|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Формирование и модификация списков на Prolog**

|  |  |
| --- | --- |
| **Студент: Луговой Д.М.**  **Группа: ИУ7-61Б**  **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**

**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.

**Текст процедуры …; Вопрос:…..**

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

**Ответы на вопросы**

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

Рекурсия в Prolog – упоминание в теле правила терма, который сопоставим с заголовком этого же правила. Для организации хвостовой рекурсии такой терм нужно поместить в конец тела описываемого правила.

* **Какое первое состояние резольвенты?**

Начальное состояние резольвенты – вопрос, т.е. цель доказательства.

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

Метод разбиения списка на начало и остаток заключается в разделении головы и хвоста списка вертикальной чертой |. Термы, находящиеся до вертикальной черты – начало списка, после - остаток. Начало списка – обязательно один или более элементов, остаток – всегда один терм, являющийся списком.

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

Для выделения элементов в списке можно использовать переменные, например, для получения первых двух подряд идущих элементов нужно использовать списковую структуру [H1, H2 | \_ ], H1 будет конкретизирована первым элементом, H2 – вторым. Чтобы выделить 1-й и 3-й элементы нужно использовать списковую структуру   
[H1, \_, H3 | \_ ], H1 будет конкретизирована первым элементом, H3 – третьим.

* **Как формируется новое состояние резольвенты?**

Новое состояние резольвенты формируется с помощью редукции и отката. В ходе редукции первая подцель резольвенты заменяется телом сопоставимого с ней правила из БЗ (т.е. правила, заголовок которого успешно унифицируется с подцелью), а затем к конъюнкции подцелей из резольвенты применяется подстановка, найденная при унификации подцели и заголовка правила. Также резольвента меняется в ходе отката, она возвращается в свое предыдущее состояние.

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

Работа системы останавливается, когда резольвента находится в исходном состоянии (т.е. является вопросом), а метка просмотренных знаний находится в самом конце БЗ (т.е. альтернатив для унификации с целью из резольвенты нет). Также работа системы останавливается в случае попытки отката при переходе через отсечение, при откате отсечение запрещает использование для согласования других правил, поэтому работа системы завершается.

**Текст программы**

domains

list = integer\*.

predicates

get\_bigger(list, list, integer).

get\_odd(list, list).

delete(list, list, integer).

list\_to\_set(list, list).

clauses

get\_bigger([], [], \_) :- !.

get\_bigger([H|T], [H|T3], Num) :- H > Num, get\_bigger(T, T3, Num), !.

get\_bigger([\_|T], Res, Num) :- get\_bigger(T, Res, Num).

get\_odd([], []) :- !.

get\_odd([\_, H2 | T], [H2 | T3]) :- get\_odd(T, T3), !.

get\_odd([\_ | T], Res) :- get\_odd(T, Res).

delete([], [], \_) :- !.

delete([Num|T], Res, Num) :- delete(T, Res, Num), !.

delete([H|T], [H|T3], Num) :- delete(T, T3, Num).

list\_to\_set([], []) :- !.

list\_to\_set([H|T], [H|T1]) :- delete(T, T2, H), list\_to\_set(T2, T1).

goal

get\_bigger([1,2,3,4,5], Res, 2).

%get\_odd([1,2,3,4,5], Res).

%delete([1,2,3,2,4], Res, 2).

%list\_to\_set([1,2,3,2,4,3,5], Set).

Предикат get\_bigger является предикатом для формирования списка из элементов числового списка, больших заданного значения, с использованием хвостовой рекурсии. Его аргументы: 1-ый – исходный список, из которого берутся элементы; 2-ой – результирующий список; 3-ий – число, с которым производится сравнение. Выход из рекурсии осуществляется, когда исходный список становится пустым. Во втором правиле осуществляется проверка, что голова исходного списка больше заданного числа, и в случае истинности голова присоединяется к результирующему списку. Иначе благодаря третьему правилу голова пропускается.

