ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ
«АКАДЕМИЧЕСКАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ПРИ ПЕРМСКОМ ГОСУДАРСТВЕННОМ УНИВЕРСИТЕТЕ»

Пастухова Г.В.

ТЕОРИЯ АЛГОРИТМОВ

Учебное пособие

Рекомендовано УМО по математике
Волго-Вятского региона в
качестве учебного пособия

Пермь
2013

Рецензент:

Профессор кафедры математического моделирования систем и процессов Пермского государственного технического университета, доктор физико-математических наук М.Б. Гитман

Автор: Г.В. Пастухова.

Содержание

Уверяю вас, единственный способ
избавиться от драконов –
это иметь своего собственного.
Е.Л. Шварц «Убить дракона»

Введение

Удивительно, как повлиял XX век на математику. Это время и великих достижений, и великих разочарований. Основной удар пришелся на основания математики или метаматематику, которая в своем составе имела теорию множеств, логику и теорию алгоритмов. Последняя, своим появлением обязана основному разочарованию того периода – теореме Геделя. Если вкратце, то суть теоремы в «круговой порочности» тех разделов математики, которые в своем построении используют натуральный ряд: такие разделы математики не являются, по сути, объективным отражением мира.

Не смотря на относительную молодость теории алгоритмов (началом принято считать 1930-1936 гг.), термин «алгоритм» появился многие столетия назад. Он обязан своим происхождением великому ученому средневекового Востока Мухаммеду ибн Мусса ал-Хорезми (Мухаммед сын Муссы из Хорезма, ок. 783 – ок. 850). В 1983 году в городе Ургенч – областном центре Хорезмской области Узбекистана, отмечалось 1200-летие дня рождения Хорезми организацией Международного симпозиума «Алгоритм в современной математике и ее приложениях» академиком А.П. Ершовым, на который приехали и некоторые создатели теории алгоритмов.

Рис.1 Приезд С.К. Клини и его супруги на симпозиум

Ал-Хорезми известен как математик, астроном и географ. В девятом веке он переселился в Багдад, где с 815 года возглавлял «Дом мудрости» – хранилище рукописей, созданное арабскими халифами.

Многообразные научные интересы ал-Хорезми относились к математике, теоретической и практической астрономии, географии и истории. Наибольшую славу ал-Хорезми принесли его математические труды. Он является автором двух знаменитых трактатов: по арифметике и алгебре. Подлинный арабский текст арифметического трактата утерян. Европейцы познакомились с его содержанием по латинскому переводу, выполненному в XII веке. В трактате рассматривается запись чисел в десятичной системе счисления, заимствованная из «индийской арифметики», способы вычисления с натуральными числами и дробями, правило извлечения квадратного корня. Латинский перевод трактата не имеет заглавия. Он начинается словами «Dixit algorizmi» – «Сказал Алгоризми». Почему то, именно та часть имени, которая фактически является географическим названием – ал-Хорезми (в латинской трактовке звучало как Algorizmi) и осталась при переводе сочинения, которое получило большую популярность в Европе. Имя автора стало нарицательным: «алгорисмом» или «алгоритмом» средневековые математики называли всю «индийскую арифметику», основанную на десятичной позиционной системе счисления. Позднее это слово стало обозначать любую систему вычислений, выполненных по определенному правилу. В современной математике термин «алгоритм» означает точное предписание о выполнении вычислений в строгом порядке.

Алгебраический трактат ал-Хорезми «Китаб аль-джебр аль-мукабала» («Книга о восстановлении и противопоставлении») сохранился до настоящего времени в арабской копии. В этом сочинении впервые алгебра рассматривалась как самостоятельный раздел математики, изучающий общие методы решения линейных и квадратных уравнений. Для преобразования квадратных уравнений к определенному виду ал-Хорезми вводит два особых действия: аль-джебр (восполнение) и аль-мукабала (противопоставление). Аль-джебр

состоит в перенесении отрицательного члена из одной части уравнения в другую. От этого термина произошло современное слово «алгебра». Аль-мукабала состоит в сокращении равных членов в обеих частях уравнения.

Глава 1. Интуитивное представление об алгоритмах

1.1. Понятие алгоритма

Почему танец красив? Ответ: потому что это
несвободное движение, потому что весь
глубокий смысл танца именно в абсолютной,
эстетически-подчиненной, идеальной
несвободе. И если верно, что наши предки
отдавались танцу в самые вдохновенные
моменты своей жизни, то это значит только одно:
инстинкт несвободы издревле органически присущ человеку.
Е.И. Замятин «Мы»

Понятие алгоритма такое же древнее, как и сама математика. Хотя в современной науке это понятие вышло за рамки математики, до двадцатого столетия оно ассоциировалось только с математическими проблемами. Оно является фундаментальным понятием так же, как понятие «число» или «множество» и его суть проще пояснить на конкретных примерах. Вместо слова «алгоритм» мы часто употребляем, такие синонимы как «метод», «способ», «правило». Например, метод Гаусса, правило сложения двух обыкновенных дробей, правило буравчика, правила дорожного движения и т.д.

На интуитивном уровне сущность алгоритма разные источники разъясняют примерно одинаково:

[8]. «Алгоритм – точное предписание, которое задает вычислительный процесс (называемый в этом случае алгоритмическим), начинающийся с произвольного исходного данного и направленный на получение полностью определенного этим исходным данным результата… Вообще говоря, не предполагается, что результат будет обязательно получен: процесс применения алгоритма к конкретному исходному данному может оборваться безрезультатно или же продолжаться бесконечно».

[7]. «В математике принято понимать под «алгоритмом» точное предписание, определяющее вычислительный процесс, ведущих от варьируемых исходных данных к исходному результату».

[10]. «Алгоритм – это детерминированная процедура, которую можно применить к любому элементу некоторого класса символических входов, которая для каждого такого входа дает, в конце концов, соответствующий символический выход».

[12]. «Алгоритм – точное предписание о выполнении в определенном порядке некоторой системы операций, позволяющее решать совокупность задач определенного класса».

Можно привести очень много примеров из математики, где решение задачи происходит «без всякой изобретательности», чисто механически, следуя перечню указаний. Рассмотрим вначале два примера из античной математики.

Пример 1.1.1 Греческий астроном и географ из Александрии Эратосфен около 2000 лет назад предложил следующий метод отыскания простых чисел в натуральном ряду:

а) Обозревай натуральные числа в порядке возрастания, найди самое меньшее не вычеркнутое число, большее единицы. Пусть это $p>1$. Оно простое. Далее переходи к следующему указанию.

б) Вычеркивай все числа кратные p и переходи к выполнению указания а) начиная с числа $p+1$.

Несмотря на древность изобретения Эратосфена, современные электронно-вычислительные машины пользуются этим методом при пересчете простых чисел в заданном промежутке.

Ну и классическим примером алгоритма является алгоритм Евклида, не обойдем вниманием его и мы:

Пример 1.1.2 Алгоритм Евклида для нахождения наибольшего общего делителя двух целых положительных чисел a и b. Его также нетрудно описать в виде нескольких предписаний:

а**)** Обозревай заданные два числа *a* и *b*. Переходи к следующему указанию.

б**)** Сравни обозреваемые числа $a=b$, $a<b$, $a>b$. Переходи к следующему указанию.

в**)** Если обозреваемые числа равны, то каждое из них дает искомый результат. Процесс вычислений прекращается. В противном случае переходи к следующему указанию.

г**)** Если $a<b$, то переставь их местами и переходи к следующему указанию.

д**)** Вычитай второе число из первого и обозревай два числа: вычитаемое и остаток. Переходи к указанию б).

Каковы бы ни были два числа *a* и *b*, за конечное число применений а) – д), всегда получим желаемый результат.

1.2. Общие свойства алгоритмов

Для любого алгоритма характерны следующие свойства:

1) Детерминированность[1)] алгоритма. Описываемый метод должен быть общедоступен и настолько точно описан, что не остается место произволу. Его можно сообщить другому лицу в виде конечного списка указаний о том, как надлежит поступить в любой возникшей ситуации. При этом система величин, получаемых в данный момент времени, однозначно определяется величинами, полученными в предшествующие моменты, и эта однозначность не зависит от произвола действующего лица, а также от того, кто его выполняет**.**

2) Массовость алгоритма. Алгоритм служит для решения целой серии (в принципе бесконечной) однотипных задач.

3) Дискретность алгоритма.

[1)] *Determinare*(лат) – определять, обуславливать

Алгоритмический процесс происходит дискретно во времени, начало которого задается конечной системой величин, полученных в предыдущих шагах.

4) Направленность (результативность) алгоритма.

Алгоритмический процесс, будучи применен к любой задаче заданного типа, должен завершиться через конечное число шагов и выдать результат.

5) Элементарность шагов (доступность) алгоритма. Система указаний должна состоять из команд настолько простых и элементарных, что процесс вычислений можно было бы поручить и машине.

Уровень «простоты» указаний естественно зависит от «интеллекта» исполнителя. В этом легко убедится, сравнив команды в программах, предназначенных для ЭВМ первого и пятого поколений.

1.3. Способы задания алгоритмов

Наиболее распространенный способ задания алгоритма в математике (и не только) – это словесный. Алгоритм задается в виде конечного перечня простейших указаний, не оставляющих возможности произвола исполнителя. Именно так описаны алгоритмы в примерах 1, 2, 7, 9, 10. Рассмотрим ещё один школьный пример – правило умножения обыкновенных дробей: «чтобы умножить дробь на дробь, умножают числитель на числитель и знаменатель на знаменатель. Первый результат есть числитель произведения, второй – его знаменатель. Если среди сомножителей имеются смешанные числа, то их предварительно обращают в неправильную дробь. Ещё до перемножения можно сокращать любой множитель числителя с любым множителем знаменателя».

Зададим это правило в виде последовательности простейших указаний:

а) Рассмотрим две дроби $p \neq 0$ и $q \neq 0$;

б) Если p или q – смешанные числа, то запишем их в виде неправильной дроби $p = \frac{a}{b}$, $q = \frac{c}{d}$, $(a, b, c, d \in N)$. Причем

НОД (a, b) = 1 и *НОД (c, d)* = 1;

в) Найдем *НОД (a, d) = k* и *НОД (b, c) = r*;

г) Сократим *a* и *d* на общий множитель *k* и получим соответственно числа a_1 и d_1;

d) Сократив *b* и *c* на общий множитель *r* и получим соответственно числа b_1 и c_1;

е) Умножим a_1 и c_1 и получим числитель произведения $s=a_1 \cdot c_1$;

ж) Умножим b_1 и d_1 и получим знаменатель произведения $t=b_1 \cdot d_1$;

з) Число $\frac{s}{t}$ искомое;

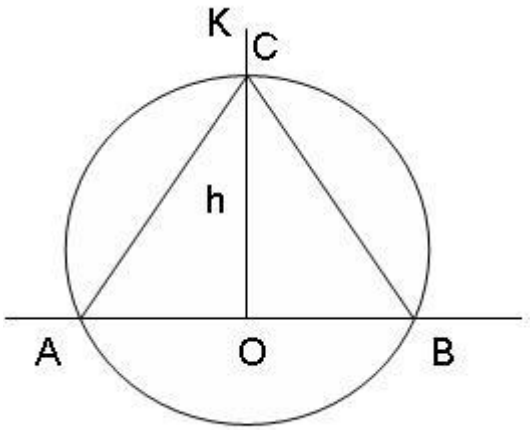
и) Если дробь $\frac{s}{t}$ неправильная (т.е. $s > t$), то превратим его в смешанную дробь;

k) Конец.

Рассмотрим образец описания алгоритма задачи на построение

С помощью циркуля и линейки постройте равнобедренный треугольник по боковой стороне *b* и высоте *h* проведенной к *b*, опишите пошаговое исполнение в виде перечня простейших предписаний.

1) Если $b \leq h$, то задача не имеет решения, иначе шаг 2)



2) Взять произвольную прямую *a*

3) Взять произвольную точку *O* на прямой *a*

4) Из точки *O* восстановить перпендикуляр *OK* к прямой *a*

5) На прямой *OK* из точки *O* отложить отрезок *OC=h*

6) Провести окружность с центром в точке *C* и радиуса *R=b*

7) Окружность пересечет прямую *a* в двух точках *A, B*

8) Соединим точку *C* с точками *A* и *B*

9) Конец.

