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»

Исправление ошибок 17, 18, 19, 20 лабораторных работ

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

Москва, 2020 г.

Исправление ошибок 17-ой лабораторной работы

Замечание

А нельзя результат сразу определить в заголовке, а потом убедиться в его правильности! Исправьте тексты программ и одну табл.

Исправленные тексты программ

Реализация программы без использования отсечения

```
domains
num1, num2, num3, result = integer
predicates
max_two(num1, num2, result).
max_three(num1, num2, num3, result).
clauses
max_two(Num1, Num2, Num1) :- Num2 <= Num1.

max_two(Num1, Num2, Num2) :- Num2 > Num1.

max_three(Num1, Num2, Num3, Num1) :- Num1 >= Num2, Num1 >= Num3.

max_three(Num1, Num2, Num3, Num2) :- Num2 > Num1, Num2 > Num3.

max_three(Num1, Num2, Num3, Num3) :- Num3 >= Num2, Num3 > Num1.
```

Реализация программы с использованием отсечения

```
domains
num1, num2, num3, result = integer
predicates
max_two(num1, num2, result).
max_three(num1, num2, num3, result).
clauses
max_two(Num1, Num2, Num1) :-
Num1 > Num2, !.

max_two(_, Num2, Num2).
```

```
max_three(Num1, Num2, Num3, Num1) :-
Num1 > Num2, Num1 > Num3, !.

max_three(_, Num2, Num3, Num2) :-
Num2 > Num3, !.

max_three(_, _, Num3, Num3).
```

Исправленная таблица для программы с использованием отсечения

Порядок работы системы для 4 примера:

```
goal

        max_three(2, 2, 2, Result).
%Вывод:

        Result=2
        1 Solution
```

№ 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	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_two(Num1, Num2, Num1). Неудача (функторы max_three и max_two не равны).	Прямой ход к следующему предложению.
2	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_two(_, Num2, Num2). Неудача (функторы max_three и max_two не равны).	Прямой ход к следующему преложению.

3	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(Num1, Num2, Num3, Num1). Успех. Подстановка 2 = Num1, 2 = Num2, 2 = Num3, Result = Num1.	Прямой ход к 2 > 2.
4	2 > 2, 2 > 2, Result = 2, !.	T1 = 2 > 2. Знак > имеет смысл сравнения, поскольку с обеих сторон от знака находятся конкретные значения. Неудача (2 !> 2).	Откат к предыдущему состоянию резольвенты: max_three(2, 2, 2, Result), Реконкретизация 2, 2, 2, Result.
5	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(_, Num2, Num3, Num2). Успех. Подстановка 2 = Num2, 2 = Num3, Result = Num2.	Прямой ход к 2 > 2.
6	2 > 2, Result = 2, !.	T1 = 2 > 2. Знак > имеет смысл сравнения, поскольку с обеих сторон от знака находятся конкретные значения. Неудача (2 !> 2).	Откат к предыдущему состоянию резольвенты: max_three(2, 2, 2, Result), Реконкретизация 2, 2, Result.
7	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(_, _, Num3, Num3). Успех. Подстановка 2 = Num3, Result = Num3.	Прямой ход к Result = 2.
8	Result = 2.	T1 = Result = 2. Знак = имеет смысл присвоения, поскольку с одной из сторон от знака стоит свободная переменная. Подстановка Result = 2.	Вывод: Result = 2. Откат к предыдущему состоянию резольвенты: max_three(2, 2, 2, Result), Реконкретизация 2, 2, 2, Result.

9	max_three(2, 2, 2, Result). конец clauses, опустошение резольвенты, завершение работы.		

Замечание

Каково назначение использования алгоритма унификации?

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

Исправление

Назначение использования алгоритма унификации двух термов состоит в том, чтобы подобрать нужное в данный момент правило.

Замечание

Каков результат работы алгоритма унификации?

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

Результатом – не подстановка??

Исправление

Результатом использования алгоритма унификации может быть успех согласования базы знаний и вопроса, в качестве побочного эффекта формируется подстановка; или может быть тупиковая ситуация (неудача).

Исправление ошибок 18-ой лабораторной работы

Замечание

factorial_help: А нельзя в 1-м правиле Rezult определить в заголовке?

Factorial A Help не всегда =1??? Можно прямо в терме!