Предикат get\_odd является предикатом для формирования списка из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0), с использованием хвостовой рекурсии. Его аргументы: 1-ый - исходный список, из которого берутся элементы; 2-ой - результирующий список. Выход из рекурсии осуществляется, когда исходный список становится пустым. С помощью второго правила берется сразу 2 элемента из головы списка, и к результирующему списку присоединяется второй, т.е. нечетный (т.к. нумерация с 0). Если в списке нечетное число элементов, то благодаря третьему правилу список обрабатывается корректно.

Предикат delete является предикатом для удаления заданного элемента из списка с использованием хвостовой рекурсии. Его аргументы: 1-ый – исходный список, из которого берутся элементы; 2-ой – результирующий список; 3-ий – заданный элемент. Выход из рекурсии осуществляется, когда исходный список становится пустым. Во втором правиле осуществляется проверка, что голова исходного списка равна заданному числу, и в случае истинности голова пропускается. Иначе благодаря третьему правилу голова присоединяется к результирующему списку.

Предикат list\_to\_set является предикатом для преобразования списка во множество с использованием хвостовой рекурсии. Его аргументы: 1-ый – исходный список, из которого берутся элементы; 2-ой – результирующее множество. Выход из рекурсии осуществляется, когда исходный список становится пустым. Во втором правиле осуществляется присоединение головы исходного списка к результирующему списку, затем происходит удаление головы исходного списка из его хвоста с помощью предиката DELETE, и результат удаления используется далее как исходный список.

**Примеры работы**

|  |  |
| --- | --- |
| Вопрос | Результат |
| Задание 1. Сформировать список из элементов числового списка, больших заданного значения  **get\_bigger([1,2,3,4,5], Res, 2).** | Res=[3,4,5]  1 Solution |
| Задание 2. Сформировать список из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0)  **get\_odd([1,2,3,4,5], Res).** | Res=[2,4]  1 Solution |
| Задание 3. Удалить заданный элемент из списка  **delete([1,2,3,2,4], Res, 2).** | Res=[1,3,4]  1 Solution |
| Задание 4. Преобразовать список в множество  **list\_to\_set([1,2,3,2,4,3,5], Set).** | Set=[1,2,3,4,5]  1 Solution |

Таблица порядка работы системы, для 1-го задания,   
вопрос: get\_bigger([1,2,3,4,5], Res, 2). (Сформировать список из элементов списка [1, 2, 3, 4, 5], больших 2)