Контрольная работа №1

1. Опишите в виде перечня элементарных шагов алгоритм построения с помощью циркуля и линейки прямоугольного треугольника, если даны гипотенуза c и разность острых углов $\angle\alpha - \angle\beta = \angle\gamma$, сами углы не даны.

2. Опишите в виде перечня элементарных шагов, и задайте в виде блок-схемы следующие алгоритмы:

1) Построение окружности через три данные различные точки плоскости;

2) Вычисление суммы по конечному набору действительных чисел a_1, $a_2, \ldots, a_n$.

3. Задайте в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Проверка равенства числа a нулю.

2) Целые числа a и b сравнимы по модулю m ($a \equiv b \pmod m$).

4. Опишите в виде перечня указаний и задайте в виде блок-схемы следующие алгоритмы:

1) Перевод десятичной дроби в двоичную дробь.

2) Нахождение наименьшего общего кратного двух целых чисел m и n.

5. Опишите в виде блок-схемы алгоритм, который определял бы для любых двух обыкновенных дробей (положительный или отрицательный) какая из них больше.

6. Задайте в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Нахождение суммы двух натуральных чисел по рекурсии:

$$\begin{cases} a+0=a \\ a+(b+1)=(a+b)+1 \end{cases};$$

2) Проверка кратности целого числа a целому числу b.

7. Опишите в виде перечня указаний и обоснуйте результат следующих алгоритмов:

1) Построение касательной к окружности с центром в точке O и радиусом R из точки A, находившейся вне окружности, используя циркуль и линейку.

2) Разделение данного отрезка на семь равных частей.

8. Опишите в виде перечня указаний и обоснуйте результат следующих алгоритмов:

1) Построение перпендикуляра к отрезку AB в точке B (не продолжая отрезок AB), пользуясь только циркулем и линейкой.

2) Построение треугольника по трем сторонам.

9. Опишите в виде перечня указаний и задайте в виде блок-схемы следующие алгоритмы:

1) Перевод целых чисел с десятичного в двоичное счисление.

2) Сложение двух обыкновенных дробей.

10. Опишите в виде перечня указаний и обоснуйте результат следующих алгоритмов:

1) Разделение прямого угла на три равные части.

2) Нахождение центра вписанной окружности в данный треугольник.

11. Опишите в виде перечня указаний, и задайте в виде блок-схемы следующие алгоритмы:

1) Округление десятично дроби до n знака после запятой.

2) Подстановка при решении системы двух линейных уравнений с двумя неизвестными.

12. Опишите в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Сравнение целых чисел a и b.

2) Проверка целого числа a на простоту.

13. Опишите в виде перечня указаний (элементарных шагов) алгоритм построения с помощью циркуля и линейки равнобедренного треугольника, если известны угол α при равных сторонах и высота h, опущенная из вершины угла α.

14. Опишите в виде перечня указаний и обоснуйте результат следующих алгоритмов:

1) Построение прямоугольного треугольника с острым углом 30^{o}, если задан один из катетов, с помощью циркуля и линейки.

2) Определение общих делителей для любых натуральных чисел a, $b \in \{1,2,3...\}$.

15. Опишите в виде перечня указаний и задайте в виде блок-схемы следующие алгоритмы;

1) Сложение при решении системы двух уравнений (линейных) с двумя неизвестными.

2) Сложение двух чисел с разными знаками.

16. Опишите в виде перечня указаний алгоритм сложения двух положительных обыкновенных дробей (используя НОК).

17. Опишите в виде перечня указаний и обоснуйте результат следующих алгоритмов:

1) Построение треугольника по данной стороне и двум прилежащим к ней углам;

2) Построение касательной к данной окружности из точки А, находящийся вне этой окружности, пользуясь только циркулем и линейкой.

18. Описать в виде перечня указаний и задать в виде блок-схемы следующие алгоритмы:

1) Перевод десятичной дроби в двоичную дробь.

2) Нахождение наименьшего общего кратного двух целых чисел m и n.

19. Опишите в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Проверка на равенство целых чисел a и b.

2) Нахождение остатка от деления целого числа a на целое число b.

20.Задайте в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Проверка a и b на простоту, с разницей между ними равной двум (простые близнецы).

2) Решение квадратного уравнения $ax^2 + bx + c = 0$ методом выделением полного квадрата.

21.Задайте в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1) Проверка неравенства числа A на положительность.

2) Получение делителей целого числа a (включая 1 и самого числа a).

22.Задайте в виде перечня указаний и изобразите в виде блок-схемы следующие алгоритмы:

1)Определение делителей любого натурального числа a ($a \in \{1,2,3,4,5\}$).

2) Сравнение по *mod* 7 двух любых натуральных чисел a и b.

23. С помощью циркуля и линейки разделите угол $\alpha = 45^o$ на три равные части, опишите пошаговое исполнение в виде перечня простейших предписаний.

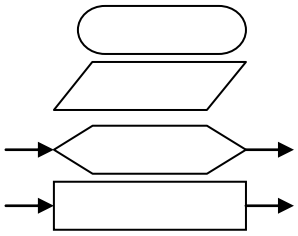
24. Опишите в виде блок – схемы алгоритм, который вычисляет сумму любых двух целых чисел.

25. С помощью циркуля и линейки постройте треугольник по стороне, медиане, проведенной к одной из двух других сторон, и углу между данными стороной и медианой, опишите пошаговое исполнение в виде перечня простейших предписаний.

26. С помощью циркуля и линейки постройте треугольник по углу, высоте и биссектрисе, проведенным из вершины этого угла, опишите пошаговое исполнение в виде перечня простейших предписаний.

Задание алгоритма в виде конкретной программы для ЭВМ – так же является описательной формой, но только на искусственном языке (Бейсик, Фортран, Паскаль, Пролог…). Очень часто такому заданию предшествует графическая форма задания алгоритма (в виде блок-схемы). Это форма имеет ряд преимуществ благодаря наглядности, обеспечивающая в частности, высокую «читаемость » алгоритма и явное отображение управления в нём

Блок-схема – это ориентированный граф, указывающий порядок исполнения команд алгоритма. Вершины такого графа изображаются в виде некоторых геометрических фигур:

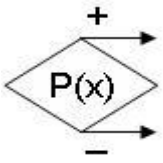


Эллипс: обозначение начала и конца алгоритма.

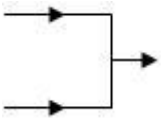
Параллелограмм: обозначение ввода и вывода данных.

Шестиугольник: обозначение цикла «пока».

Прямоугольник: обозначение процедур.

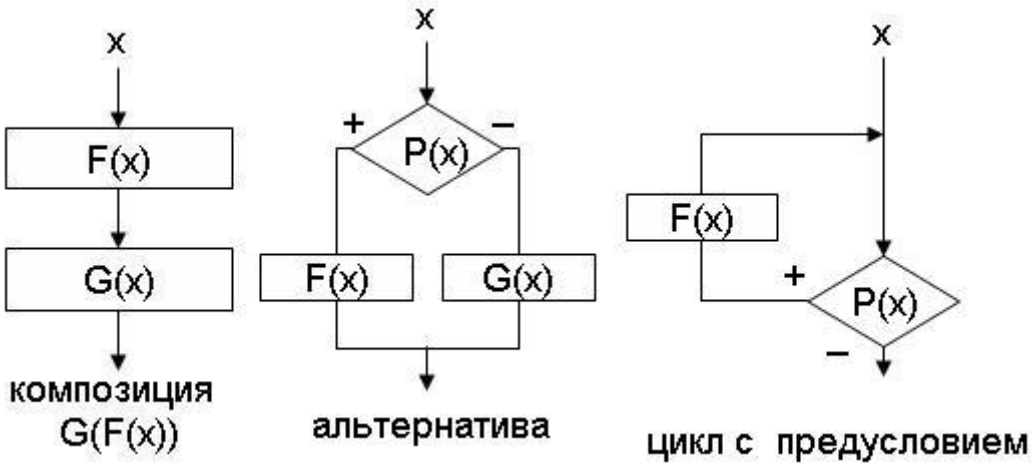


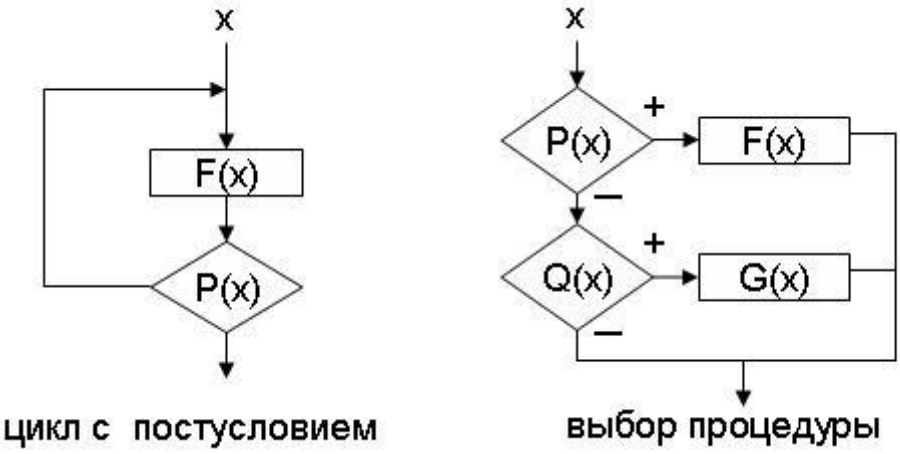
Ромб: обозначение условия «предикатная вершина».



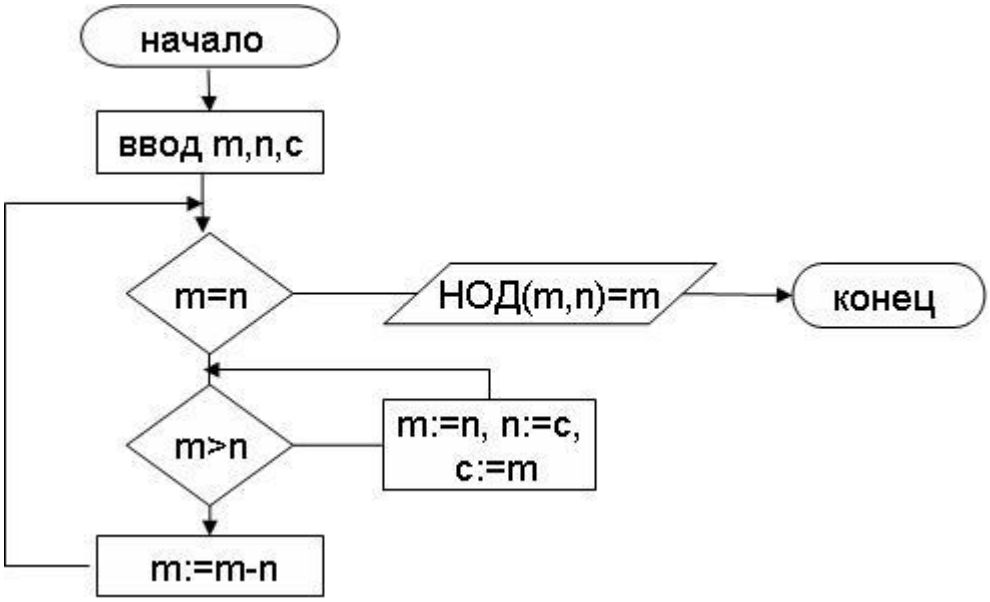
Развилка: обозначение условия «объединяющая вершина».

Блок-схемы могут содержать пять элементарных блок-схем:





В качестве примера рассмотрим блок-схему алгоритма нахождения наибольшего общего делителя двух положительных чисел *m* и *n*.



