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»

ОТЧЕТ

к лабораторной работе № 20

По курсу: «Функциональное и логическое программирование»

Студент ИУ7-64Б
Лозовский А.А.

Преподаватель
Толпинская Н.Б

Москва, 2020 г.

Задание

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 систем

Ответы на вопросы

1. Как организуется хвостовая рекурсия в Prolog?

Организация хвостовой рекурсии:

- Рекурсивный вызов единственен и расположен в конце тела правила.
- До вычисления рекурсивного вызова не должно быть возможности сделать откат (т.е точки отката отсутствуют). Этого можно добиться, например, с помощью предиката отсечения.

Для выхода из рекурсии используется отдельное правило, в конце которого может находиться предикат отсечения.

2. Какое первое состояние резольвенты?

На первом шаге в резольвенте находится заданный вопрос (цель).

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

Список может быть разделен на «начало» и «конец». Начало списка – группа первых элементов(один и более), конец – обязательно один список. Разделить список можно с помощью вертикальной черты (|)

Пример: [1, 2, 3] -> [1 | [2, 3]] -> [1, 2 | [3]] -> [1, 2, 3 | []].

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

Два подряд идущий элемента могут быть выделены с помощью вертикальной черты и именованных переменных, например

get_fisrt_and_second([H1, H2 | T]),

Первый и третий с помощью вертикальной черты, именованных переменных и анонимной.

get_fisrt_and_third([H1, _, H3 | T])

5. Как формируется новое состояние резольвенты?

Преобразования резольвенты выполняются с помощью редукции – замены текущей цели на тело найденного в программе правила (с помощью унификации текущей цели и заголовка правила программы).

Преобразование резольвенты разделено на два этапа:

1) Берется верхняя из подцелей резольвенты (по стековому принципу) и заменяется на тело правила, найденного в программе.
2) Затем к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор цели и сопоставленного с ней правила).

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

Система завершает работу, в случае если метка расположна в конце процедуры(которая доказывается для ответа на поставленный вопрос) и не осталось альтернатив (для каждой подцели были найдены все возможные наборы значений и везде были проставлены метки), либо если ответ не был найден, но были просмотрены все возможные варианты.

domains

```
list = integer*
```

predicates

```
/* 1 арг – список, на обработку, 2 арг. – результирующий список, 3 арг. – число, с которым сравниваются элементы */
add(list, list, integer)

/*1 арг – список, на обработку, 2 арг. – результирующий список*/
odd(list, list)

/* 1 арг – список, на обработку, 2 арг. – результирующий список, 3 арг. – число, с которым сравниваются элементы спика*/
delete_all(list, list, integer)

/*1 арг – список, на обработку, 2 арг. – результирующий список*/
set(list, list)
```

clauses

```
/*задание 1*/
add([],[], _).
add([H|T], [H|T2], El) :- H > El, add(T, T2, El), !.
add([_|T], T2, EL) :- add(T, T2, El).

/*задание 2*/
odd([], []).
odd([_,H|T],[H|T2]) :- odd(T, T2), !.
odd([_|T], T2) :- odd(T, T2).

/*задание 3*/
delete_all([], [], _).
delete_all([El|T],T2, El) :- delete_all(T, T2, El), !.
delete_all([H|T], [H|T2], El):- delete_all(T, T2, El).

/*задание 4*/
set([], []).
set([H|T],[H|T1]) :- delete_all(T, T2, H), set(T2, T1).
```

Примеры целей и результатов работы программы

1. **Goal** add([6, 1, 5, 0, -3, 3, 8], Result, 3).
   Result Result=[6, 5, 8]

2. **Goal** add([1, 2, 3, 4], Result, 5).
   Result Result=[]

3. **Goal** odd([1, 2, 3, 7, 1], Result).
   Result Result=[2, 7]

4. **Goal** odd([1], Result).
   Result Result=[]

5. **Goal** odd([1, 2], Result).
   Result Result=[2]

6. **Goal** delete_all([1, 2, 3, 2, 5], Result, 2).
   Result Result=[1,3,5]

7. **Goal** set([1, 2, 2, 3, 4, 5, 2, 2, 3], Result).
   Result Result=[1,2,3,4,5]

Описание порядка поиска объектов

Текст процедуры:

I add([],[], _).
II add([H|T], [H|T2], El) :- H > El, add(T, T2, El), !.
III add([_|T], T2, EL) :- add(T, T2, El).

