|  |  |
| --- | --- |
| 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 № 19**

**Обработка списков на Prolog**

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

Москва.

2020 г.

**Цель работы –** изучить способы организации, представления и обработки списков в программах на Prolog, методы создания эффективных рекурсивных программ обработки списков и порядок их реализации.

**Задачи работы**

Приобрести навыки использования 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 эффективную программу, (комментируя назначение аргументов), позволяющую:**

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка
3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)

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.

**Текст процедуры, Вопрос:…..**

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

**Ответы на вопросы**

* **Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?**

Рекурсия – это ссылка при описании объекта на этот же объект. 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. Выход из рекурсии организуется с помощью отдельного предложения этой же процедуры, при организации выхода необходимо проследить за тем, чтобы система снова не использовала нижележащие правила в ходе отката, это возможно сделать с помощью отсечения.

* **В каких пределах программы переменные уникальны?**

Именованные переменные уникальны в пределах одного предложения, любая анонимная переменная уникальна.

* **Какое первое состояние резольвенты?**

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

* **В какой момент, и каким способом системе удается получить доступ к голове списка?**

Для доступа к голове списка используется метод разбиения списка на начало и остаток, он заключается в разделении головы и хвоста списка вертикальной чертой |. Термы, находящиеся до вертикальной черты – начало списка. Для получения доступа к голове может использоваться переменная, пример: [H | ], переменная H будет конкретизирована головой списка при унификации исходного списка и терма [H | ].

* **Каково назначение использования алгоритма унификации?**

Алгоритм унификации предназначен для логического вывода, система использует его для того, чтобы "увидеть одинаковость" термов, чтобы подобрать необходимое знание из БЗ, заголовок которого сопоставим с первой подцелью резольвенты, и дать ответ «Да» на поставленный вопрос. Унификация является основным вычислительным шагом работы программы.

* **Каков результат работы алгоритма унификации?**

Результатом работы унификации является успех или неудача, побочным эффектом является построенная в ходе унификации подстановка.

* **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.

* **Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?**

Подстановка, полученная при унификации термов, применяется к новой резольвенте, первая подцель которой была заменена телом правила, с заголовком которого подцель успешно унифицировалась, путем конкретизации переменных соответствующими термами из подстановки. Подстановка применяется ко всей резольвенте, т.е. ко всем ее подцелям.

* **В каких случаях запускается механизм отката?**

Механизм отката запускается в случае попадания в тупиковую ситуацию, т.е. если из текущего состояния перейти в новое состояние невозможно, или в случае, если осуществляется поиск всех возможных ответов и очередной ответ был найден.

* **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

length(list, integer).

length\_helper(list, integer, integer).

sum(list, integer).

sum\_helper(list, integer, integer).

sum\_odd(list, integer).

sum\_odd\_helper(list, integer, integer).

clauses

length(List, Len) :- length\_helper(List, Len, 0).

length\_helper([], Len, Len) :- !.

length\_helper([\_|T], Len, Cur) :- Cur1 = Cur + 1, length\_helper(T, Len, Cur1).

sum(List, Sum) :- sum\_helper(List, Sum, 0).

sum\_helper([], Sum, Sum) :- !.

sum\_helper([H|T], Sum, Cur) :- Cur1 = Cur + H, sum\_helper(T, Sum, Cur1).

sum\_odd(List, Sum) :- sum\_odd\_helper(List, Sum, 0, 0).

sum\_odd\_helper([], Sum, Sum) :- !.

sum\_odd\_helper([\_, H | T], Sum, Cur) :- Cur1 = Cur + H2, sum\_odd\_helper(T, Sum, Cur1), !.

sum\_odd\_helper([\_ | T], Sum, Cur) :- sum\_odd\_helper(T, Sum, Cur).

goal

length([1, 1, 1, 1], Len).

%sum([1, 2, 3, 4], Sum).

%sum\_odd([1, 2, 3, 4], Sum).

Предикат length является вспомогательным предикатом для поиска длины списка. Основной предикат для поиска длины списка с использованием хвостовой рекурсии – length\_helper. Его аргументы: 1-ый - текущий список; 2-ой - результат поиска длины списка; 3-ий – текущая длина списка. Выход из рекурсии осуществляется, когда текущий список становится пустым.

Предикат sum является вспомогательным предикатом для вычисления суммы элементов списка. Основной предикат для вычисления суммы элементов списка с использованием хвостовой рекурсии – sum\_helper. Его аргументы: 1-ый - текущий список; 2-ой - результат вычисления суммы элементов списка; 3-ий – текущая сумма элементов списка. Выход из рекурсии осуществляется, когда текущий список становится пустым.

Предикат sum\_odd является вспомогательным предикатом для вычисления суммы элементов списка, стоящих на нечетных позициях. Основной предикат для вычисления суммы элементов списка, стоящих на нечетных позициях, с использованием хвостовой рекурсии – sum\_odd\_helper. Его аргументы: 1-ый - текущий список; 2-ой - результат вычисления суммы элементов списка; 3-ий – текущая сумма элементов списка. Выход из рекурсии осуществляется, когда текущий список становится пустым. С помощью второго правила берется сразу 2 элемента из головы списка, и к сумме прибавляется второй, т.е. нечетный (т.к. нумерация с 0). Если в списке нечетное число элементов, то благодаря третьему правилу список обрабатывается корректно.

