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

Тема Обработка списков на Prolog

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

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

Ответить на вопросы (коротко**):**

1. Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как можно организовать выход из рекурсии в Prolog?
2. Какое первое состояние резольвенты?
3. В каких пределах программы переменные уникальны?
4. В какой момент, и каким способом системе удается получить доступ к голове списка?
5. Каково назначение использования алгоритма унификации?
6. Каков результат работы алгоритма унификации?
7. Как формируется новое состояние резольвенты?
8. Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?
9. В каких случаях запускается механизм отката?
10. Когда останавливается работа системы? Как это определяется на формальном уровне?

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	…	…	…
…	…	…	…

Что такое рекурсия?

Рекурсия — это ссылка на самого себя.

Как организуется хвостовая рекурсия в Prolog?

Хвостовая рекурсия в Prolog организуется за счет расположения повторного вызова функции последней подцелью в конъюктивном правиле.

Как можно организовать выход из рекурсии в Prolog?

В Prolog рекурсия орагинизуется с помощью нескольких правил, часть из которых не являются рекурсивными и служат для выхода из рекурсии, они используют отсечения для выхода из рекурсии.

Какое первое состояние резольвенты?

Первое состояние резольвенты - заданный вопрос.

В каких пределах программы переменные уникальны?

Именованные переменные уникальны в рамках одного предложения. Анонимные переменные уникальны везде.

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

Получить голову или хвост списка можно при унификации списка с [H|T], H - голова, T - хвост.

Каково назначение использования алгоритма унификации?

Назначение - поиск знания, которое является ответом на конкретный вопрос.

Каков результат работы алгоритма унификации?

Результат работы алгоритма унификации — ответ «да» или «нет», а также конкретизация переменных.

Как формируется новое состояние резольвенты?

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

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

Если алгоритм унификации завершился успешно и найдена подстановка, соответствующие переменные конкретизируются полученными значениями.

В каких случаях запускается механизм отката?

Механизм отката к предыдущему шагу выполняется в случае, когда унификация завершается тупиковой ситуацией(неудачей). Кроме того, механизм используется для того, чтобы получить все возможные ответы.

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

Завершение работы программы достигается, когда резольвента пуста.

Текст программы

```
domains
        list = integer*.

predicates
```

```
  len(list, integer).
  len(list, integer, integer).

  sum(list, integer).
  sum(list, integer, integer).

  sumOdd(list, integer).
  sumOdd(List, integer, integer).

clauses

  len(List, Len) :- len(List, 0, Len).
  len([], Len, Len):- !.
  len([_|T], Cur, Len) :- NewLen = Cur + 1, len(T, NewLen, Len).

  sum(List, Sum) :- sum(List, 0, Sum).
  sum([], Sum, Sum):- !.
  sum([H|T], Cur, Sum) :- NewSum = Cur + H, sum(T, NewSum, Sum).

  sumOdd(List, Sum):- sumOdd(List, 0, Sum).
  sumOdd([], Sum, Sum):- !.
  sumOdd([_], Sum, Sum):- !.
  sumOdd([_|[H|T]], Cur, Sum):- NewSum = Cur+H, sumOdd(T, NewSum, Sum).

goal

 % len([1, 2, 3, 4, 5], Len).
 %sum([1, 2, 8, 9], Sum).
 sumOdd([1, 2, 0, 8, 1], Sum).
```

Примеры работы:

Найти длину списка (по верхнему уровню);

len([1, 2, 3, 4, 5], Len). → 5
len([1]) → 1
len([]) → 0

Найти сумму элементов числового списка

sum([1, 2, 8, 9], Sum). → 20
sum([1, 2], Sum). → 3

Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)

sumOdd([1, 2, 0, 8, 1], Sum). → 10
sumOdd([1, 2, 0, 8], Sum). → 10
sumOdd([1, 2], Sum). → 2
sumOdd([1], Sum). → 0

Текст процедуры

1. sum(List, Sum) :- sum(List, 0, Sum).
2. sum([], Sum, Sum):- !.
3. sum([H|T], Cur, Sum) :- NewSum = Cur + H, sum(T, NewSum, Sum).

Вопрос: sum([1, 2, 3], Sum).