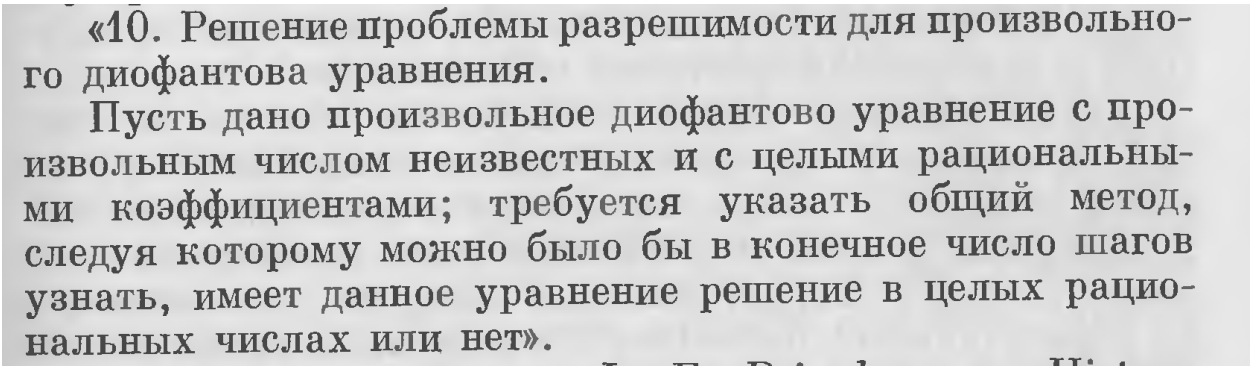
Блок-схема любой программы, как и сама программа также является алгоритмом решения той или иной задачи. Многие столетия для математиков, и не только для них, не было нужды давать точное определение понятию «алгоритма» – хватало интуитивного, лишь бы был этот самый алгоритм. Но когда потребовалось доказать принципиальную невозможность его построения – надо было точно знать, отсутствие чего доказывается. И действительно, та или иная математическая проблема вообще не имеет решения или пока нет возможности ее доказать? А проблема была такая…

1.4. Необходимость уточнения понятия алгоритма

К началу XX века накопились проблемы, в частности в теории множеств, которые требовали консолидации усилий всех математиков. Началось все с «парадокса Рассела», который при внимательном рассмотрении так же устроен, как и «парадокс лжеца», и «парадокс брадобрея».

Д. Гильберт собрал математиков всего мира на II Международном Конгрессе математиков, проходившем в Париже с 6-12 августа 1900 г., на кото-

ром был составлен «план по выходу из кризиса», состоящий из 23 проблем, среди которых одна проблема – проблема №10 звучала так:



«10. Решение проблемы разрешимости для произвольного диофантова уравнения.

Пусть дано произвольное диофантово уравнение с произвольным числом неизвестных и с целыми рациональными коэффициентами; требуется указать общий метод, следуя которому можно было бы в конечное число шагов узнать, имеет данное уравнение решение в целых рациональных числах или нет».

Уточним, решить диофантово уравнение – значит найти целые (рациональные) корни многочлена от *n* переменных с целыми (рациональными) коэффициентами.

Возможно Д. Гильберт и предполагал существование такого метода, но именно попытки решения этой задачи, то есть возможности построения общего метода или доказательства его отсутствия и можно считать началом зарождения теории алгоритмов.

Очевидно, что решение этой проблемы Гильберта начались с поиска того самого метода, но к 30 годам XX века, количество исследований перешло на качественно новый уровень, что в первую очередь произошло и благодаря созданию первых компьютеров, и, соответственно, первых программ. И теперь объектом исследования стал не поиск того или иного общего метода или алгоритма, а сам алгоритм как таковой. Оказывается, дать строгое определение алгоритма порой намного сложней, чем его построить.

Удивительно то, что попытки дать строгое определение понятию «алгоритм» были сделаны в разное время, разными способами, разными учеными, но в конечном итоге все они описали один и тот же класс функций. Свести алгоритм к функции можно так: говорят, что функция *f(x)* вычислима, если существует алгоритм, вычисляющий ее значения *f*, по каждому аргументу *x* из области определения. Приведем ниже основные направления по уточнению понятия «алгоритма» и их авторов:

1) Общерекурсивные функции (К. Гёделя, С. Клини, Ю. Эрбран)

2) μ- рекурсивные функции (Л. Гёделя, С. Клини)

3) λ- определение функции (А. Чёрч, С. Клини)

4) Машины Тьюринга-Поста (А. Тьюринг, Е. Пост)

5) Марковские алгоритмы (А. А. Марков)

6) Графические схемы (Р. Петер)

Глава 2. Нормальные алгоритмы Маркова

2.1. Понятие ассоциативного исчисления

Человеческая история идет вверх кругами.
Круги разные – золотые, кровавые,…
Но все они одинаково разделены на 360 градусов.
И вот он нуля – вперед: 10, 20, 30,…, 200, 360 градусов
– и опять ноль, но присмотритесь
– это же совсем другой, новый ноль.
Е.И. Замятин «Мы»

Рассмотрим следующий способ уточнения понятия алгоритма, который был разработан в 1949 году Марковым Андреем Андреевичем (младшим). Пусть нам задан алфавит – совокупность (конечная) попарно-различных символов $A = \{a_1, a_2, a_3, \ldots, a_n, \ldots\}$. Символы a_i, составляющие алфавит, называются буквами.

Определение 2.1.1 Словом в алфавите A называется любая конечная последовательность букв, записанных подряд слева направо.

Примеры:

а) Записи любого натурального числа в десятичном счислении 21, 3, 321, 1101 есть примеры слов в алфавите $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

б) Пусть $A = \{0, 1\}$. Тогда любое двоичное число 000, 001, 0010, 1101, 1111, … задаёт слово в этом алфавите.

в) Если алфавит $A = \{a, b, c\}$, то тогда словами будут следующие наборы: *abcc, cbbabbc, b, bbb,* $\wedge$ (знак $\wedge$ – означает пустое слова, не содержащие ни одной буквы).

Определение 2.1.2 Количество букв в слове называется длиной слова. Длина пустого слова равна нулю.

Определение 2.1.3 Два слова P и Q в одном и том же алфавите A называется равными (обозначается $P = Q$), если они одинаково записаны т.е. на соответствующих позициях стоят одинаковые буквы.

Определение 2.1.4 Композицией двух слов P и Q в одном алфавите A называется новое слово в том же алфавите, которое получается приписываем к слову P справа слова Q и обозначается $P \circ Q$ (или PQ).

Например, если P = 2131 и Q = 965,то QP = 9652131, PQ=2131965.

Очевидно операция композиции слов не коммутативно $(PQ \neq PQ)$, но ассоциативна т. е. если даны три слова P, Q и R в одном алфавите A, то $(P \circ Q) \circ R = P \circ (Q \circ R) = PQR$.

Например, P = 231, Q = 453, R = 19, тогда $(P \circ Q) \circ R$ = (231∘453) ∘ 19 = 231453∘19 = 23145319; $P \circ (Q \circ R)$ = 231∘(453∘19)= 231∘45319 = 23145319.

В силу ассоциативности операции композиции слов вводимое нами счисление называется ассоциативным.

Определение 2.1.5 Пусть нам даны два слова P и Q в алфавите A. Говорят, что слово P входит в слово Q (или слово Q содержит слово P), если слово Q представимо в виде композиции $Q = RPS$, где R или S – быть может пустые слова.

Например, слово 12 входит в слово 241291247, причем 2 раза.

Очевидно, что каждое слово содержит себя и пустое слово Λ входит в любое другое слово.

Если слово P входит в слово Q несколько раз, то тогда говорят о первом, втором и т. д. вхождениях, обозревая слово Q слева направо.

Определение 2.1.6 Говорят, что два слова P и Q в одном и том же алфавите A, соединённые между собой чёрточкой задает *неориентированную* подстановку $P - Q$. Если же слова P и Q соединены стрелочкой $P \to Q$, то подстановка называется *ориентированной*.

Допустим слово R содержит слово P из подстановки $P - Q$.Заменим какое – то вхождение слова P в слово R на слово Q. При этом получится новое

слово R_1. При этом говорят, что слово R_1 получено из слова R с помощью подстановки $P - Q$ за 1 шаг.

В случае, когда подстановка неориентированная, разрешается заменять как вхождение P на Q, так и вхождение Q на P. Если же подстановка $P \to Q$ – ориентированная, то замена разрешается лишь в указанном стрелкой направлении.

Например, применяя подстановку $ab - abb$ к слову $R = babba$ слева направо, получим слово $R_1 = babbba$, а применяя ту же подстановку вправо налево – получим слово $R_2 = baba$.

Может оказаться, что подстановка $P - Q$ не применима к слову R, т. к. R не содержит ни P, ни Q.Например, подстановка $ab - abb$ к слову $S = acbbcaa$ применима.

Если рассматривать не одну, а несколько подстановок в одном алфавите

$$P_1 - Q_1, P_2 - Q_2, P_3 - Q_3, \ldots, P_s - Q_s, \qquad (*)$$

и применять эти подстановки к слову R и к получаемым при этом словам R_1, $R_2, \ldots$ до тех пор, пока это возможно получится некоторое множество слов, которое называется множеством, порожденным из слово R с помощью системы подстановок $(*)$.

Например, если иметь подстановки 1) $aa \to c$, 2) $bb \to a$, 3) $cc \to \Lambda$, то, применяя их к слову $R = abb$, получится следующее множество: $\{abbc, aac, cc, \Lambda\}$.

Определение 2.1.7 Совокупность всех слов в алфавите A вместе с какой-нибудь конечной системой подстановок называется ассоциативным счислением.

Из определения 2.1.7 следует. Что для задания конкретного счисления достаточно указать алфавит и конечную последовательность подстановок в этом алфавите: например, алфавит $B = \{a, b, *, 0\}$ и в нем система из четырёх подстановок 1) $ab \to b*$; 2) $a* \to 0*$; 3) $*0 \to 0$; 4) $0 \to \Lambda$ задаёт ассоциативное счисление.

Определение 2.1.8 Два слова P и Q в некотором ассоциативном счислении называются *смежными*, если одно из них получается из другого за одно применение какой – либо подстановки данного счисления.

Так в предыдущем примере слова $aa*0b*a$ и $aab*a$ смежные, т. к. второе слово получается из первого с помощью подстановки $*0\to 0$. В то же время слова $ab0ba$ и $a*0b$ не являются смежными.

Определение 2.1.9 Два слова P и Q в данном ассоциативном счислении называются *эквивалентными* (обозначается $P\sim Q$),если существует такая конечная цепочка слов $P, P_1, P_2, \ldots, P_s, Q$, которая начинается со слова P, завершается словом Q и которой пары $(P, P_1), (P_1, P_2), (P_2, P_3), \ldots, (P_s, Q)$ смежные.

Нетрудно понять, что:

a) каждое слово эквивалентно самому себе (рефлексивность);

b) если $P\sim R$ и $R\sim Q$, то $P\sim Q$ (транзитивность);

c) если подстановки счисления не ориентированы, то из $P\sim Q$ следует, что $Q\sim P$ (симметричность).

Из перечисленных свойств следует, что в ассоциативных счислениях с системой неориентированных подстановок отношение эквивалентности $\sim$ разбивает множество слов в алфавите счисления на классы эквивалентности: произвольные два слова R_1 и R_2 будут в одном классе тогда, когда $R_1 \sim R_2$. При этом количество различных классов для одних ассоциативных счислений может быть конечным, для других бесконечным.

Пример 2.1.1 Рассмотрим ассоциативное счисление I_1 алфавитом $A_1 = \{a, b, c\}$ и системой неориентированных подстановок

1. $ba\to ab$
2. $ca\to ac$
3. $cb\to bc$
4. $aa\to c$
5. $bb\to a$
6. $cc\to \Lambda$

то, что слово R получается из слова P за одну подстановку с номером i, будим обозначать в виде $P \Rightarrow R$.

Покажем, что слова $ababacb$ и cb эквивалентны: $ab\underline{ab}acb \overset{1}{\Rightarrow} aa\underline{bb}acb \overset{5}{\Rightarrow} \underline{aa}aacb \overset{4}{\Rightarrow} c\underline{aa}cb \overset{4}{\Rightarrow} \underline{cc}cb \overset{6}{\Rightarrow} cb$.

Докажем, что любое слово ω счисления I_1 эквивалентны одному и только одному из следующих восьми слов: Λ, a, b, c, ab, ac, bc, abc.