**Примеры работы**

|  |  |
| --- | --- |
| Вопрос | Результат |
| Задание 1. Найти длину списка.  **length([1, 1, 1, 1], Len).** | Len=4  1 Solution |
| Задание 2. Найти сумму элементов числового списка.  **sum([1, 2, 3, 4], Sum).** | Sum=10  1 Solution |
| Задание 3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)  **sum\_odd([1, 2, 3, 4], Sum).** | Sum=6  1 Solution |

Таблица порядка работы системы, для 2-го задания,   
вопрос: sum([1, 2, 3, 4], Sum). (Сумма элементов списка [1, 2, 3, 4])

|  |  |  |  |
| --- | --- | --- | --- |
| № 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:  sum([1, 2, 3, 4], Sum)  Начальное состояние резольвенты – вопрос.  Запускается унификация цели с очередным заголовком правила из БЗ. | - Попытка унификации sum([1, 2, 3, 4], Sum)  = sum(List, Sum)  - Результат: Успех, подстановка {List=[1,2,3,4], Sum=Sum} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 2 | Резольвента:  sum\_helper([1,2,3,4], Sum, 0)  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 с 1-м правилом  sum\_helper([1,2,3,4], Sum, 0)  = sum\_helper([], Sum, Sum)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом  sum\_helper([1,2,3,4], Sum, 0)  = sum\_helper([H|T], Sum, Cur)  -Результат: успех, подстановка {H=1, T=[2,3,4], Sum=Sum, Cur=0} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 3 | Резольвента:  Cur1 = 0 + 1  sum\_helper([2,3,4], Sum, Cur1)  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 Cur1 = 0 + 1  - Результат: успех, подстановка  {Cur1 = 1} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 4 | Резольвента:  sum\_helper([2,3,4], Sum, 1)  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  sum\_helper([2,3,4], Sum, 1)  = sum\_helper([], Sum, Sum)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом sum\_helper([2,3,4], Sum, 1)  = sum\_helper([H|T], Sum, Cur)  -Результат: успех, подстановка {H=2, T=[3,4], Sum=Sum, Cur=1} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 5 | Резольвента:  Cur1 = 1 + 2  sum\_helper([3,4], Sum, Cur1)  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 Cur1 = 1 + 2  - Результат: успех, подстановка  {Cur1 = 3} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 6 | Резольвента:  sum\_helper([3,4], Sum, 3)  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  sum\_helper([3,4], Sum, 3)  = sum\_helper([], Sum, Sum)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом sum\_helper([3,4], Sum, 3)  = sum\_helper([H|T], Sum, Cur)  -Результат: успех, подстановка {H=3, T=[4], Sum=Sum, Cur=3} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 7 | Резольвента:  Cur1 = 3 + 3  sum\_helper([4], Sum, Cur1)  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 Cur1 = 3 + 3  - Результат: успех, подстановка  {Cur1 = 6} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 8 | Резольвента:  sum\_helper([4], Sum, 6)  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  sum\_helper([4], Sum, 6)  = sum\_helper([], Sum, Sum)  - Результат: неудача, переход к следующему правилу  - Попытка унификации со 2-м правилом sum\_helper([4], Sum, 6)  = sum\_helper([H|T], Sum, Cur)  -Результат: успех, подстановка {H=4, T=[], Sum=Sum, Cur=6} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 9 | Резольвента:  Cur1 = 6 + 4  sum\_helper([], Sum, Cur1)  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 Cur1 = 6 + 4  - Результат: успех, подстановка  {Cur1 = 10} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 10 | Резольвента:  sum\_helper([], Sum, 10)  В резольвенте в ходе редукции первая подцель была удалена и была применена подстановка.  Запускается унификация первой подцели. | - Попытка унификации с 1-м правилом  sum\_helper([], Sum, 10)  = sum\_helper([], Sum, Sum)  - Результат: успех, подстановка {Sum=10} | Прямой ход, редукция резольвенты, так как унификация успешна |
| 11 | 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 |
| 12 | Резольвента пуста, так отсечение истинно и произошла редукция резольвенты. |  | Вывод ответа Sum=10, так как резольвента пуста, откат резольвенты к предыдущему состоянию |
| 13 | 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 программ поиска длины списка, вычисления суммы элементов числового списка и вычисления суммы элементов числового списка, стоящих на нечетных позициях, достигнута за счет использования хвостовой рекурсии, благодаря которой стек резольвенты не растет. Также в предикатах length\_helper, sum\_helper и sum\_odd\_helper 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 программе вычисления суммы элементов числового списка, стоящих на нечетных позициях, перебор осуществляется по 2 элемента, что существенно сокращает объем необходимой работы.

**Замечания к лабораторной работе №15**

|  |  |  |
| --- | --- | --- |
| 5 | - Попытка унификации get\_property\_by\_name("Ivanov", Type, Money) =get\_property\_by\_name(Name, Type, Money)  - Результат: Успех, подстановка {Name=”Ivanov”, Type=Type, Money=Money} | Прямой ход, переход к началу БЗ |

**Зачем к началу? А что искать? А что тело не надо проверить?**

**Ответ:** Некорректно сформулировал, имелось в виду, что поиск знаний для сопоставления с новой первой подцелью резольвенты будет производится с начала БЗ.

**Замечания к лабораторной работе №17**

* **Каково назначение использования алгоритма унификации?**

Алгоритм унификации предназначен для логического вывода, система использует его для того, чтобы "увидеть одинаковость" ЗАЧЕМ???термов и дать ответ «Да» на поставленный вопрос. СРАЗУ??

**Ответ:** Необходимо иметь возможность увидеть одинаковость термов, чтобы подобрать нужное знание из БЗ, заголовок которого сопоставим с первой подцелью резольвенты. Это позволит с помощью редукции получить новую резольвенту. Путем поиска знаний для сопоставления с первыми подцелями резольвенты и ее дальнейшей редукции, система возможно сможет получить пустую резольвенту и тогда даст ответ «Да» на поставленный вопрос.