Исправленный текст программы n!

```
domains

num, help, result = integer

predicates

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

clauses

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

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


factorial(0, 1) :-!.

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

Замечание

Таблица: T1 = factorial(0, Result);

T2 = factorial(0, 1). Успех.

Подстановка 0 = 0, Result = 1 константы в подстановку не заносятся!

И что на этом работа системы заканчивается??? ИСПРАВИТЬ!!

Исправленная таблица для программы n!

Порядок работы системы для 3 примера:

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

```
Result=1
1 Solution
```

№ 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). 0 = 0, Result = 1 успех (подобрано знание) => Подстановка {Result = 1}.	Проверка тела правила factorial(0, 1).
4	!	Удаление из памяти альтернативных путей унификации цели Успех - ответ - «Result = 1», метка на правиле factorial(0, 1).	Других альтернатив нет => система завершает работу с единственным результатом - «Result = 1».

Замечание

fib_help(_, Second, Help, Num, Result) : проверки Help = Num, Result = Second можно перенести в заголовок!

Исправленный текст 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, Num, Num, 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) :-
fib_help(1, 1, 1, Num, Result).
```

Исправление ошибок 19-ой лабораторной работы

Замечание

Не поняла: Где CLAUSES?

Исправление

Прошу прощения, при конвертации из pdf в doc clauses съехали на предыдущую строку

Исправленный текст программы «найти длину списка (по верхнему уровню)»

```
domains
```

```
        lst = integer*
        count, sum = integer
        predicates

        len_list(lst, count).
        len_list_help(lst, count Help, count Result).
clauses

        len_list_help([], Help, Help).
        len_list_help([_|T], Help, Answer) :-
        Next_help = Help + 1, len_list_help(T, Next_help, Answer).

        len_list(List, Answer) :- len_list_help(List, 0, Answer).
```

Замечание

найти сумму элементов числового списка, стоящих на нечетных позициях

А нельзя в заголовке выделить сразу два элемента?

Исправленный текст программы «найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)»

```
domains

        lst = integer*
        count, sum = integer
        predicates

        sum_odd_list(lst, sum).
        sum_odd_list_help(lst, sum Help, sum Answer).
clauses

        sum_odd_list_help([], Help, Help).

        sum_odd_list_help([_|[]], Help, Help) :-!.

        sum_odd_list_help([_, Next_H|T], Help, Answer) :-
        New_help = Help + Next_H,
        sum_odd_list_help(T, New_help, Answer).

        sum_odd_list(List, Answer) :- sum_odd_list_help(List, 0, Answer).
```

Исправление ошибок 20-ой лабораторной работы

Замечание

сформировать список из элементов числового списка, больших заданного значения

list_create_help([], _, Help, Help). А результат не пустой список?

А незя формирование нового списка перенести в заголовок и Help не нужен?

Исправленный текст программы «сформировать список из элементов числового списка, больших заданного значения».

```
domains

        lstI = integer*
        number = integer
predicates

        list_create(lstI, number, lstI)
clauses
        list_create([], _, []).

        list_create([H|T], Number, [H|T2]) :-
        H > Number, list_create(T, Number, T2), !.

        list_create([_|T], Number, T2) :-
        list_create(T, Number, T2).
```

Замечание

Объясните обязательность ! во 2-м правиле

Объяснение ! во 2-ом правиле

Предикат отсечения нужен во 2-ом правиле, чтобы отсечь бесперспективный путь доказательства, в данном случае это 3-е правило list_create([_|T], Number, T2), поскольку если элемент списка больше заданного значения, необходимо добавить его в результирующий список, применив правило 2, а третье правило служит для тех случаев, когда элемент списка не больше заданного значения, соответственно, применять 3-е правило, если 2-е правило успешно, не

нужно.

В противном же случае, мы получим несколько решений, вместо одного необходимого.

Замечание

список из элементов, стоящих на нечетных позициях: Выделяйте сразу по два элемента!

Исправленный текст программы «сформировать список из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0)»

```
domains

        lstI = integer*
        number = integer
predicates

        odd_list_help(lstI, lstI, lstI)
        odd_list(lstI, lstI)
clauses

        odd_list_help([], Help, Help).

        odd_list_help([_|[]], Help, Help) :-!.

        odd_list_help([_, Next_H|T], Help, Answer) :-
        odd_list_help(T,  [Next_H|Help], Answer).

        odd_list(List, Answer) :- odd_list_help(List, [], Answer).
```

Замечание

удалить заданный элемент Зачем reverse?

Объяснение

Поскольку элементы последовательно добавляются в голову результирующего списка, полученный результат будет перевернутый. Поэтому, в правиле, предназначенном для выхода из рекурсии вызывается правило reverse(Help, Answer), которое переворачивает список.

Но можно обойтись и без reverse.

Исправленный текст программы «удалить заданный элемент из списка (один или все вхождения)»

```
domains
        lstI = integer*
        number = integer
predicates
        delete_elem_from_list(lstI, number, lstI)
clauses
        delete_elem_from_list([], _, []).

        delete_elem_from_list([H|T], Number, [H|T3]) :-
        H <> Number, delete_elem_from_list(T, Number, T3),!.

        delete_elem_from_list([_|T], Number, T3) :-
        delete_elem_from_list(T, Number, T3).
```