Вопрос

Goal add([4], Result, 2).

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
Шаг1	add([4], Result, 2)	ТЦ: add([4], Result, 2).	Поиск знания с начала базы знаний.
	add([4], Result, 2)	ТЦ: add([4], Result, 2). Сравниваемые термы: add([4], Result, 2) ПРI: add([],[], _). Результат: унификация невозможна	Возврат к ТЦ, метка переносится ниже.
	add([4], Result, 2)	ТЦ: add([4], Result, 2). Сравниваемые термы: add([4], Result, 2) ПРII: add([H|T], [H|T2], El) Результат: успех (подобрано знание) Подстановка: {H=4, T=[], Result=[4|T2], El=2}	Проверка тела ПРII

Шаг2	4 > 2 add([], T2, 2) !. Резольвента изменилась в 2 этапа.	ТЦ: 4 > 2 Результат: успех	Пустое тело заменяет цель в резольвенте
Шаг3	add([], T2, 2) !. Резольвента изменилась в 2 этапа.	ТЦ: add([], T2, 2)	Поиск знания с начала базы знаний.
	add([], T2, 2) !.	ТЦ: add([], T2, 2) Сравниваемые термы: add([], T2, 2) ПРI: add([],[], _). Результат: успех (подобрано знание) Подстановка: {T2=[]}	Пустое тело заменяет цель в резольвенте
Шаг4	!. Резольвента изменилась в 2 этапа.	Выполнение отсечения Результат: успех.	Пустое тело заменяет цель в резольвенте
	Пусто		Успех, резольвента пуста –> откат. Попытка отката ! приводит к завершению работы процедуры (правило доказано). Завершение работы программы: Result = [4|[]].

Вывод

Для повышения эффективности программы на пролог, можно использовать отсечения, чтобы ограничить количество вычислений, в случае, если они избыточны (например, как в случае с взаимоисключающими правилами), также можно использовать хвостовую рекурсию, ее главное отличие от стандартной реализации рекурсии в том, что при вычислениях, не возникает истощения памяти.

Исправления

ЛР17

Каково назначение использования алгоритма унификации? … поиск совпадений в текущей базе знаний **ЧЕГО, ЗАЧЕМ? Ответ не конкретный!**

Ответ: Для поиска ответа на вопрос системе необходимо найти подходящее знание в БЗ, для поиска такого знания используется алгоритм унификации. Формально, он помогает системе понять, что заголовок подошел: алгоритм попарно пытается сопоставить термы (текущую цель и термы из БЗ) и построить для них общий пример (для этого используется подстановка).

Каков результат работы алгоритма унификации?
Алгоритм унификации завершается «неудачей» или успехом. В случае успеха в результирующей ячейке сформируется подстановка. В качестве побочного эффекта будет построен наиболее общий терм??????.

Ответ: Алгоритм унификации может завершиться «успехом» и «неудачей». В случае успеха результирующая ячейка будет содержать подстановку(наиболее общий унификатор).

Как применяется подстановка? …путем замены текущей переменной на соответсвующий терм. ЭТО какой?

Ответ: Подстановкой называется множество пар, вида: $\{ X_i = t_i \}$, где X_i – переменная, а t_i – терм. Т.е происходит конкретизация переменной термом. Применение подстановки заключается в замене каждого вхождения переменной X_i на соответствующий терм (t_i). В результате применения подстановки переменные конкретизируются значениями, которые будут далее использованы при доказательстве истинности тела выбранного правила то есть значения переменных переходят на следующих шаг доказательства.

ЛР18

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

Ответ: Для поиска ответа на вопрос системе необходимо найти подходящее знание в БЗ, для поиска такого знания используется алгоритм унификации. Формально, он помогает системе понять, что заголовок подошел: алгоритм попарно пытается сопоставить термы (текущую цель и термы из БЗ) и построить для них общий пример (для этого используется подстановка).

Как применяется подстановка? …путем замены текущей переменной на соответсвующий терм. ЭТО какой?

Ответ: Подстановкой называется множество пар, вида: $\{ X_i = t_i \}$, где X_i – переменная, а t_i – терм. Т.е происходит конкретизация переменной термом. Применение подстановки заключается в замене каждого вхождения переменной X_i на соответствующий терм (t_i). В результате применения подстановки переменные конкретизируются значениями, которые будут далее использованы при доказательстве истинности тела выбранного правила то есть значения переменных переходят на следующих шаг доказательства.