Действительно, подстановки (1), (2), (3) позволяют данное слово ω представить в виде $\omega_1 = a^k b^m c^n$, где $a^\kappa = \underbrace{aaa....a}_{k}$.

Далее пользуясь подстановкой (5) получим слово ω_2, содержащее не более одной буквы b т. е. $\omega_2 = a^p c^n$ или $\omega_2 = a^p bc^n$. Подстановка (4) позволит избавиться от парных букв a. Наконец подстановка (6) сократит слова на чётное количество буквы c. В результате мы получим слова, которые содержат каждую букву алфавита A_1 не более чем один раз.

Приведенные выше рассуждения можно проиллюстрировать в виде следующего дерева, берущего начало со слова ω:

$$\omega \overset{1}{\Rightarrow} \omega_1 = a^k b^m c^n \overset{5}{\Rightarrow} \ldots \Rightarrow \left[\begin{array}{l} \text{если } m = 2t,\ mo\ a^p c^n \overset{4}{\Rightarrow} \left[\begin{array}{l} \text{если } p = 2s,\ mo\ c^r \overset{6}{\Rightarrow} \left[\begin{array}{l} \text{если } r = 2d,\ mo\ \Lambda. \\ \text{если } r = 2d+1,\ mo\ c. \end{array}\right. \\ \text{если } p = 2s+1,\ mo\ ac^r \overset{6}{\Rightarrow} \left[\begin{array}{l} \text{если } p = 2d,\ mo\ a. \\ \text{если } r = 2d+1,\ mo\ ac. \end{array}\right. \end{array}\right. \\ \text{если } m = 2t+1,\ mo\ a^p bc^n \overset{4}{\Rightarrow} \left[\begin{array}{l} \text{если } p = 2s,\ mo\ bc^r \overset{6}{\Rightarrow} \left[\begin{array}{l} \text{если } r = 2d,\ mo\ b. \\ \text{если } r = 2d+1,\ mo\ ab. \end{array}\right. \\ \text{если } p = 2s+1,\ mo\ abc^r \overset{6}{\Rightarrow} \left[\begin{array}{l} \text{если } r = 2d,\ mo\ ab. \\ \text{если } r = 2d+1,\ mo\ abc. \end{array}\right. \end{array}\right. \end{array}\right.$$

Итак, в примере 2.1.1 множество всех слов в алфавите $A_1=\{a,b,c\}$ разбилось на восемь классов эквивалентности, притом никакие два слова, принадлежащие разным классам, не эквивалентны (попробуйте обосновать методом от противного).

В каждом ассоциативном исчислении возникает *проблема эквивалентности слов:* для любых слов ω_1 и ω_2 ассоциативного исчисления *I* требуется узнать, можно ли одно из данных слов преобразовать в другое, применяя конечное число раз допустимые подстановки, т.е. эквивалентны слова ω_1 и ω_2 или нет.

Решить проблему эквивалентности слов для конкретного исчисления – это значит указать такой метод, с помощью которого для любых двух слов можно установить: эквивалентны они или нет.

Если для некоторого исчисления *I* такой метод найден, то говорят, что проблема эквивалентности слов для этого исчисления *разрешима*. Если же доказано, что общего такого метода не существует, то в этом случае говорят, что для ассоциативного исчисления *I* проблема эквивалентности слов *неразрешима.*

Возможен и третий случай, когда этот метод еще не найден, но и не доказано, что он не существует. В этом случае говорят, что проблема эквивалентности слов для этого ассоциативного исчисления не решена (открыта).

Для ассоциативного исчисления из примера 2.1.1 проблема эквивалентности слов разрешима: два слова ω_1 и ω_2 эквивалентны тогда и только тогда, когда они эквивалентны одному и тому же слову из множества *{Λ,a,b,c,ab,ac,bc,abc}.* Так, например, слова *ababacb* и *cacba* эквивалентны одному и тому же слову *bc*, следовательно, эквивалентны между собой. А слова *cacba* и *caabbc* не эквивалентны т.к. *cacba~bc, caabbc~ac*, но *bc* не эквивалентно *ac*.

Можно сформулировать *обобщенную проблему эквивалентности слов*: указать алгоритм, решающий проблему эквивалентности слов для любого ассоциативного счисления.

Разрешимость столь широкой проблемы с самого начала вызвала большие сомнения. Впоследствии эти сомнения подтвердились: было доказано, что обобщенная проблема не разрешима. Более того, в 1946-1948 гг. А. А. Марковым и Э. Постом были построены примеры ассоциативных счис-

лений с неразрешимой проблемой эквивалентности слов. Такие доказательства стали возможны благодаря появлению математического определения понятия алгоритма.

Для тех ассоциативных счислений, где проблема эквивалентности неразрешима, уместно ставить *ограниченную проблему* эквивалентности слов: Для любых двух слов ω_1 и ω_2 данного ассоциативного счисления *I* требуется узнать, можно ли преобразовать одно из них в другое, применяя не более *k* раз допустимых подстановок (*k* – фиксированное число). Другими словами существует ли цепочка слов, соединяющая слова ω_1 и ω_2 , длина которой не превосходит *k*.

Разумеется, ограниченная проблема эквивалентности для любого ассоциативного счисления разрешима: для этого достаточно перебрать все цепочки преобразований слова ω_1 , длина которых не превосходит *k*. Таких цепочек, разумеется, конечное число. Если хотя бы одна из цепочек заканчивается словом ω_2 , то ω_1 и ω_2 . Иначе нет.

Любое ассоциативное счисление можно вообразить в виде бесконечного алгоритма, где каждое слово означает площадку, а смежные слова соединены коридором. В таком случае ω_1 и ω_2 означает, что площадка ω_2 достижима с площадки ω_1. В случае ограниченной проблемы эквивалентности площадка ω_2 должна быть достижима с площадки ω_1 не более чем за *k* переходов.

2.2. Нормальные алгоритмы Маркова

Определение 2.2.1 Алгоритмом по Маркову (нормальным алгоритмом) называется ассоциативное исчисление с системой ориентированных подстановок, в котором строго оговорено в какой последовательности следует применять эти подстановки.

А именно:

а) система допустимых подстановок задается в определенном порядке (номеруется):

$$1)\ P_1 \to Q_1;\ \ 2)\ P_2 \to Q_2; \ldots;\ \ n)\ P_n \to Q_n;$$

б) алгоритм предписывает, отправляясь от заданного слова R, просмотреть ориентированные подстановки в порядке возрастания их номеров, разыскивая подстановку с наименьшим номером i $(P_i \to Q_i)$, левая часть которой P_i входит в слово R. Если такой подстановки не найдется, то процесс прекращается. В противном случае берется самое левое вхождение P_i в R и оно заменяется Q_i. При этом слово R преобразуется в слово R_1. Далее, вместо слова R берется слово R_1, и процесс начинается сначала. Так поступают до тех пор, пока процесс не прекратится. Это может произойти в двух случаях:

1) К полученному слову R_k не применима ни одна из данных подстановок;

2) Когда при получении слова R_k применена подстановка с меткой, т.е. заключительная подстановка (ее будем метить точкой справа).

При этом считается, что данный алгоритм слово R переработал в слово R_k.

Если процесс переработки слова R ни при каком шаге не обрывается, то говорят, что алгоритм к слову R не применим (результат не определен).

Итак, любое ассоциативное исчисление с упорядоченной системой ориентированных подстановок, которое снабжено описанной выше «инструкций» относительно очередности применения подстановок, задает нормальный алгоритм.

Пример 2.2.1 Пусть в алфавите $A_2=\{a,b,c\}$ задана система ориентированных подстановок:

1. $cb \to cc$
2. $cca \to ab$
3. $ab \to bca$
4. $ca \to \Lambda$

Этот алгоритм, будучи применен к слову cabc, никогда не обрывается:

$c\underline{ab}c \overset{3}{\Rightarrow} \underline{cb}cac \overset{1}{\Rightarrow} c\underline{ccab} \overset{2}{\Rightarrow} cabc$ – получилось первоначальное слово, т.е. алгоритм закончился.

Если же брать слово *baaacca*, то после нескольких шагов процесс оборвется на слове *bb*:

$$baaa\underline{cca} \overset{2}{\Rightarrow} baa\underline{ab}ca \overset{3}{\Rightarrow} ba\underline{ab}caca \overset{3}{\Rightarrow} b\underline{ab}cacaca \overset{3}{\Rightarrow} bb\underline{ca}cacaca \overset{4}{\Rightarrow} ... \Rightarrow bb.$$

Два нормальных алгоритма отличаются друг от друга алфавитом и системой ориентированных подстановок или даже только порядком подстановок.

Пример 2.2.2 Пусть в одном и том же алфавите заданы два нормальных алгоритма I_1 и I_2, отличающиеся друг от друга только порядком подстановок:

I_1:	I_2:
1. $ab \to bac$	1. $bc \to bb$
2. $ac \to ca$	2. $ab \to bac$
3. $aa \to \Lambda$	3. $ac \to ca$
4. $bc \to bb$	4. $aa \to \Lambda$
5. $bb \to \Lambda$	5. $bb \to \Lambda$

Покажем, что $I_1(abbc) \neq I_2(abbc)$:

$$I_1(\underline{ab}bc) \overset{1}{\Rightarrow} b\underline{ac}bc \overset{2}{\Rightarrow} bc\underline{ab}c \overset{1}{\Rightarrow} bcb\underline{ac}c \overset{2}{\Rightarrow} bcbc\underline{ac} \overset{2}{\Rightarrow} \underline{bc}bcca \overset{4}{\Rightarrow} bb\underline{bc}ca \overset{4}{\Rightarrow} bbb\underline{bc}a \overset{4}{\Rightarrow}$$

$$\Rightarrow \underline{bbbb}ba \overset{5}{\Rightarrow} ... \overset{5}{\Rightarrow} ba.$$

$$I_2(ab\underline{bc}) \overset{1}{\Rightarrow} \underline{ab}bb \overset{2}{\Rightarrow} b\underline{ac}bb \overset{3}{\Rightarrow} \underline{bc}abb \overset{1}{\Rightarrow} bb\underline{ab}b \overset{2}{\Rightarrow} bbb\underline{ac}b \overset{3}{\Rightarrow} bb\underline{bc}ab \overset{1}{\Rightarrow} bbbb\underline{ab} \overset{2}{\Rightarrow}$$

$$\Rightarrow bbbbb\underline{ac} \overset{3}{\Rightarrow} bbbbbca \overset{1}{\Rightarrow} \underline{bb}\ \underline{bb}\ \underline{bb}a \Rightarrow ... \overset{5}{\Rightarrow} a.$$

Этот пример показывает, что в нормальных алгоритмах имеет значение не только сами подстановки, но и их порядок.

2.3. Вычисление арифметических функций с помощью нормальных алгоритмов

Определение 2.3.1 Функция $y=F(x_1,x_2,\ldots,x_n)$ называется арифметической функцией, если аргументы x_i и значение y принимают только натуральные значения из $N^*=\{0, 1, 2, 3,\ldots\}$.

Положительные натуральные числа зададим как слова в однобуквенном алфавите $A=\{1\}$: пусть число n задается в виде слова из n «палочек».

Построим алгоритм, который любое слово вида $\underbrace{111...1}_{m\ раз}*\underbrace{111...1}_{n\ раз}$ перерабатывает в слово вида $\underbrace{111...111}_{m\cdot n\ раз}$, т.е. вычисляет произведение двух чисел.