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты, и вывод: дальнейшие действия (почему?) | Для каких термов запускается алгоритм унификации: Т1=Т2 и каков **результат** (и подстановка) | Дальнейшие действия: прямой ход или откат (почему и к чему приводит?) |
| 1 | Резольвента:  get\_bigger([1,2,3,4,5], Res, 2)  Начальное состояние резольвенты – вопрос.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации с 1-м правилом  get\_bigger([1,2,3,4,5], Res, 2)  = get\_bigger([], [], \_)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  get\_bigger([1,2,3,4,5], Res, 2)  = get\_bigger([H|T], [H|T3], Num)  -Результат: успех, подстановка {H=1, T=[2,3,4,5], Res=[H|T3], Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 2 | Резольвента:  1 > 2  get\_bigger([2,3,4,5], T3, 2)  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации 1 > 2  - Результат: унификация неуспешна | Откат резольвенты к предыдущему состоянию, так как альтернативных правил для унификации с первой подцелью нет |
| 3 | Резольвента:  get\_bigger([1,2,3,4,5], Res, 2)  Резольвента вернулась к предыдущему состоянию в ходе отката.  Запускается унификация цели с очередным заголовком правила из БЗ | - Попытка унификации с 3-м правилом  get\_bigger([1,2,3,4,5], Res, 2) = get\_bigger([\_|T], Res, Num)  - Результат: успех, подстановка {\_=1, T=[2,3,4,5], Res=Res, Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 4 | Резольвента:  get\_bigger([2,3,4,5], Res, 2)  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  get\_bigger([2,3,4,5], Res, 2)  = get\_bigger([], [], \_)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  get\_bigger([2,3,4,5], Res, 2)  = get\_bigger([H|T], [H|T3], Num)  -Результат: успех, подстановка {H=2, T=[3,4,5], Res=[H|T3], Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 5 | Резольвента:  2 > 2  get\_bigger([3,4,5], T3, 2)  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации 2 > 2  - Результат: унификация неуспешна | Откат резольвенты к предыдущему состоянию, так как альтернативных правил для унификации с первой подцелью нет |
| 6 | Резольвента:  get\_bigger([2,3,4,5], Res, 2)  Резольвента вернулась к предыдущему состоянию в ходе отката.  Запускается унификация цели с очередным заголовком правила из БЗ | - Попытка унификации с 3-м правилом  get\_bigger([2,3,4,5], Res, 2) = get\_bigger([\_|T], Res, Num)  - Результат: успех, подстановка {\_=2, T=[3,4,5], Res=Res, Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 7 | Резольвента:  get\_bigger([3,4,5], Res, 2)  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  get\_bigger([3,4,5], Res, 2)  = get\_bigger([], [], \_)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  get\_bigger([3,4,5], Res, 2)  = get\_bigger([H|T], [H|T3], Num)  -Результат: успех, подстановка {H=3, T=[4,5], Res=[H|T3], Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 8 | Резольвента:  3 > 2  get\_bigger([4,5], T3, 2)  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации 3 > 2  - Результат: унификация успешна | Прямой ход, редукция резольвенты, так как унификация успешна |
| 9 | Резольвента:  get\_bigger([4,5], T3, 2)  !  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  get\_bigger([4,5], T3, 2)  = get\_bigger([], [], \_)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  get\_bigger([4,5], T3, 2)  = get\_bigger([H|T], [H|T3], Num)  -Результат: успех, подстановка {H=4, T=[5], T3=[H|T3], Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 10 | Резольвента:  4 > 2  get\_bigger([5], T3, 2)  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации 4 > 2  - Результат: унификация успешна | Прямой ход, редукция резольвенты, так как унификация успешна |
| 11 | Резольвента:  get\_bigger([5], T3, 2)  !  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  get\_bigger([5], T3, 2)  = get\_bigger([], [], \_)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  get\_bigger([5], T3, 2)  = get\_bigger([H|T], [H|T3], Num)  -Результат: успех, подстановка {H=5, T=[], T3=[H|T3], Num=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 12 | Резольвента:  5 > 2  get\_bigger([], T3, 2)  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации 5 > 2  - Результат: унификация успешна | Прямой ход, редукция резольвенты, так как унификация успешна |
| 13 | Резольвента:  get\_bigger([], T3, 2)  !  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  get\_bigger([], T3, 2)  = get\_bigger([], [], \_)  - Результат: успех, подстановка {T3=[], \_=2} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 14 | Резольвента:  !  В резольвенте в ходе редукции цель была заменена телом правила, заголовок которого был успешно унифицирован с вопросом, и была применена подстановка, найденная при унификации. | Отсечение истинно, так как встретилось при прямом ходе | Прямой ход, редукция резольвенты, так как унификация успешна |
| 15 | Резольвента пуста, так отсечение истинно и произошла редукция резольвенты. |  | Вывод ответа Sum=10, так как резольвента пуста, откат резольвенты к предыдущему состоянию |
| 16 | Резольвента:  !  Резольвента вернулась к предыдущему состоянию в ходе отката. | Отсечение ложно, так как был произведен откат. | Отсечение запрещает использовать оставшиеся правила, конец работы |

**Вывод**

Эффективность работы реализаций поставленных задач может быть достигнута за счет использования хвостовой рекурсии, благодаря которой стек резольвенты не растет. Также в предикатах get\_bigger, get\_odd, delete и list\_to\_set в условии выхода из рекурсии используется отсечение, что позволяет не перебирать оставшиеся правила и отбросить бесперспективные пути решения. Все проверки были внесены в заголовки правил, что также способствует эффективности работы. В реализации задач формирования списков создание списков внесено в заголовки правил, а в задаче формирования списка из элементов, стоящих на нечетных позициях, используется сразу 2 первых элемента исходного списка, что существенно уменьшает объем производимых работ.