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

О т ч е т

по лабораторной работе № 18

Дисциплина: «Функциональное и логическое программирование»

Выполнила: Овчинникова А. П.

Группа: ИУ7-65Б

Преподаватель: Толпинская Н. Б.

Строганов Ю. В.

Москва, 2020

Задание

Ответить на вопросы.

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!

Теоретическая часть

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

Рекурсия – это ссылка на определяемый объект во время его определения.

Для осуществления хвостовой рекурсии в Prolog рекурсивный вызов определяемого предиката должен быть последней подцелью в теле рекурсивного правила и к моменту рекурсивного вызова не должно остаться точек возврата. То есть у подцелей, расположенных левее рекурсивного вызова определяемого предиката, не должно оставаться каких-то непроверенных вариантов и у процедуры не должно быть предложений, расположенных ниже рекурсивного правила. Пролог распознает хвостовую рекурсию и устраняет связанные с ней дополнительные расходы. Этот процесс называется оптимизацией хвостовой рекурсии или оптимизацией последнего вызова.

Базис рекурсии — это предложение, определяющее некую начальную ситуацию или ситуацию в момент прекращения. Как правило, в этом предложении записывается некий простейший случай, при котором ответ

получается сразу даже без использования рекурсии. Это предложение часто содержит условие, при выполнении которого происходит выход из рекурсии или отсечение.

2. Какое первое состояние резольвенты?

Первое состояние резольвенты – заданный вопрос.

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

Процесс унификации запускается автоматически, если есть что доказывать, то надо запускать алгоритм унификации. Пользователь имеет право запустить этот процесс вручную, с помощью утверждения T1 = T2, включенного в текст программы. Если резольвента не пуста – запускается алгоритм унификации (для хранения резольвенты используется стек, соответственно, если стек не пуст – запускается алгоритм унификации).

Назначение унификации – подобрать нужное в данный момент правило. Система подбирает сопоставимые с целью правила с помощью алгоритма унификации.

В результате унификации формируются подстановки.

4. В каких пределах программы переменные уникальны?

Именованные переменные уникальны в рамках предложения, а анонимная переменная – любая уникальна.

5. Как применяется подстановка, полученная с помощью алгоритма унификации?

В случае успешного согласования программы (базы знаний) и вопроса, в качестве побочного эффекта формируется подстановка, которая содержит значения переменных, при которых вопрос является примером программы. Соответствующие переменные конкретизируются полученными значениями.

6. Как изменяется резольвента?

Преобразования резольвенты выполняются с помощью редукции. Новая резольвента образуется в два этапа:

- В текущей резольвенте выбирается одна из подцелей (по стековому принципу – верхняя) и для нее выполняется редукция – замена подцели на тело найденного правила (если удалось найти правило).
- Затем к полученной конъюнкции целей применяется подстановка, полученная как наибольший общий унификатор выбранной цели и заголовка сопоставленного с ней правила.

Если для редукции цели из резольвенты был выбран факт из БЗ, то новая резольвента будет содержать в конъюнкции на одну цель меньше. Если задан простой вопрос и подобран для редукции факт, то произойдет немедленное его согласование. А если для простого вопроса подобрано правило, то число целей в резольвенте не уменьшится, т. к. цель будет заменена телом подобранного правила.

7. В каких случаях запускается механизм отката?

В ситуации, когда решение не найдено, и из данного состояния невозможен переход в новое состояние, автоматически включается бэктрэкинг. Происходит возврат к моменту, где еще можно сделать другой альтернативный выбор, то есть к предыдущему состоянию резольвенты. Бэктрэкинг возможен только при наличии альтернативных путей унификации цели.

Код программы

Программа 1. Факториал.

```
predicates

fact_inner(integer, integer, integer, integer). % number, result, current number, current number factorial
fact(integer, integer). % number, result

clauses
```

```
fact_inner(N, F, N, F) :- !. % base
fact_inner(N, F, N1, F1) :-
        N2 = N1 + 1, % new (next) current number
        F2 = F1 * N2, % new (next) factorial (of N2)
        fact_inner(N, F, N2, F2). % recursion

fact(N, F) :- fact_inner(N, F, 1, 1).

goal

fact(5, RES).
%fact(4, RES).
%fact(3, RES).
%fact(2, RES).
%fact(1, RES).
%fact(0, RES). % error
```

Программа 2. Фибоначчи.

```
predicates

fib_inner(integer, integer, integer). % serial number, fib, next fib
fib(integer, integer). % serial number, result

clauses

fib_inner(1, 1, 1) :- !. % first two fib = 1
fib_inner(N, FN, FN1) :-
```

```
        N1 = N - 1,
        fib_inner(N1, FN_1, FN), % FN_1 is N-1 fib, FN is N fib
        FN1 = FN + FN_1.  % N+1 fib

fib(N, FN) :-
        fib_inner(N, FN, _).

goal

fib(1, RES).
%fib(2, RES).
%fib(3, RES).
%fib(4, RES).
%fib(5, RES).
%fib(6, RES).
%fib(7, RES).
```

Примеры работы программы

Программа 1. Факториал.

Вопрос	Ответ
fact(5, RES).	RES=120 1 Solution
fact(4, RES).	RES=24 1 Solution
fact(3, RES).	RES=6 1 Solution
fact(2, RES).	RES=2 1 Solution
fact(1, RES).	RES=1 1 Solution
fact(0, RES).	PROGRAM ERROR. Module:OBJ\GOAL$000.PRO Pos:610 Message:1031 Arithmetic overflow

Программа 2. Фибоначчи.