Пример 2.3.1 В алфавите $A_3=\{1, *, 0, v\}$ нормальный алгоритм I_3 задается следующей упорядоченной системой ориентированных подстановок:

1. $*11\to 0*1$
2. $*1\to\Lambda$
3. $10\to 01v$
4. $v0\to 0v$
5. $0\to\Lambda$
6. $v\to 1$

Пусть $m = 3$, $n = 4$, т.е. зададим I_3 слово

$$111*1111\overset{1}{\Rightarrow}1110*111\overset{1}{\Rightarrow}11100*11\overset{1}{\Rightarrow}111000*1\overset{2}{\Rightarrow}11\underline{10}00\overset{3}{\Rightarrow}1\underline{10}1v00\overset{3}{\Rightarrow}\underline{10}1v1v00\overset{3}{\Rightarrow}$$

$$\overset{3}{\Rightarrow}01v1v1\underline{v0}0\overset{4}{\Rightarrow}01v1v\underline{10}v0\overset{3}{\Rightarrow}01v1\underline{v0}1vv0\overset{4}{\Rightarrow}01v\underline{10}v1vv0\overset{3}{\Rightarrow}01\underline{v0}1vv1vv0\overset{4}{\Rightarrow}0\underline{10}v1vv1vv0\overset{3}{\Rightarrow}$$

$$\overset{3}{\Rightarrow}001vv1vv1vv\underline{0}\overset{4}{\Rightarrow}001vv1vv1\underline{v0}v\overset{4}{\Rightarrow}001vv1vv\underline{10}vv\overset{3}{\Rightarrow}001vv1vv\underline{01}vvv\overset{4}{\Rightarrow}001vv1\underline{v0}v1vvv\overset{4}{\Rightarrow}$$

$$\overset{4}{\Rightarrow}001vv\underline{10}vv1vvv\overset{3}{\Rightarrow}001vv\underline{01}vvv1vvv\overset{4}{\Rightarrow}\overset{3}{\ldots}\Rightarrow 0001vvv1vvv1vvv\overset{5}{\Rightarrow}\ldots\Rightarrow$$

$$\Rightarrow 1vvv1vvv1vvv\overset{6}{\Rightarrow}\ldots\Rightarrow 111111111111.$$

Обобщенно можно сказать, что подстановка (1) и (2), примененная к слову $\underbrace{111...1}_{m}*\underbrace{111...1}_{n}$, перерабатывает n единиц, расположенных справа от $*$, в $(n - 1)$ нулей. При этом m не меняется, а $*$ исчезает. Далее, каждая перестановка пары 10 с помощью подстановки (3) порождает символ v. При этом 1

не исчезает. Поэтому последовательное применение подстановок (3) и (4), «перетаскивает» все нули на начало слова и каждое такое перемещение нуля через единицы порождает *m* символов *v* в итоге подстановки (1), (2), (3),(4) приводят нас к слову вида:

$$\underbrace{00...0}_{n-1}\ \underbrace{1\underbrace{vv...v}_{n-1}1\underbrace{vv...v}_{n-1}...1\underbrace{vv...v}_{n-1}}_{m\ групп}$$

А теперь, применяя подстановку (5), избавимся от нулей и в оставшемся слове, будет содержаться m∗n символов. Применяя подстановку (6) получим *(m∗n)* расположенных подряд символов и на этом процесс закончится.

Пример 2.3.2 Алгоритм I_4 для вычисления функции $y=\left[\frac{x}{2}\right]$.

1. $a11\to aa$

2. $a1\to a$

3. $0a\to 10$

4. $0\to\Lambda$. *(подстановка с меткой)*

5. $11\to 0a$

$$\omega=\overbrace{111...1}^{x\ раз}\overset{5}{\Rightarrow}0a\overbrace{111...1}^{x-2}\overset{1}{\Rightarrow}0aa\overbrace{111...1}^{x-4}\Rightarrow...\Rightarrow\begin{cases}0\underbrace{aa...a}_{t},\ если\ x=2t;\\ 0aa...a1\overset{2}{\Rightarrow}0\overbrace{aa...a}^{t}\overset{3}{\Rightarrow}0aa...a\overset{3}{\Rightarrow}10\overbrace{aa...a}^{t-1}\overset{3}{\Rightarrow}...\\ \Rightarrow\overbrace{111..1}^{t}0\overset{4}{\Rightarrow}111..1,\ если\ x=2t+1.\end{cases}$$

Так как (4) – заключительная подстановка, то процесс завершается, и мы получаем $\left[\frac{x}{2}\right]$.

Примеры 2.3.1, 2.3.2 показывают, как можно приспособить нормальные алгоритмы Маркова для вычисления арифметических функций. Можно пока-

зать, что все известные из арифметики и теории чисел функции вычислимы с помощью нормальных алгоритмов. Это обстоятельство позволило А. Маркову предложить в качестве определения понятия алгоритма нормальный алгоритм, и это, в силу теоремы Детловса имеет достаточное основание.

Пример 2.3.3 Нормальный алгоритм Маркова в алфавите $A=\{1,0,*\}$ задан следующей системой ориентированных подстановок:

1) $11*\to*$ 4) $*1\to1$

2) $*11\to*$ 5) $1*\to1$

3) $1*1\to1$ 6) $*\to0$

В какое слово перерабатывает этот алгоритм слово вида 111…1*111…1, то есть, какую арифметическую функцию *f(x,y)* он вычисляет?

Решение:

$$\overbrace{111...11}^{x}*\overbrace{111...1}^{y}\overset{1}{\Rightarrow}\overbrace{11...11}^{x-2}*\overbrace{111...1}^{y}\Rightarrow...\Rightarrow\begin{cases}\text{если } x=2t,\ mo\ *\overbrace{111...1}^{y}\overset{2}{\Rightarrow}*\overbrace{111...1}^{y-2}\overset{2}{\Rightarrow}...\Rightarrow(1)\\ \text{если } x=2t+1,\ mo\ 1*\overbrace{111...1}^{y}\overset{2}{\Rightarrow}1*\overbrace{111...1}^{y-2}\overset{2}{\Rightarrow}...\Rightarrow(2)\end{cases}$$

$$(1)\Rightarrow\begin{cases}\text{если } y=2k,\ mo\ *\overset{6}{\Rightarrow}0\\ \text{если } y=2k+1,\ mo*1\overset{4}{\Rightarrow}1\end{cases}\qquad(2)\Rightarrow\begin{cases}\text{если } y=2k,\ mo1*\overset{4}{\Rightarrow}1\\ \text{если } y=2k+1,\ mo1*1\overset{3}{\Rightarrow}1\end{cases}$$

алгоритм завершился. Данный алгоритм вычисляет двухместный предикат

$$P(x,y)=\begin{cases}0, \text{если } x \text{ и } y \text{ четные}\\ 1 \text{ в остальных случаях}\end{cases}.$$

2.4. Марков Андрей Андреевич

Андрей Андреевич Марков (22.09.1903 - 11.10.1979 гг.) – выдающийся советский математик и логик, член-корреспондент АН СССР, заведующий кафедрой математической логики Московского государственного университета.

Андрей Андреевич Марков родился в 1903 году в Санкт-Петербурге в семье знаменитого русского ученого

А. А. Маркова. После окончания Восьмой Петроградской Гимназии в 1919 году, Андрей Марков поступил в Ленинградский государственный университет. Далее Марков продолжил образование в аспирантуре в Астрономическом Институте, окончил которую в 1928 году. В 1935 году Маркову присвоена ученая степень физико-математических наук, а с 1953 года Марков был членом-корреспондентом Академии Наук Союза Советских Социалистических Республик. Более двадцати лет, начиная с 1933 года, работал профессором в Ленинградском университете, а также в Математическом институте имени Стеклова с 1939 года по 1972 год. Кроме того, был заведующим кафедрой математической логики МГУ имени Ломоносова.

Научная деятельность А. А. Маркова чрезвычайно плодотворна и многогранна. Она распространяется на многие области математики и смежных с ней наук. Почти в каждой из тех областей науки, которые привлекли внимание А. А. Маркова, с его именем связаны научные достижения первостепенного значения.

Основные циклы опубликованных работ А. А. Маркова относятся к следующим наукам разделам: (циклы перечисляются в основном в хронологическом порядке): теоретическая физика, небесная механика, общая теория динамических систем и смежные с ней вопросы теории меры и топологии, комбинаторная топология (теория кос и зацеплений), теория полиномов, теория топологических групп, алгоритмические проблемы алгебры, общая теория алгоритмов, конструктивная математическая логика и основания математики, конструктивный математический анализ, алгоритмические проблемы топологии, теория булевых функций (некоторые проблемы минимизации), теория вычислимых инвариантов бинарных отношений. Кроме циклов исследований, относящихся к перечисленным областям, А. А. Марковым опубликованы работы по геометрии, теории пластичности, аксиоматической теории множеств, истории науки и др.

Формирование А. А. Маркова как математика происходило на фоне триумфальных успехов созданной Г. Кантором и Р. Дедекиндом теории мно-

жеств. Эти успехи привели к формированию и чрезвычайно широкому распространению теоретико-множественного способа мышления, основанного на специфических актах воображения, составляющих в целом абстракцию актуальной бесконечности. Небольшое замешательство, вызванное выявлением некоторых антиномий в так называемой «наивной» теории множеств, было быстро преодолено посредством уточнения «правил игры». В результате интенсивной работы многих исследователей фундамент математики был оформлен в виде четких аксиоматических систем теории множеств. Это предоставило возможность строить важнейшие конкретные разделы математики (алгебру, топологию, математический анализ, теорию вероятностей и др.) чисто дедуктивным путем и достичь высокого, до этого в большинстве областей математики немыслимого уровня отчетливости архитектуры математических теорий, достичь нового уровня «строгости», столь высоко ценимого математиками. Все это способствовало росту престижа теории множеств и появлению у большинства математиков чувства удовлетворенности создавшимся положением, передаваемого каждому новому контингенту молодежи, готовящей себя к профессии математика.

Этот общий фон математической жизни первой половины текущего столетия длительное время определял научные интересы А. А. Маркова и формировал направления его творческой деятельности. Переход А. А. Маркова к проблемам конструктивного направления в математике, происшедший, как уже упоминалось, в конце 40-х годов, связан с некоторыми давно назревавшими глубинными процессами в области исследований оснований математики. Еще в начале XX столетия были высказаны (в развернутом виде – Л. Э. Я. Брауэром и Г. Вейлем) сомнения в удовлетворительности теории множеств в качестве логической основы математики. В критическом анализе, с которым выступили Брауэр и Вейль, речь шла не о каких-либо дефектах теории множеств, связанных с ее формальным развертыванием по заданным «правилам игры», а совсем о другой стороне дела. Брауэр и Вейль формулировали свои сомнения, апеллируя к требованию интуитивной ясности. По их

мнению, представления об «актуально бесконечных множествах» и некоторые связанные с этими представлениями логические средства не соответствуют математической интуиции. Конечно, если бы все дело сводилось к особенностям человеческой интуиции, то критика Брауэра и Вейля не имела бы тех последствий, которые она фактически вызвала. В действительности же Брауэр и Вейль в субъективистских терминах поставили чрезвычайно сложную и принципиальную методологическую проблему, основное содержание которой Д. Гильберт сформулировал следующим образом: «Раньше мы уже выяснили, что какие бы опыты и наблюдения и какую бы отрасль науки мы ни рассматривали, нигде в действительности мы не находим бесконечности. Должны ли мысли о вещах быть столь непохожими на то, что происходит с вещами, должны ли они сами по себе идти другим путем, совершенно в стороне от действительности?» [Д. Гильберт «О бесконечном», 1925 г.].

В 30-х годах текущего столетия в математике произошло событие большого принципиального значения: было выработано несколько эквивалентных друг другу уточнений общего понятия алгоритма и сложилось убеждение, что проблема выработки искомого точного понятия получила окончательное (в принципиальном отношении) решение. Этот момент оказался переломным в судьбе многих идей Брауэра и Вейля. Постепенно, на первых порах в эклектической смеси с представлениями и методами классической теории множеств, но все более и более освобождаясь от них, усилиями многих исследователей начали вырисовываться контуры новых принципов и методов построения математических теорий, начало складываться на современной основе конструктивное направление в математике.

А. А. Марков на протяжении многих лет своей научной деятельности применял классическую теорию множеств в качестве основы многих своих исследований. Однако с первых шагов становления общей теории алгоритмов его отношение к классической теории множеств стало меняться. В теории алгоритмов он увидел не только мощные технические, но и богатые обще логические возможности и начал изучать их с пристальным вниманием.

Работа С. К. Клини «Об истолковании интуиционистской теории чисел» (1945г.), уточнившая намеченный Л. Э. Я. Брауэром и развитый А. Н. Колмогоровым подход к конструктивному пониманию суждений, внесла значительную ясность в принципиальные основы нового (конструктивного) направления в математике.

