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, 20 лабораторных работ

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

Москва, 2020 г.

Исправление ошибок по 17-ой лабораторной работе

Замечание

Табл шаг8:

Знак = имеет смысл присвоения, в Prolog такого нет!!! поскольку с одной из сторон от знака стоит свободная переменная. Подстановка Result = 2.

Вывод:

Result = 2. Откат к предыдущему состоянию резольвенты: На каком основании??,

max_three(2, 2, 2, Result), Реконкретизация 2, 2, 2, Result. И почему конец? на след шаге?

Исправленная таблица

№ 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).	Нет альтернативных путей унификации цели. 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 конкретизации переменной Result. Подстановка Result = 2.	Вывод: Result = 2. Нет альтернативных путей унификации цели. 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, нет альтернативных путей унификации цели, откат к предыдущему состоянию резольвенты. Резольвента пуста => завершение работы.		

Замечание

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

Правило – это предложение, а сравниваются термы. – в материалах это есть!!

Для подбора ЗНАНИЯ (оно же в заголовке?!) Вы в 18 в таблице пишете: (подобрано знание) а в ответе?

Исправление

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

Исправление ошибок по 18-ой лабораторной работе

Замечание

В табл 4ш:Удаление из памяти альтернативных путей унификации цели Успех, А ВЫ думаете они храняться , а как их система находит?

У Вас превратное впечатление, что многое происходит само собой!

Вы пишете:Других альтернатив нет , а как система это понимает? ОТВЕТТЕ!

Ответ

Знания по умолчанию просматриваются сверху вниз, и, соответственно, когда метка находится в конце Базы Знаний, система понимает, что альтернативных путей унификации цели нет.

Поскольку в лекции №2 говорилось, что предекат отсечения ! (cut) «отсекает в определенном случае бесперспективные пути доказательства», предположу, что он переводит метку в конец Базы Знаний. Таким образом, система поймет, что альтернативных путей унификации цели нет и выполнит откат к предыдущему состоянию резольвенты.

Исправление ошибок по 20-ой лабораторной работе

Замечание

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.

А как ВЫ получите много рез-тов? Вы уйдете в рекурсию, и до ! дойдете только с пустым!!! Списком, а если Вы с ним попадете на 3-е правило, то разбить на части список не удастся – выход СОГЛАСНЫ?

Ответ

Согласен, но не совсем. Да, действительно, когда мы с пустым списком попадаем на 3-е правило, разбить на части список не удастся, но система выполнит откат лишь до предыдущего состояния резольвенты, и дальше продолжит искать альтернативные варианты, а не завершит работу. Чтобы объяснить, как я получу много результатов, приведу порядок работы системы на следующему примере:

Текст программы без предиката !

```
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).

goal
        list_create([1, 2, 3], 2, Result)
```

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	list_create([1, 2, 3], 2, Result)	T1 = list_create([1, 2, 3], 2, Result); T2 = list_create([[], _, []]). Неудача, [] не равно [1, 2, 3].	Прямой ход к следующему предложению.
2	list_create([1, 2, 3], 2, Result)	T1 = list_create([1, 2, 3], 2, Result); T2 = list_create([H|T], Number, [H|T2]). Успех (подобрано знание) => {H = 1, T = [2, 3], 2 = Number, Result = [1|T2]})	Проверка тела правила правила 2: 1 > 2, list_create([2, 3], 2, T2).

3	1 > 2 list_create([2, 3], 2, T2)	T1 = 1 > 2. Знак > имеет смысл сравнения поскольку с обеих сторон от зна- ка находятся конкрет- ные значения. Неудача (1 не больше 2).	Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: list_create([1, 2, 3], 2, Result)
4	list_create([1, 2, 3], 2, Result)	T1 = list_create([1, 2, 3], 2, Result); T2 = list_create([_|T], Number, T2). Успех (подобрано значение) => {T = [2, 3], Number = 2, Result = T2}.	Проверка тела правила 3: list_create([2, 3], 2, Result). Поиск с начала предложений
5	list_create([2, 3], 2, Result)	T1 = list_create([2, 3], 2, Result); T2 = list_create([], _, []). Неудача ([2, 3] не равно []).	Прямой ход к следующему предложению.
6	list_create([2, 3], 2, Result)	T1 = list_create([2, 3], 2, Result); T2 = list_create([H|T], Number, [H|T2]). Успех (подобрано знание) => {H = 2, T = [3], Number = 2, Result = [2|T2]}.	Проверка тела правила 2: 2 > 2, list_create([3], 2, T2).
7	2 > 2 list_create([3], 2, T2)	T1 = 2 > 2. Знак > имеет смысл сравнения поскольку с обеих сторон от знака находятся конкретные значения. Неудача (2 не больше 2).	Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: list_create([2, 3], 2, Result)