Вопрос	Ответ

fib(1, RES).	RES=1 1 Solution
fib(2, RES).	RES=1 1 Solution
fib(3, RES).	RES=2 1 Solution
fib(4, RES).	RES=3 1 Solution
fib(5, RES).	RES=5 1 Solution
fib(6, RES).	RES=8 1 Solution
fib(7, RES).	RES=13 1 Solution

Порядок работы системы для вопроса *fact(2, RES).* приведен в таблице 1. Порядок работы системы для вопроса *fib(2, RES).* приведен в таблице 2.

Таблица 1. 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
1	Резольвента: fact(2, RES).	Попытка унификации: T1= fact(2, RES). T2=fact_inner(N, F, N, F) :- !. Результат: неудача. Разные функторы.	Откат, переход к следующему предложению
2	Резольвента: fact(2, RES).	Попытка унификации: T1= fact(2, RES). T2=fact_inner(N, F, N1, F1) :- N2 = N1 + 1, F2 = F1 * N2, fact_inner(N, F, N2, F2). Результат: неудача. Разные функторы.	Откат, переход к следующему предложению
3	Резольвента: fact(2, RES).	Попытка унификации: T1= fact(2, RES). T2= fact(N, F) :- fact_inner(N, F, 1, 1). Результат: успех. Подстановка: {N=2, F=RES}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.
4	Резольвента: fact_inner(2, RES, 1, 1).	Попытка унификации: T1= fact_inner(2, RES, 1, 1). T2= fact_inner(N, F, N, F) :- !. Результат: неудача, 3 != 1	Откат, переход к следующему предложению

5	Резольвента: fact_inner(2, RES, 1, 1).	Попытка унификации: T1= fact_inner(2, RES, 1, 1). T2=fact_inner(N, F, N1, F1) :- N2 = N1 + 1, F2 = F1 * N2, fact_inner(N, F, N2, F2). Результат: успех. Подстановка: {N=2, F=RES, N1=1, F1=1}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.
6	Резольвента: N2 = 1 + 1, F2 = 1 * N2, fact_inner(3, RES, N2, F2).	Попытка унификации: N2 = 1 + 1 Результат: успех. Подстановка: {N2=2}	Прямой ход. К резольвенте применяется подстановка.
7	Резольвента: F2 = 1 * 2, fact_inner(2, RES, 2, F2).	Попытка унификации: F2 = 1 * 2 Результат: успех. Подстановка: {F2=2}	Прямой ход. К резольвенте применяется подстановка.
8	Резольвента: fact_inner(2, RES, 2, 2).	Попытка унификации: T1=fact_inner(2, RES, 2, 2). T2= fact_inner(N, F, N, F) :- !. Результат: успех. Подстановка: {N=2, F=RES, F=2}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.
9	Резольвента: !.	Попытка унификации: !. Результат: успех.	Прямой ход. Попытка отката завершает использование процедуры. Предложений больше нет.
10	Резольвента пуста.	Все переменные связанны.	Вывод результата на экран. Завершение программы.

Таблица 2. Порядок работы системы.

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков **результат** (и	Дальнейшие действия: прямой ход

		подстановка)	или откат
1	Резольвента: fib(2, RES).	Попытка унификации: T1= fib(2, RES). T2= fib_inner(1, 1, 1) :- !. Результат: неудача. Разные функторы.	Откат, переход к следующему предложению
2	Резольвента: fib(2, RES).	Попытка унификации: T1= fib(2, RES). T2= fib_inner(N, FN, FN1) :- N1 = N - 1, fib_inner(N1, FN_1, FN), FN1 = FN + FN_1. Результат: неудача. Разные функторы.	Откат, переход к следующему предложению
3	Резольвента: fib(2, RES).	Попытка унификации: T1= fib(2, RES). T2= fib(N, FN) :- fib_inner(N, FN, _). Результат: успех. Подстановка: {N=2, FN=RES}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.
4	Резольвента: fib_inner(2, RES, _).	Попытка унификации: T1= fib_inner(2, RES, _). T2= fib_inner(1, 1, 1) :- !. Результат: неудача. 1 != 2	Откат, переход к следующему предложению
5	Резольвента: fib_inner(2, RES, _).	Попытка унификации: T1= fib_inner(2, RES, _). T2= fib_inner(N, FN, FN1) :- N1 = N - 1, fib_inner(N1, FN_1, FN), FN1 = FN + FN_1. Результат: успех. Подстановка: {N=2, FN=RES, FN1=_}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется подстановка.
6	Резольвента: N1 = 2 - 1, fib_inner(N1, FN_1, RES), _ = RES + FN_1.	Попытка унификации: N1 = 2 - 1, Результат: успех. Подстановка: {N1=1}	Прямой ход. К резольвенте применяется подстановка.
7	Резольвента: fib_inner(1, FN_1, RES), _ = RES + FN_1.	Попытка унификации: T1=fib_inner(1, FN_1, RES), T2= fib_inner(1, 1, 1) :- !. Результат: успех. Подстановка: {1=1, FN_1=1, RES=1}	Прямой ход. Содержимое резольвенты заменяется телом найденного правила. К резольвенте применяется

			подстановка.
8	Резольвента: !. _ = 1 + 1.	Попытка унификации: !. Результат: успех.	Прямой ход.
9	Резольвента: _ = 1 + 1.	Попытка унификации: _ = 1 + 1. Результат: успех. Подстановка: {_=2}	
10	Резольвента пуста.	Все переменные связанны.	Вывод результата на экран, завершение работы программы.

Вывод

Таким образом, в программе для вычисления факториала эффективность достигается за счет использования хвостовой рекурсии. В программе для вычисления чисел Фибоначчи эффективность достигается за счет того, что вместо вычисления N-1 и N-2 чисел Фибоначчи с самого начала (т. е. повторно вычисляются числа, которые уже были вычислены), вычисляется сразу два числа Фибоначчи: то, которое необходимо найти и следующее за ним.