А.А. Марковым разработано понятие нормального алгоритма (в авторской транслитерации – «алгорифм»), оказавшееся очень удобным для многих целей. Построен развернутый аппарат теории нормальных алгоритмов; в частности, обстоятельно разработан аппарат построения по заданным алгоритмам новых алгоритмов, обладающих заданными свойствами (см. [48], гл. I-IV). Установлена неразрешимость некоторых массовых проблем теории нормальных алгоритмов, играющих роль исходных массовых проблем при доказательствах неразрешимости разнообразных массовых проблем алгебры (см. [48], гл. V). Разработан существенный для построения конкретных теорий конструктивной математики (в частности, конструктивной топологии) специальный аппарат теории алгоритмов, связанный с оперированием над системами слов (см. [37]).

Также существенно дополнена имеющая принципиальное значение теорема С. К. Клини о представлении частично рекурсивных функций через примитивно рекурсивные, а именно, дана исчерпывающая характеризация примитивно рекурсивных функций, допустимых в качестве внешней функции в формуле С. К. Клини, и как следствие из этой характеризации установлена возможность использования в роли внешней функции весьма простых примитивно рекурсивных функций (см. [39]).

Получены очень интересные результаты о сложности нормальных алгоритмов, вычисляющих булевы функции (эти результаты были доложены весной 1963 г. в семинаре по математической логике Московского университета). Принципиальное значение этой работы состоит в открытии совершенно нового метода выявления неразрешимых массовых проблем, позволяюще-

го также усиливать в интересном направлении результаты об уже известных неразрешимых массовых проблемах.

Создана капитальная монография "Теория алгорифмов" [48], суммирующая основные результаты А. А. Маркова по теории нормальных алгоритмов и упомянутые выше результаты по алгоритмическим проблемам алгебры. Ее фактическая роль в современной математике далеко выходит за рамки суммарного изложения научных достижений автора. Ввиду того, что изложение носит систематический и целостный характер, монография в целом дает последовательное и чрезвычайно содержательное построение теории алгоритмов и аппарата теории ассоциативных исчислений, а также систематическое изложение разработанного автором тонкого аппарата сведения одних массовых проблем алгебры к другим. При этом теория ассоциативных исчислений строится (по-видимому, впервые в литературе) как теория конструктивных объектов, независимо от теоретико-множественного понятия ассоциативной системы. Тем самым здесь заложены основы методики построения алгебры на базе строго конструктивных понятий.

Эта монография уже послужила и продолжает служить источником новых идей и новых постановок проблем, она вызвала к жизни много новых работ различных авторов и новые направления исследований. В настоящее время «Теория алгорифмов» А. А. Маркова стала настольной книгой как у специалистов, так и у научной молодежи.

В области исследования алгоритмических проблем алгебры особенно значительные и особенно принципиальные результаты получены А. А. Марковым в 1962 г. Они связаны с введенным А. А. Марковым понятием вычислимого инварианта. Роль инвариантов в математике очень велика. Этим определяется важность в теориях конструктивного характера тех проблем, которые касаются свойств вычислимых инвариантов.

Пусть R есть рефлексивное, симметричное и транзитивное бинарное отношение, определенное для конструктивных объектов какого-либо типа. *Вычислимым инвариантом отношения R для объекта P* Марков А.А. на-

зывает всякий алгоритм, применимый к каждому объекту рассматриваемого типа и перерабатывающий в один и тот же объект все объекты, связанные с P отношением R. Объекты P и Q А. А. Марков называет *неотличимыми по инвариантам отношения* R, если $I(P) = I(Q)$ всякий раз, когда I есть вычислимый инвариант отношения R как для P, так и для Q. В работе [33] (более подробное изложение – в [38]) устанавливаются следующие основные результаты об ассоциативных исчислениях и групповых исчислениях (последние в цитируемых работах называются инверсивными исчислениями; групповые (инверсивные) исчисления в литературе также часто называются конечно-определенными группами.)

Теорема 1. Могут быть построены инверсивное исчисление G и слово A в его алфавите такие, что, во-первых, A не эквивалентно в G пустому слову и, во-вторых, A и пустое слово неотличимы по инвариантам эквивалентности слов в G.

Теорема 2. Каково бы ни было ассоциативное исчисление A, можно построить ассоциативное исчисление B в четырехбуквенном алфавите такое, что, во-первых, A включаемо в B и, во-вторых, B неотличимо по инвариантам взаимной преобразуемости от тривиального исчисления, построенного в том же алфавите, что и B.

Следствие теоремы 2. Каковы бы ни были ассоциативные исчисления A и C, можно построить ассоциативные исчисления D и F в одном и том же алфавите такие, что, во-первых, A включаемо в D, во-вторых, F изоморфно C и, в-третьих, D и F неотличимы по инвариантам взаимной преобразуемости. При этом, если алфавит исчисления C содержит q букв, то D и F можно построить как ассоциативные исчисления в $(q+4)$-буквенном алфавите.

Теорема 3. Каково бы ни было инверсивное исчисление G, можно построить инверсивное исчисление H с восьмибуквенным положительным алфавитом такое, что, во-первых, G включаемо в H и, во-вторых, H неотличимо по инвариантам взаимной приводимости от явно единичного исчисления с таким же положительным алфавитом, как у H.

Следствие теоремы 3. Каковы бы ни были инверсивные исчисления G и K, можно построить инверсивные исчисления M и N в одном и том же алфавите такие, что, во-первых, G включаемо в M, во-вторых, N изоморфно K и, в-третьих, M и N неотличимы по инвариантам взаимной переводимости. При этом, если положительный алфавит исчисления K содержит q букв, то M и N можно построить как инверсивные исчисления с $(q+8)$-буквенным положительным алфавитом.

Эти теоремы представляют собой первые в математической литературе результаты о существовании (точнее, о возможности построения) конкретных конструктивных объектов, не связанных заданным бинарным отношением и в то же время неотличимых по инвариантам рассматриваемого отношения. Таким образом, эти теоремы кладут начало существенно новому и чрезвычайно важному направлению исследований. При этом перечисленные теоремы делают весьма эффективный почин в новом направлении, они дают ответы на наиболее важные из возникающих в теории ассоциативных и инверсивных (т. е. групповых) исчислений вопросов указанного характера. Кроме того, перечисленные теоремы обладают большой силой обобщения. Дело в том, что из неотличимости по инвариантам отношения R некоторых конкретных объектов P и Q, не связанных этим отношением, следует невозможность алгоритма, распознающего для любого конструктивного объекта S рассматриваемого типа, находится ли он в отношении R к P. Поэтому теорема 1 представляет собой усиление известной теоремы П. С. Новикова о существовании конечно-определенной группы с неразрешимой проблемой эквивалентности слов, из следствия теоремы 2 сразу вытекает упомянутая выше общая теорема из монографии А. А. Маркова [48] о нераспознаваемости свойств ассоциативных исчислений, а следствие теоремы 3 является усилением упомянутого выше результата С. И. Адяна и М. О. Рабина. [Ю. В. Линника, Н. А. Шанина «Андрей Андреевич Марков. К пятидесятилетию со дня рождения» // Успехи математических наук Т. IX, вып. 1(59), (1954), стр. 145-149].

К столетнему юбилею А.А. Маркова был издан сборник его трудов, изучение которого полезно и увлекательно: Марков А. А. Избранные труды / Сост. и общ. ред. Н. М. Нагорного. — М.: Изд-во МЦНМО, 2002. — Т. 1. — 533 с. — (Математика. Механика. Физика). Т. 2. — 648 с. — (Теория алгорифмов и конструктивная математика; Математическая логика; Информатика и смежные вопросы). Также необходимо сказать, что нормальные алгоритмы созданные Марковым являются основой для современного языка программирования – Рефала.

Контрольная работа №2

1. Нормальный алгоритм Маркова в алфавите $A = \{1,0,*\}$ задан следующей системой ориентированных подстановок:

1) $1*1 \to 0*$
2) $10 \to 01$
3) $1* \to *$
4) $*1 \to *$
5) $* \to \Lambda$
6) $0 \to 1.$

В какое слово перерабатывает этот алгоритм слово вида 111…1 $*$ 111…1, то есть, какую арифметическую функцию $f(x, y)$ он вычисляет.

2. Нормальный алгоритм Маркова в алфавите $A = \{ a, в, c, \}$ определён следующей системой ориентированных подстановок:

1) $в \to acc$
2) $ca \to acc$
3) $aa \to \Lambda$
4) $ccc \to \Lambda.$

Докажите, что любое слово $\omega \in A$ перерабатывается этим алгоритмом в одно из следующих слов: $\{ \Lambda, c, cc, a, ac, acc \}$.

3.Задайте нормальный алгоритм Маркова, который любое слово вида 111…1 $*$ 111…1 перерабатывает в 1, если $|x - y|$ четное число и в 0, если $|x - y|$ нечетное число. Проверьте правильность вашего алгоритма, построив дерево доказательства.

4.Задайте нормальный алгоритм Маркова, для вычисления функции *min (x, y), m, n* $\in$ *N*. Обоснуйте действие вашего алгоритма, построив дерево доказательств.

5.Нормальный алгоритм Маркова в алфавите *A = {а, в, с}* задан следующей системой ориентированных подстановок:

1) $св \to сс$
2) $сса \to ав$
3) $ав \to вса$.

Поясните, что значит, данный алгоритм не применим к слову ω в алфавите *A*. Придумайте слово в алфавите *A,* над которым данный алгоритм работает бесконечно долго.

6. Нормальный алгоритм Маркова в алфавите *A =*{1,*, *a, b}* задан следующей системой ориентированных подстановок:

1) $*11 \to a*1$
2) $*1 \to a$
3) $1a \to a1b$
4) $ba \to ab$
5) $b1 \to 1b$
6) $a1 \to a$
7) $ab \to b$
8) $b \to 1$
9) $1 \to 1$.

Выполняя пошаговое исполнение алгоритма над словом 1111…111 выясните, какую функцию она вычисляет.

7. В одном и том же алфавите *A = {a, в, с}* заданы два алгоритма Маркова ω_1 и ω_2, которые отличаются друг от друга лишь порядком подстановок:

ω_1:

1) $ab \to bac$
2) $ac \to ca$
3) $aa \to \Lambda$
4) $bc \to bb$
5) $b\,b \to \Lambda$;

ω_2:

1) $bc \to bb$
2) $ab \to bac$
3) $ac \to ca$

4) $aa \to \Lambda$ 5) $bb \to \Lambda$.

Докажите, что алгоритмы не равносильны, в частности $\omega_1(abbc) \neq \omega_2(abbc)$.

8. Нормальный алгоритм Маркова в алфавите $A = \{1, *, 0\}$ задан следующей системой ориентированных подстановок:

1) $1*1 \to *$ 4) $*11 \to *1$

2) $11 \to 1*$ 5) $*1 \to 0$

3) $1* \to 1$ 6) $* \to 0$.

В какое слово перерабатывает этот алгоритм слово вида $\underbrace{111...11}_{x} *$ $\underbrace{111...11}_{y}$, то есть, какой предикат определен этим алгоритмом?

9. Задайте нормальный алгоритм Маркова для вычисления функции

$$dw(x, y) = \begin{cases} 1, \ x \vdots y; \\ 0, \ \textit{в остальных случаях}. \end{cases}$$

Проверьте правильность Вашего алгоритма, построив дерево доказательств.

10.

1) Задайте нормальный алгоритм Маркова для вычисления предиката $x \vdots 3 = \begin{cases} 1, \textit{если } x = 3, \\ 0, \textit{в остальных случаях}. \end{cases}$

2) Пошаговым исполнением проверьте действительность алгоритма.

11. Нормальный алгоритм Маркова в алфавите $A = \{a, в, c\}$ задан следующей системой ориентированных подстановок:

1) $b \to acc$ 4) $ccc \to \Lambda$.

2) $ca \to acc$

3) $aa \to \Lambda$

Докажите, что любое слово $\omega \in A$ перерабатывается этим алгоритмом в одно из следующих слов $\{\Lambda, a, c, ac, acc, cc\}$.

