По стековому принципу берется верхняя под-цель и заменяется на тело подходящего правила. Затем применяется найденная на текущем этапе подстановка.

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

Завершение работы программы достигается, когда резольвента пуста.

Текст программы

```
domains
  list = integer*.

predicates

  bigger(list, integer, list)

  oddlist(list, list)

  delete(list, integer, list)

  createSet(list, list)

clauses

  bigger([], _ ,[]):-!.
  bigger([H|T], Num, Res) :- H <= Num, bigger(T, Num, Res).
  bigger([H|T], Num, [H|Res]) :- H > Num, bigger(T, Num, Res).

  oddlist([], []):- !.
  oddlist([_], []):- !.
  oddlist([_|[H|T]], [H|Res]):- oddlist(T, Res).
```

```
delete([], _ , []):- !.
delete([H|T], Num, [H|Res]) :- H <> Num, delete(T, Num, Res).
delete([H|T], H, Res):- delete(T,H, Res).

createSet([], []):- !.
createSet([H|T], [H| Res]):- delete(T, H, Tmp), createSet(Tmp, Res).

goal

%bigger([1,2,3,4,5], 2, Res).
%oddlist([0,1,2,3,4,5,6,7,8,9], Res).
%delete([1,2,2,2,2,2,1],2,Res).
createSet([1,2,2,2,2,1,1,1], Res).
```

Примеры работы:
Сформировать список из элементов числового списка, больших заданного значения;

bigger([1,2,3,4,5], 2, Res). → [3,4,5]

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

oddlist([0,1,2,3,4,5,6,7,8,9], Res). → [1,3,5,7,9]
oddlist([0,1,2,3,4,5,6,7,8], Res). → [1,3,5,7]

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

delete([1,2,2,2,2,2,1],2,Res). → [1,1]

Преобразовать список в множество (можно использовать ранее разработанные процедуры).
createSet([1,2,2,2,2,1,1,1], Res). → [1,2]

Текст процедуры

1. bigger([], _ ,[]):-!.
2. bigger([H|T], Num, Res) :- H <= Num, Bigger(T, Num, Res).
3. bigger([H|T], Num, [H|Res]) :- H > Num, bigger(T, Num, Res).

Вопрос: bigger([1,3], 2, Res).

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	bigger([1,3], 2, Res).	ТЦ: bigger([1,3], 2, Res).	Поиск знания с начала базы знаний.
	bigger([1,3], 2, Res).	ПР1: [] = [1,3] _ = 2 Res = []	Метка переносится ниже

		Неудача	

	bigger([1,3], 2, Res).	ПР2:
		[H|T] = [1,3]
		Num = 2
		Res = Res
		Успех
		Подстановка: {
		H = 1, T = [3], Num = 2,
		Res= Res
		}
			Тело ПР2 заменяет цель в резольвенте

2	1 <= 2 bigger([3], 2, Res)	1 <= 2 успех	Переход к следующей подцели
3	bigger([3], 2, Res)	ТЦ: Bigger([3], 2, Res)	Поиск знания с начала базы знаний.
	bigger([3], 2, Res)	ПР1: [] = [3] _ = 2 Res = [] Неудача	Метка переносится ниже

	bigger([3], 2, Res)	ПР2:
		[H|T] = [3]
		Num = 2
		Res = Res
		Успех
		Подстановка: {
		H = 3, T = [], Num = 2,
		Res= Res
		}
			Тело ПР2 заменяет цель в резольвенте

4	3 <= 2 bigger([], 2, Res).	3 <= 2 Ложь	Откат к шагу 3. Метка переносится ниже.
5	bigger([3], 2, Res)	ПР3: [H|T] = [3] Num = 2 [H|Res] = Res Успех Подстановка: {	Тело ПР3 заменяет цель в резольвенте

		H = 3, T = [], Num = 2, Res=[3 | Res] }	
6	3 > 2 bigger([], 2, Res).	3 > 2 успех	Переход к следующей подцели
7	bigger([], 2, Res).	ТЦ : bigger([], 2, Res).	Поиск знания с начала базы знаний.
	bigger([], 2, Res).	ПР1: [] = [] _ = 2 Res = [] Успех Подстановка: {Res = []}	Тело ПР1 заменяет цель в резольвенте
8	!		Так как встречен знак отсечения не будет попыток найти другие решения. Система завершает работу. Найдено решение Res = [3|[]] = [3]

Выводы: Эффективность работы достигнута за счет использования хвостовой рекурсии и отсечений. Также разбиения на части и проверки выполняются в заголовке правила. Все это позволяет уменьшить количество шагов, необходимых для достижения результата.