8	list_create([2, 3], 2, Result)	T1 = list_create([2, 3], 2, Result); T2 = list_create([_|T], Number, T2). Успех (подобрано знание) => {T = [3], Number = 2, Result = T2}.	Проверка тела правила 3: list_create([3], 2, Result).
9	list_create([3], 2, Result)	T1 = list_create([3], 2, Result); T2 = list_create([], _, []). Неудача ([3] не равно []).	Прямой ход к следующему предложению.
10	list_create([3], 2, Result)	T1 = list_create([3], 2, Result); T2 = list_create([H|T], Number, [H|T2]). Успех (подобрано знание) => {H = 3, T = [], Number = 2, Result = T2}.	Проверка тела правила 2: $3 > 2$, list_create([], 2, Result).
11	$3 > 2$ list_create([], 2, Result)	T1 = $3 > 2$. Знак $>$ имеет смысл сравнения поскольку с обеих сторон от знака находятся конкретные значения. Успех.	Прямой ход к list_create([], 2, Result).
12	list_create([], 2, Result)	T1 = list_create([], 2, Result); T2 = list_create([], _, []). Успех (подобрано знание) => {[] = [], 2 = _, Result = []}.	Пустое тело заменяет цель в резольвенте.

13	Пусто	успех - ответ - «Result = [3]», метка на правиле 1.	Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: list_create([], 2, Result). Прямой ход к следующему предложению.
14	list_create([], 2, Result)	T1 = list_create([], 2, Result); T2 = list_create([H|T], Number, [H|T2]). Неудача ([] не равно [H|T]).	Прямой ход к следующему предложению.
15	list_create([], 2, Result)	T1 = list_create([], 2, Result); T2 = list_create([_|T], Number, T2). Неудача ([] не равно [_|T]).	Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: list_create([3], 2, Result). Прямой ход к следующему предложению.
16	list_create([3], 2, Result)	T1 = list_create([3], 2, Result); T2 = list_create([_|T], Number, T2). Успех (подобрано знание) => {T = [], Number = 2, Result = T2}.	Проверка правила 3: list_create([], 2, Result).
17	list_create([], 2, Result)	T1 = list_create([], 2, Result); T2 = list_create([], _, []). Успех (подобрано знание) => {Result = []}.	Пустое тело заменяет цель в резольвенте.

18	Пусто	успех - ответ - «Result = []», метка на правиле 1.	Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: list_create([], 2, Result). Прямой ход к следующему предложению.
...	...	...	...

Отсюда видно, что результатов получается больше чем 1. Провел тестирование в программе, результат тот же.

Замечание

список из элементов, стоящих на нечетных позициях: Выделяйте сразу по два элемента! За один шаг можно выделить не только голову! А начало из 2-х элементов, ПОПРОБУЙТЕ,

Ответ

Извините, но тут какая-то ошибка, я же попробовал и прислал вам уже исправленный текст программы, где в заголовке третьего правила: odd_list_help([_, Next_H|T], Help, Answer) как раз и происходит выделение СРАЗУ головы и второго элемента списка.

Текст программы:

```
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

Зачем reverse? На след-ий шаг рекурсии уходит хвост, добавляйте в голову хвоста, и формируйте результат в заголовке правила!! Исправьте! Пришлите текст3-ей прогр-мы!

Ответ

Да, я это осознал и в прошлый раз вам уже прислал исправленный вариант без reverse, где как раз и происходит добавление элемента в голову хвоста (аналогично примеру из 3 лекции с append). Поправьте, если я чего-то не понимаю.

```
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).
```