12.Нормальный алгоритм Маркова задан следующей системой ориентированных подстановок:

1) $1*1 \to 0*$

2) $10 \to 01$

3) $* \to \Lambda$

4) $0 \to 1$

В какое слово перерабатывает этот алгоритм слово вида $\underbrace{111...11}_{x} * \underbrace{111...11}_{y}$, то есть, какую арифметическую функцию $f(x, y)$ оно вычисляет?

13. Задайте нормальный алгоритм Маркова для вычисления функции

$$F(x, y) = \begin{cases} |x - y|, \text{ если } x - \text{чётное число и } y - \text{нечётное число}, \\ 0, \text{ в остальных случаях}. \end{cases}$$

Обоснуйте свой алгоритм в виде дерева дедуктивного вывода.

14. Нормальный алгоритм Маркова в алфавите $A = \{1,0\}$ задан следующей системой ориентированных подстановок:

1) $11 \to 0$

2) $01 \to 1$

3) $00 \to 0$.

В какое слово перерабатывает этот алгоритм слово вида 111…11 , то есть, какой предикат определен этим алгоритмом?

15. Задайте нормальный алгоритм Маркова для вычисления функции

$|x - y|$ $(x, y \in N)$. Обоснуйте действие вашего алгоритма, построив дерево доказательства.

16. Нормальный алгоритм Маркова в алфавите $A = \{a, в, c\}$ задан следующей системой ориентированных подстановок: $b \to ac$

1) $cc \to ac$

2) $aa \to \Lambda$

3) $ca \to \Lambda$

Доказать, что любое слово $\omega \in A$ перерабатывается этим алгоритмом в одно из следующих слов: $\{\Lambda, a, c, ac\}$.

17. Композиция двух алгоритмов Маркова: по завершении работы алгоритма 1 начинает работу алгоритма 2

алг1: $1)11 \to a$ $2)a1 \to a$ $3)1 \to 0$

алг2: $1)a \to 1$

1) Выяснить какую функцию вычисляет композиция алгоритмов 1 и 2?

2) Построить алгоритм для вычисления функции $\left[\frac{x}{3}\right]$.

18. Задайте нормальный алгоритм Маркова для вычисления функции, $F(x,y) = \begin{cases} |x-y|, \textit{ если } x, y - \textit{четные, натуральные числа}, \\ 0 \textit{ в остальных случаях.} \end{cases}$.

Обоснуйте свой алгоритм в виде дерева дедуктивного вывода.

19. Задайте нормальный алгоритм Маркова для вычисления функции *max(m,n), n,m∈N*. Обоснуйте свой алгоритм в виде дерева доказательства.

20. Задайте нормальный алгоритм Маркова, который любое слово вида $\underbrace{111...1}_{x \textit{ раз}} * \underbrace{111...1}_{y \textit{ раз}}$ перерабатывает в *1*, если разность |*x*-*y*| есть число четное и в *0*, если |*x*-*y*| есть число нечетное.

21. Задайте нормальный алгоритм Маркова для вычисления функции, $F(x,y) = \begin{cases} x \div y, \textit{ если } x, y - \textit{четные, натуральные числа}, \\ 0 \textit{ в остальных случаях.} \end{cases}$ Обоснуйте свой алгоритм в виде дерева дедуктивного вывода.

22. Задайте нормальный алгоритм Маркова для вычисления функции, $F(x,y)=\begin{cases} x+y, \text{если } x, y - \text{четные, натуральные числа}, \\ 0 \text{ в остальных случаях.} \end{cases}$ Обоснуйте свой алгоритм в виде дерева дедуктивного вывода.

23. Нормальный алгоритм Маркова в алфавите $A=\{*,0,1\}$ задан следующей системой ориентированных подстановок:

1. $1*1\to *$
2. $11*\to 1*$
3. $*11\to *1$
4. $*1\to 1$
5. $1*\to 1$
6. $*\to 0$

В какое слово перерабатывает этот алгоритм слово, вида $\underbrace{111...1}_{x \text{ раз}}*\underbrace{111...1}_{y \text{ раз}}$? Т.е. какой предикат задан этим алгоритмом?

24. Нормальный алгоритм Маркова в алфавите $A=\{a, b, c\}$ задан следующей системой ориентированных подстановок:

1. $b\to acc$
2. $ca\to acc$
3. $aa\to \Lambda$
4. $cccc\to \Lambda$

Докажите, что любое слово этого исчисления перерабатывается к одному из следующих слов: $\Lambda, c, cc, ccc, a, ac, acc, accc$.

25. Нормальный алгоритм Маркова в алфавите $A=\{1, 0, a\}$ задан следующей системой ориентированных подстановок:

1. $aa1\to a11$
2. $a1\to 1$
3. $aa\to a1$
4. $a\to 1$
5. $11\to a$

$6.1 \to 0$

В какое слово перерабатывает этот алгоритм слово вида $w = \underbrace{111...1}_{n\ раз}$? Т.е. какую арифметическую функцию он вычисляет?

26. Задайте нормальный алгоритм Маркова для вычисления функции $x \dot{-} y = \begin{cases} x - y, \text{если } x \geq y, \\ 0, \text{если } x < y. \end{cases}$. Проверьте правильность вашего алгоритма, построив дерево доказательства.

Глава 3. Машины Тьюринга

3.1. Описание машины Тьюринга

Я не боюсь этого слова – «ограниченность»,
ведь работа рассудка сводится к непрерывному
ограничению бесконечности, к раздроблению
этой самой бесконечности на удобные,
легко переваримые порции.
Е.И. Замятин «Мы»

В 1936 году английский математик Алан Мэтисон Тьюринг ввел понятие вычислимости на абстрактных машинах, которое, как и алгоритмы Маркова являются еще одним способом уточнения понятия алгоритма. Справедливости ради уточним, что Тьюринг спроектировал некий аналог работы современного компьютера, вернее – его программного обеспечения и сделал это гораздо раньше Маркова, хотя нормальные алгоритмы требуют меньшего функционала для своего описания.

Работы Тьюринга не были единственны в своем роде – в это же время Эмилем Леоном Постом также были разработаны аналогичные устройства. К слову сказать, Пост по месту своего рождения – русский, он родился в Польше, которая в то время входила в состав Российской Империи.

Появление практически в одно и то же время математических идей история знает не мало, некие внутриматематические процессы начинают естественный переход количества в качество и выплескиваются наружу в виде теорий и/или тео-

рем. Начало XX характеризуется в математике как «период кризиса» и способы выхода из кризиса появились в головах сразу нескольких ученых.

Немного исторических фактов сложат четкую картину происходящего. А.М.Тьюринг опубликовал первую свою работу в журнале Лондонского математического сообщества (London Mathematical Society (LMS)), том. 58, за 1936 г. (рукопись подана в редакцию – 19.04.1935 г.) под названием «О вычислимых числах, с применением к проблеме невычислимости». Она насчитывает более 20 страниц и является готовым и уже выверенным материалом для дальнейшего использования. Само общество основано 16 января 1865 г. и первым его президентом был Август де Морган, чьим именем названы известные законы алгебры высказываний.

Статья Поста вышла в свет в журнале символической логики (Jornal of Symbolic Logic) в третьем номере, за сентябрь 1936 г. Она насчитывает 3 страницы и носит более общий характер, что касается самого устройства машины, зато Пост выскажет впервые мысль об определимости невычислимости как таковой, то есть проблема в принципе неразрешима, если она не может быть вычислена на том устройстве, которое он описывает в статье. Кстати, Тьюринг использует для алфавита двоичную систему, Пост же – десятичную. Сами же устройства машин принципиально не различаются, Пост более строго описывает сами программы, Тьюринг же – устройство машин.

Приведем описание машины Тьюринга.

Представим себе некоторое устройство, снабженное сколь угодно длинной (потенциально бесконечной) лентой, разделенной на ячейки (будем называть ее лентой памяти), записывающим устройством, которое одновременно и стирает ранее записанную информацию в том месте, где пишет (например, как записывающее устройство магнитофона), *считывающей головкой*, через которую пропущена лента памяти. В процессе работы машина может пребывать в одном из следующих внутренних состояний: $S=\{s_0,s_1,s_2,\ldots,s_k\}$ и за один такт работы может обозревать одну ячейку памяти. Состояние s_1 называется начальным состоянием. s_0 – состоянием покоя (стоп состояние). Записи на лентах делаются в символах

внешнего алфавита: $A=\{a_0,a_1,..,a_n\}$. В каждую ячейку записывается только один символ. a_0 означает пустую ячейку. Заметим, что в любой момент времени t, в том числе и перед началом работы, только конечное число ячеек ленты бывают заполненными, а на всех остальных клетках будем считать, что записаны a_0.

Схематически машина Тьюринга выглядит следующим образом:

В момент времени t, в зависимости от внутреннего состояния s и от записи a в обозреваемой ячейке, машина выполняет последовательно следующие три действия:

1. Переходит в новое внутреннее состояние $s' \in S$ (если в этом нет необходимости, то остается в прежнем состоянии);

2. Печатает в обозреваемой ячейке памяти нужный символ $a' \in A$;

3. Передвигает считывающую головку вдоль ленты на одну ячейку или вправо (п), или влево (л), или же останавливается (ост.) – s_0.

Свою работу машина Тьюринга T начинает в состоянии s_1. При этом считывающая головка обозревает самую левую непустую ячейку ленты. Прочитав в состоянии s_1 символ a_{i_1}, машина, согласно своей программе, перейдет в новое внутреннее состояние s_1', напечатает вместо a_{i_1} новый символ из алфавита A или повторит a_{i_1}, и при этом либо передвинет считающую головку на одну ячейку в нужном направлении, либо остановится. В следующем такте (если не произойдет

остановка) действия машины будут определяться уже новой парой (s_1', a_{i_2}), где s_1' – состояние ее, a_{i_2} – содержание той ячейки, к которой передвинулась головка. Так будет продолжаться до тех пор, пока не произойдет остановка машины. Оставшееся при этом справа от считающей головки слово v будет называться результатом применения машины T к слову u. Другими словами, говорят, что машина T переработала слово u в слово v. Может оказаться, что машина T, начиная с некоторого слова u, работает бесконечно, никогда не останавливается. В этом случае говорят, что к слову u машина T неприменима.

Из изложенного выше можно заключить, что конкретная машина Тьюринга T представляет собой конечный набор пятерок $s_j a_j s_r a_l \lambda_l$, где λ_l означает (п) или (л) или (ост.) поэтому программу машины удобно задавать в виде прямоугольной таблицы размерами $k\times(n+1)$, где k,n – количество символов соответственно в алфавитах S и A. На пересечении i – строки, соответствующей состоянию s_i, и j-го столбца, соответствующего символу a_j внешнего алфавита, помещается информация $s_r a_l \lambda_l$ – то, что должна выполнить машина в состоянии s_i при встрече с символом a_j на ленте $(1\le i\le k,\ 0\le j\le n)$.

Пример 3.1.1 Зададим машину T_1, которая копирует начальную информацию справа от нее, пропустив предварительно одну свободную ячейку.

Допустим, что информация на ленте задана в виде конечной последовательности из единиц, а символ * означает, пустую ячейку. Кроме того, нам понадобится вспомогательный при вычислениях символ 0. Итак, $A_1=\{1,*,0\}$. Пусть $S=\{s_0,s_1,s_2,s_3,s_4,s_5\}$ внутренние состояния машины. Программу T_1 определим в виде следующей таблицы:

T_1	1	*	0
S_1	$S_2$0п	S_5 * л	–
S_2	$S_2$1п	S_3 * п	–
S_3	$S_2$1п	$S_4$1л	–
S_4	$S_4$1л	S_4 * л	$S_1$0п

S_5	–	S_6 * п	$S_5$1л
S_6	$S_6$1п	S_0 * ост.	–

В таблице машины имеются незаполненные клетки (прочерк). Это потому, что в работе машины не возникнут такие ситуации. Тем не менее, во избежание недоразумений, договоримся, что в таких ситуациях T_1 останавливается, не изменяя при этом своего внутреннего состояния и обозреваемого на ленте символа.

