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»

Дисциплина: «Функциональное и логическое программирование»

Лабораторная работа №18

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватели: Толпинская Н. Б.,
Строганов Ю. В.

Москва, 2020 г.

Цель работы

Изучить рекурсивные способы организации программ на Prolog, методы формирования эффективных рекурсивных программ и порядок реализации таких программ.

Задачи работы

Приобрести навыки использования рекурсии на Prolog, эффективного способа ее организации и прядка работы соответствующей программы.

Изучить возможность и необходимость использования системных предикатов в рекурсивной программе на Prolog, принципы и особенности порядка работы такой программы. Способ формирования и изменения резольвенты в этом случае и порядок формирования ответа.

Задание

Ответить на вопросы:

- Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии в Prolog?
- Какое первое состояние резольвенты?
- В каком случае система запускает алгоритм унификации? Каково назначение использования алгоритма унификации? Каков результат работы алгоритма унификации?
- В каких пределах программы переменные уникальны?
- Как применяется подстановка, полученная с помощью алгоритма унификации?
- Как изменяется резольвента?
- В каких случаях запускается механизм отката?

Используя хвостовую рекурсию, разработать программу, позволяющую найти

1. **n!,**
2. **n-е** число Фибоначчи.

Убедиться в правильности результатов.

Для одного из вариантов **ВОПРОСА** и каждого **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!

Вопрос:...

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков **результат** (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1...	...	...	Комментарий, вывод...
...	...	...	...

Реализация программы n!.

```
domains

num, help, result = integer

predicates

factorial(num, result)
factorial_help(num, help, result)

clauses

factorial_help(1, Help, Result) :- Result = Help, !.

factorial_help(Num, Help, Result) :-
Help1 = Help * Num, Num1 = Num - 1, factorial_help(Num1, Help1, Result).


factorial(0, 1) :-!.

factorial(Num, Result) :- Help = 1, factorial_help(Num, Help, Result).
```

Порядок и особенности выполнения программы и формирования результата.

Правило factorial(Num, Result) запускает рекурсивное правило factorial_help(Num, Help, Result), вводя дополнительный параметр Help = 1, который накапливает в себе полученный результат и служит для формирования хвостовой рекурсии.

С помощью правила factorial(0, 1) учитывается отдельный случай: 0!, который должен быть равен 1.

Выход из рекурсии осуществляется с помощью правила factorial_help(1, Help, Result), когда Num = 1.

Тесты

Пример 1:

```
goal
        factorial(5, Result).
%Вывод:
```

```
        Result=120
        1 Solution
```

Пример 2:

```
goal
        factorial(1, Result).
%Вывод:
        Result=1
        1 Solution
```

Пример 3:

```
goal
        factorial(0, Result).
%Вывод:
        Result=1
        1 Solution
```

Порядок работы системы:

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков **результат** (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	factorial(0, Result).	T1 = factorial(0, Result); T2 = factorial_help(1, Help, Result). Неудача (функторы factorial и factorial_help не равны).	Прямой ход к следующему предложению.
2	factorial(0, Result).	T1 = factorial(0, Result); T2 = factorial_help(Num, Help, Result). Неудача (функторы factorial и factorial_help не равны).	Прямой ход к следующему предложению.

3	factorial(0, Result).	T1 = factorial(0, Result); T2 = factorial(0, 1). Успех. Подстановка 0 = 0, Result = 1.	Вывод: Result = 1. Отката к предыдущему состоянию резольвенты не происходит из-за предиката отсечения !. Завершение работы.

Реализация программы n-е число Фибоначчи.

```
domains

elem, num, help, result = integer

predicates

fib(num, result)
fib_help(elem First, elem Second, help, num, result)

clauses

fib_help(_, Second, Help, Num, Result) :- Help = Num, Result = Second, !.

fib_help(First, Second, Help, Num, Result) :- New_second = First + Second,
New_help = Help + 1, fib_help(Second, New_second, New_help, Num, Result).


fib(0, 1) :-!.

fib(Num, Result) :- First = 1, Second = 1, Help = 1,
fib_help(First, Second, Help, Num, Result).
```

Порядок и особенности выполнения программы и формирования результата.

