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 №17

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

Москва, 2020 г.

Цель работы

Изучить способы организации эффективных программ на Prolog, особенности использования системных предикатов и порядок выполнения программ с их использованием.

Задачи работы

Приобрести навыки эффективного описания предметной области с использованием фактов и правил.

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

Задание

Ответить на вопросы:

- Какое первое состояние резольвенты?
- В каком случае система запускает алгоритм унификации? (т.е. Как эту необходимость на формальном уровне распознает система?)
- 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?

В одной программе написать правила, позволяющие найти

1. Максимум из двух чисел
   (a) без использования отсечения,
   (b) с использованием отсечения;

2. Максимум из трех чисел

   (a) без использования отсечения,

   (b) с использованием отсечения;

   Убедиться в правильности результатов.

   Для каждого случая пункта 2 обосновать необходимость всех условий тела.

   Для одного из вариантов **ВОПРОСА** и каждого варианта **задания 2 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 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.
```

Обоснование необходимости всех условий тел из пункта 2.

Поскольку стоит задача найти максимум из трех чисел, то алгоритм нахождения максимума без использования механизма отсечения выглядит так:

1. Проверить, является ли первое число (*Num1*) не меньше чем остальные 2 ($>=$). Если да, то вывести это число.
2. Проверить, является ли второе число (*Num2*) наибольшим ($>$). При этом необходимо учесть, что первое и вторые пункты алгоритма должны быть взаимоисключающими, поэтому во втором случае в знаках неравенства стоят строгие знаки. Если является, то вывести это число.
3. Аналогично необходимо проверить, является ли третье число (*Num3*) больше первого ($>$) и не меньше второго ($>=$). Такая расстановка знаков обусловлена все тем же смыслом, что и в пункте 2 (во втором пункте стоят строгие знаки неравенства $=>$ в третьем должны быть не строгие и, наоборот, с первым пунктом).

Тесты

Ниже приведены примеры для задания 1.

Пример 1:

```
goal
        max_two(1, 2, Result).
%Вывод:
        Result=2
        1 Solution
```

Пример 2:

```
goal
        max_two(2, 2, Result).
%Вывод:
        Result=2
        1 Solution
```

Пример 3:

```
goal
        max_two(2, 1, Result).
%Вывод:
        Result=2
        1 Solution
```

Ниже приведены примеры для задания 2.

Пример 1:

```
goal

        max_three(1, 2, 3, Result).

%Вывод:

        Result=3
        1 Solution
```

Пример 2:

```
goal

        max_three(1, 2, 2, Result).

%Вывод:

        Result=2
        1 Solution
```

Пример 3:

```
goal

        max_three(5, 2, 3, Result).

%Вывод:

        Result=5
        1 Solution
```

Пример 4:

```
        goal

        max_three(2, 2, 2, Result).

        %Вывод:

        Result=2

        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	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_two(Num1, Num2, Result). Неудача (функторы max_three и max_two не равны).	Прямой ход к следующему предложению. Аналогичная ситуация со следующем предложением. Прямой ход к следующему предложению.
3	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(Num1, Num2, Num3, Result). Успех. Подстановка 2 = Num1, 2 = Num2, 3 = Num3, Result = Result.	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. Успех.	Прямой ход к 2>=2.
5	2>=2, Result = 2.	Знак >= имеет смысл сравнения, поскольку с обеих сторон от знака находятся конкретные значения. Успех.	Прямой ход к Result = 2.

6	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 : 2 >=2, Result = 2. Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: 2>=2, 2>=2, Result = 2. Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: max_three(2, 2, 2, Result). Реконкретизация Result, 2, 2, 2.
7	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(Num1, Num2, Num3, Result). Успех. Подстановка 2 = Num1, 2 = Num2, 2 = Num3, Result = Result.	Прямой ход к 2 > 2.
8	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 Result, 2, 2, 2.

9	max_three(2, 2, 2, Result).	T1 = max_three(2, 2, 2, Result); T2 = max_three(Num1, Num2, Num3, Result). Успех. Подстановка 2 = Num1, 2 = Num2, 2 = Num3, Result = Result.	Прямой ход к 2 >= 2.
10	2 >= 2, 2 > 2, Result = 2.	T1 = 2 >= 2. Знак >= имеет смысл сравнения, поскольку с обеих сторон от знака находятся конкретные значения. Успех.	Прямой ход к 2 > 2.
11	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: 2 >= 2, 2 > 2, Result = 2. Нет альтернативных путей унификации цели. Откат к предыдущему состоянию резольвенты: max_three(2, 2, 2, Result). Реконкретизация Result, 2, 2, 2.
12	max_three(2, 2, 2, Result). конец clauses, опустошение резольвенты, завершение работы.		

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).
```

Обоснование необходимости всех условий тел из пункта 2.

Поскольку стоит задача найти максимум из трех чисел, то алгоритм нахождения максимума с использования механизма отсечения выглядит так:

1. Проверить, является ли первое число (*Num1*) наибольшим ($>$). Если да, то вывести это число, и завершить поиск других решений с помощью предиката отсечения !.
2. Аналогично проверить, является ли второе число (*Num1*) больше третьего ($>$). При этом необходимость в проверке того, что второе число не меньше первого отпадает, поскольку это следует из того, что первое условие не выполнилось, иначе дальнейшая работа программы была бы прекращена предикатом !. Если да, то вывести это число, и завершить поиск других решений с помощью предиката отсечения !.
3. Если же ни первое ни второе условия не выполнились, то значит, что ни первое и ни второе числа не наибольшие. Значит наибольшее третье. Выводим третье число.

Тесты

Тесты и их результаты полностью совпадают с тестами для реализации программы без использования отсечения.

Порядок работы системы для 4 примера 2 задания:

№ 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, Result). 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, Result). 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, Result). Успех. Подстановка 2 = Num1, 2 = Num2, 2 = Num3, Result = Result.	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, Result). Успех. Подстановка 2 = Num2, 2 = Num3, Result = Result.	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, Result). Успех. Подстановка 2 = Num3, Result = Result.	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.		

Выводы

За счет чего может быть достигнута эффективность работы системы?

Из проделанной лабораторной работы можно сделать вывод, что 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.

Это и было продемонстировано при реализации 2 задания двумя способами: с использованием отсечения было затрачено 9 шагов на поиск решения, в то время как без использования отсечения было затрачено 12 шагов.

Ответы на вопросы

Какое первое состояние резольвенты?

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

В каком случае система запускает алгоритм унификации? (т.е. Как эту необходимость на формальном уровне распознает система?)

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

Каково назначение использования алгоритма унификации?

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

Каков результат работы алгоритма унификации?

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.