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	sum([1, 2, 3], Sum).	ТЦ: sum([1, 2, 3], Sum).	Поиск знания с начала базы знаний.
	sum([1, 2, 3], Sum).	ПР1: List = [1, 2, 3] Sum = Sum Успех Пподстановка: {List = [1, 2, 3], Sum = Sum}	Тело ПР1 заменяет цель в резольвенте
2	sum([1,2,3], 0, Sum)	ТЦ: Sum([1,2,3], 0, Sum)	Поиск знания с начала бз
	sum([1,2,3], 0, Sum)	ПР1: Унификация невозможна => неудача	Метка переносится ниже
	sum([1,2,3], 0, Sum)	ПР2: [] = [1,2,3] Sum = 0 Sum = Sum Неудача	Метка переносится ниже
	sum([1,2,3], 0, Sum)	ПР3: [H|T] = [1,2,3] Cur = 0 Sum = Sum Успех Подстановка: {H=1, T = [2,3], Cur = 0, Sum=Sum }	Тело ПР3 заменяет цель в резольвенте
3	NewSum = 0 + 1 sum([2,3], NewSum, Sum).	NewSum = 1	Успех Переход к следующей цели
4	sum([2,3], 1, Sum).	ТЦ: sum([2,3], 1, Sum).	Поиск знания с начала бз

	sum([2,3], 1, Sum).	ПР1: Унификация невозможна => неудача	Метка переносится ниже
	sum([2,3], 1, Sum).	ПР2: [] = [2,3] Sum = 1 Sum = Sum Неудача	Метка переносится ниже
	sum([2,3], 1, Sum).	ПР3: [H|T] = [2,3] Cur = 1 Sum = Sum Успех Подстановка: {H=2, T = [3], Cur = 1, Sum=Sum }	Тело ПР3 заменяет цель в резольвенте
5	NewSum = 1 + 2 sum([3], NewSum, Sum).	NewSum = 3	Успех Переход к следующей цели
6	sum([3], 3, Sum).	ТЦ: sum([3], 3, Sum).	Поиск знания с начала бз
	sum([3], 3, Sum).	ПР1: Унификация невозможна => неудача	Метка переносится ниже
	sum([3], 3, Sum).	ПР2: [] = [3] Sum = 2 Sum = Sum Неудача	Метка переносится ниже
	sum([3], 3, Sum).	ПР3: [H|T] = [3] Cur = 2 Sum = Sum Успех Подстановка: {H=3, T = [], Cur = 2, Sum=Sum }	Тело ПР3 заменяет цель в резольвенте
7	NewSum = 3 + 3 sum([], NewSum, Sum).	NewSum = 6	Успех Переход к следующей

			цели
8	sum([], 6, Sum).	ТЦ: sum([], 6, Sum).	Поиск знания с начала бз
	sum([], 6, Sum).	ПР1: Унификация невозможна => неудача	Метка переносится ниже
	sum([], 6, Sum).	ПР2: [] = [] Sum = 6 Sum = Sum Успех Подстановка: {Sum = 6}	Тело ПР2 заменяет цель в резольвенте
9	!		Так как встречен знак отсечения не будет попыток найти другие решения. Система завершает работу. Найдено решение Sum = 6

Вывод:
Эффективность работы программы достигнута за счет использования отсечения и хвостовой рекурсии. Отсечение позволяет уменьшить количество проверок. За счет использования хвостовой рекурсии резольвента не увеличивается в процессе поиска ответа.

Исправления к лр 15

№15

7. Унификация каких термов запускается на **самом первом** шаге работы системы?
В начале работы программа выполняет унификацию терма-вопроса и всех За один шаг не удасться! предложений из

№15

Унификация каких термов запускается на самом первом шаге работы системы?
На самом первом шаге запускается унификация вопроса и первого в базе знаний унифицируемого с ним терма нет,формально не правильно1 – например: в тексте – обертка для рек-ого предиката раньше рабочего предиката, я наберу вопрос, унифицируемый с предикатом, а такой терм в теле!!! обертки????

Побочнымрезультатом унификации является конкретизация процесс? Переменных…Исправить!

Результат работы алгоритма унификации — подстановка.
На первом шаге запускается унификация вопроса и первого в базе знаний унифицируемого с ним заголовка правила.