Правило fib(Num, Result) запускает рекурсивное правило fib_help(First, Second, Help, Num, Result), вводя дополнительные параметры First = 1, Second

= 1, Help = 1.

Поскольку число Фибоначчи - это число, полученное в результате суммы двух предыдущих, необходимо было задать начальные значения первых двух элементов: First = 1, Second = 1.

Переменная Help - переменная счетчик, обозначающая какой по счету является элемент Second (отсчет начинается с 0).

С помощью правила fib(0, 1) учитывается отдельный случай, когда требуется вывести 0-ой элемент последовательности Фибоначчи. Соответственно, результат должен быть равен 1.

Выход из рекурсии осуществляется с помощью правила fib_help(_, Second, Help, Num, Result), когда элемент Second является Num-ным элементом по счету.

Тесты

Пример 1:

```
goal
        fib(5, Result).
%Вывод:
        Result=8
        1 Solution
```

Пример 2:

```
goal
        fib(1, Result).
%Вывод:
        Result=1
        1 Solution
```

Пример 3:

```
goal
        fib(0, Result).
%Вывод:
        Result=1
        1 Solution
```

Порядок работы системы:

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков **результат** (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	fib(0, Result).	T1 = fib(0, Result); T2 = fib_help(_, Second, Help, Num, Result). Неудача (фунткоры fib и fib_help не равны).	Прямой ход к следующему предложению, Аналогичная ситуация в следующем предложении. Прямой ход к следующему предложению.
3	fib(0, Result).	T1 = fib(0, Result); T2 = fib(0, 1). Успех. Подстановка 0 = 0, Result = 1.	Вывод: Result = 1. Отката к предыдущему состоянию резольвенты не происходит из-за предиката отсечения !. Завершение работы.

Выводы

За счет чего может быть достигнута эффективность работы системы?

Эффективность работы системы может быть достигнута за счет использования хвостовой рекурсии.

Также, эффективность работы системы может быть достигнута за счет использования специальных средств управления порядком работы системы, таким как предикат отсечения ! (cat), который позволяет отсекать в определенных случаях бесперспективные пути доказательства.

Ответы на вопросы

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

Рекурсия – это ссылка при описании объекта на описываемый объект.

Хвостовая рекурсия в Prolog - ссылка на знание, эту же процедуру, последняя в теле правила.

Пример оформления хвостовой рекурсии: p(ar1, . . . , argN) :- <выход из рекурсии>. . . . p :- t1, . . . ,tk, p(arg11, . . . , argN1).

При организации выхода из рекурсии необходимо учитываьб, что система использует механизм отката, Следовательно требуется обеспечить, чтобы после выхода из рекурсии система не пробовала использовать вновь ниже лежащие правила (возможно используя отсечение).

Какое первое состояние резольвенты?

Если задан простой вопрос, то сначала он попадает в резольвенту.

В каком случае система запускает алгоритм унификации? Каково назначение использования алгоритма унификации? Каков результат работы алгоритма унификации?

Процесс унификации запускается, если есть цель, которую необходимо доказать (формально: если резольвента не пуста).

Назначение алгоритма унификации заключается в попарном сопоставлении термов и попытке построить для них общий пример.

Результатом использования алгоритма унификации может быть успех или тупиковая ситуация (неудача).

В каких пределах программы переменные уникальны?

Переменные уникальны в пределах предложения.

Исключение – анонимные переменные – каждая такая переменная является отдельной сущностью и применяется, когда ее значение неважно для данного предложения.

Как применяется подстановка, полученная с помощью алгоритма унификации?

Применение подстановки $x_1 = t_1, \ldots, x_n = t_n$ заключается в замене каждого вхождения переменной x_i на соответствующий терм t_i.

Как изменяется резольвента?

Состояние резольвенты меняется в процессе доказательства (для хранения резольвенты система использует стек). Преобразования резольвенты выполняются с помощью **редукции**.

Редукцией цели G с помощью программы P называется замена цели G телом того правила из P, заголовок которого унифицируется с целью.

В каких случаях запускается механизм отката?

Механизм отката запускается, если возникла тупиковая ситуация (достигнут конец БЗ) либо резольвента пуста. В таких случаях происходит откат к предыдущему состоянию резольвенты.