Отдельные такты работы машины T_1 приведем в следующей таблице №1:

Такты	Ситуации на ленте
0	$\overset{s_1}{}$...**11111**...
1	...**$\overset{s_2}{0}$1111**...
5	...**01111$\overset{s_2}{}$**...
6	...**01111*$\overset{s_3}{}$*...
7	...**01111*1$\overset{s_4}{}$**...
12	...**$\overset{s_1}{0}$1111*1**...
17	...**00111*$\overset{s_3}{}$1**...
18	...**00111*1$\overset{s_3}{}$**...
19	...**00111*$\overset{s_4}{}$11**...
25	...**00$\overset{s_1}{}$111*11**...
64	...**00000$\overset{s_1}{}$*11111**...
65	...**0000$\overset{s_5}{0}$*11111**...
66	...**0000$\overset{s_5}{1}$*11111**...
70	...*$\overset{s_5}{}$*11111*11111**...
71	...**$\overset{s_6}{}$11111*11111**...
72	...**11111$\overset{s_6}{}$*11111**...
73	$\overset{s_0}{}$*11111*11111*

В описаниях ситуаций на ленте положение считывающей головки отмечено над обозреваемым символом в виде внутреннего состояния машины. Так, в первом такте машина T_1 обозревает в ячейке ленты символ *1* в состоянии s_2 . При этом она идет вправо, не меняя своего внутреннего состояния и символа в ячейке. В такте шесть она имеет состояние s_3 и обозревает пустую ячейку. В этой ситуации, согласно программе, она печатает *1*, переходит в состояние s_4 и передвигает головку на одну ячейку влево.

Пример 3.1.2 Допустим, что на ленте задана конечная последовательность из 1. Зададим машину T_2, которая передвигала бы эту группу на одну ячейку влево и останавливалась.

Машина T_2:

T_2	1	*
S_1	$S_2$1п	S_1 * п
S_2	$S_2$1п	S_3 * л
S_3	S_4 * л	_
S_4	$S_4$1л	$S_5$1л
S_5	_	S_0 * ост.

Таблица 2:

Такты	Ситуации на ленте
0	$\overset{s_1}{...}**11111**...$
1	$...*\overset{s_2}{*}11111**...$
2	$...**\overset{s_2}{1}1111**...$
6	$...**1111\overset{s_2}{1}**...$
7	$...**11111\overset{s_3}{*}**...$
8	$...**1111\overset{s_4}{*}**...$
13	$\overset{s_5}{...}**11111**...$ остановка s_0

Пример 3.1.3 Рассмотрим машину T_3 с внутренними состояниями s_0, s_1, s_2, s_3 и с внешним алфавитом $\{1,*\}$, программа которой задана следующей таблицей:

Машина T_3:

	1	*
S_1	$S_2$1п	S_1 * п
S_2	$S_2$1п	$S_3$1п
S_3	$S_3$1л	S_0 * ост.

Машина T_3 к заданной группе единиц добавляет справа одну единицу, возвращается в исходное положение и останавливается.

Пример 3.1.4 Пусть $A_4=\{1, *\}$, $S=\{s_0,s_1,s_2,s_3,s_4\}$.

Машина T_4:

	1	*
S_1	$S_2$1п	S_1 * п
S_2	$S_2$1п	S_3 * л
S_3	S_4 * л	_
S_4	$S_4$1л	S_0 * ост.

Машина T_4 отыскивает на ленте группу единиц (справа от головки), стирает одну крайнюю правую единицу и останавливается. При этом результат работы T_4 остается справа от головки машины. Если лента в начале работы T_4 была пустой, то головка бесконечно передвигается вправо.

Пример 3.1.5 Машина T_5, заданная следующей таблицей из одной строки

	1	*
S_1	$S_0$1ост.	S_1 * п

находит на ленте группу единиц справа от считывающей головки и останавливается.

3.2. Вычисление числовых функций на машинах Тьюринга

Натуральное число n можно представить в алфавите $\{1, *\}$ в виде последовательности $(n+1)$ единиц записанных подряд. Набор натуральных чисел $(n_1,n_2,\ldots,n_k)$ можно задать в виде последовательности групп единиц, представляющих данные числа, отделенных друг от друга символом $*$.

Будем считать, что машина Тьюринга T вычисляет числовую функцию $f(n)$ от натурального аргумента n, если она, начиная свою работу с записи на ленте числа n, останавливается через конечное число тактов. При этом слева от головки должна находиться запись аргумента n, а находящаяся справа от головки информация будет называться значением функции f в точке n. Т.е. конечная ситуация на ленте должна иметь следующий вид:

$\ldots{*}{*}\underbrace{111\ldots1}_{n+1}{*}\underbrace{111\ldots1}_{f(n)+1}{*}{*}\ldots$, причем головка машины находится между числами n и $f(n)$.

Например, машин T_1 вычисляет функцию $f(x)= x$.

3.3. Композиция машин Тьюринга

Рассмотрим две машины Тьюринга T_1 и T_2, имеющие один и тот же внешний алфавит A. Пусть в начальный момент времени $t=0$ на ленте задана входная информация u и машина T_1 начинает перерабатывать u, отправляясь с левой крайней незанятой ячейки. Допустим, что в момент времени t_1 машина останавливается и выдает результат u_1 (т.е. за время t_1 машина T_1 перерабатывает слово u в слово u_1). Теперь представим, что в момент остановки T_1 начинает свою работу машина T_2 и перерабатывает слово u_1 за время t_2 в слово u_2. тогда последовательная работа машин T_1 и T_2 перерабатывает информацию u в u_2.

Имея таблицы машин T_1 и T_2, нетрудно построить таблицу новой машины T, которая выполняет ту же самую работу, что и последовательно соединенные машины T_1 и T_2. Если машина T_1 имеет k_1 внутренних состояний, а T_1-k_2 внутренних состояний, то машина T будет иметь $k \le k_1+k_2$ внутренних состояний. Таблица ма-

шины T будет состоять из двух частей: верхняя часть описывает работу машины T_1, а нижняя часть – работу машины T_2. Команда «останов» машины T_1 заменена так, чтобы начинала работать машина T_2. При этом T будет называться *композицией* машин T_1 и T_2 и будет записываться $T=T_1 \circ T_2$.

Итак, композиция двух машин Тьюринга определяет снова машину Тьюринга. Например, композиция машин T_1 и T_3 из предыдущих примеров задает машину Тьюринга T_6, вычисляющую функцию $s(x)=x+1$.

Функция $f(x)=2x$ вычисляется следующей машиной $T_7=T_1 \cdot T_5 \cdot T_1 \cdot T_2 \cdot T_4$.

В самом деле: пусть начальная информация на ленте имеет вид

$...**\overbrace{111111}^{x}**...$, после остановки машины T_1 будем иметь на ленте:

$$...**\overbrace{111111}^{x} \underset{останов}{*} \overbrace{111111}^{x}**...$$

Далее, машина T_5 передвинет головку к последней группе единиц и после этого включится машина T_1. Проработав, машина T_1, оставить на ленте следующую информацию:

$$...**\overbrace{111111}^{x}*\overbrace{111111}^{x} \underset{останов}{*} \overbrace{111111}^{x}**...$$

В полученном слове машина T_2 объединит последние две группы единиц:

$$...**\overbrace{111111}^{x} \underset{останов}{*} \overbrace{111111111111}^{2x+1}**...$$

Наконец, машина T_4 сотрет крайнюю справа единицу, и мы получим:

$$...**\underbrace{111111}_{x} \underset{останов}{*} \underbrace{11111111111}_{2x}**...$$

В этом параграфе мы ограничились рассмотрением простейших машин Тьюринга, вычисляющих некоторые числовые функции от натурального аргумента, и с натуральными значениями, и не ставили перед собой задачу выяснения того, что могут и что принципиально не могут «делать» такие машины. Однако, заметим, что этот вопрос в теории алгоритмов давно исследован и считается, что любая числовая функция, вычислимая в интуитивном смысле, вычислима и на

подходящей машине Тьюринга (тезис Тьюринга), к рассмотрению этого тезиса мы еще раз вернемся.

3.4. Алан Тьюринг

Алан Матисон Тьюринг (Alan Mathison Turing, 1912-1954 гг.) может быть причислен к плеяде составляющих гордость человечества величайших математических и философских умов, таких как Р. Декарт, Г. В. Лейбниц, Б. Рассел, Д. Гильберт, А. Витгенштейн. Тьюринг признан одним из основателей информатики и теории искусственного интеллекта, его считают первым теоретиком современного программирования.

А.Тьюринг родился 23 июня 1912 г. в одной из лондонских гостиниц, на стене, которой в последнее время установлена мемориальная доска с надписью: «Здесь родился Алан Тьюринг, взломщик кодов и пионер информатики». Отец Тьюринга состоял на государственной службе в Индии. Он не хотел рисковать и растить детей в далёких провинциях, которыми управлял, поэтому после рождения Алана отец решил оставить семью в Англии, чтобы не повергать её возможному риску. В детстве Алан и его старший брат Джон довольно редко видели своих родителей – их отец до 1926 г. служил в Индии. Дети оставались в Англии и получали строгое английское воспитание, соответствующее представителям «высшего среднего класса».

Тьюринг рано начал интересоваться наукой: химией, физикой и математикой. В 16 лет он с увлечением изучал теорию относительности Эйнштейна. В 17 лет начал изучать квантовую механику. В 1931 г. Тьюринг стал студентом знаменитого Гарвардского университета. Он стал знаменитым, будучи ещё студентом. Полученные им результаты относились к основаниям математики. Ещё в 1900 г. Д. Гильберт сформулировал несколько проблем, решение которых должно было произвести потрясение в математике. Программа Гильберта состояла в нахожде-

нии (или в доказательстве существования) общей алгоритмической процедуры, способной отвечать на вопросы, центральными из которых были следующие.

1. Является ли математика *полной*? То есть, может ли любое математическое утверждение быть доказано или опровергнуто средствами самой математики?
2. Является ли математика *непротиворечивой*? То есть, нельзя ли средствами математики доказать неверное утверждение?
3. Является ли математика *разрешимой*? То есть, существует ли общая процедура, способная доказать или опровергнуть любое математическое утверждение?

Гильберт был уверен, что ответ на каждый вопрос утвердительный. Первый удар по программе Гильберта нанёс молодой Курт Гёдель. Он показал, что в математике существуют утверждения, которые нельзя ни доказать, ни опровергнуть. Таковы утверждения вида «это утверждение недоказуемо». Существование такого рода утверждений означает *неполноту* математики. Гёдель показал, что математика не может быть одновременно непротиворечивой и полной. Алан Тьюринг «добил» программу Гильберта, доказав неразрешимость математики. Тьюринг создал абстрактную вычислительную машину, названную Чёрчем «машиной Тьюринга». Эта воображаемая машина оперировала с числами «1» и «2» и представляла собой двоичный компьютер. Тьюрингу удалось чётко определить понятие алгоритма. Вместе с тем, он показал существование задач, которые не решаются на машине Тьюринга, что доказывало неразрешимость математики.

Тьюринг стремительно ворвался на математический олимп. В 1936 г. Алонзо Чёрч пригласил его в заветный Принстоне (США), куда из немецкого Гёттингена переместился центр мировой математики и физики. Здесь работали такие выдающиеся учёные, как Джон фон Нейман, Герман Вейль, Курант, Эйнштейн. В Принстоне Тьюринг написал несколько работ, в том числе статью по лямбда-исчислению (в развитие идей Клини).

В 1938 г. Тьюринг возвращается в Англию, где вскоре военно-морское ведомство поставило перед ним задачу разгадать секрет «Энигмы» – специального

устройства для шифровки радиограмм в немецком военно-морском флоте и в «люфтваффе». Британская разведка раздобыла это устройство, но расшифровать перехваченные радиограммы немцев не удавалось. Тьюрингу была предоставлена свобода действий. Он пригласил в свой отдел «Британской школы кодов и шифров» нескольких друзей-шахматистов. 27-летнего Тьюринга и его коллег охватил настоящий спортивный азарт. Немцы считали «Энигму» неприступной. Но Тьюринг уже через полгода разработал устройство, названное «Бомбой», которое позволяло читать практически все сообщения «люфтваффе». Через год был взломан более сложный вариант «Энигмы», использовавшийся нацистскими подводниками. Это во многом определило успех британского флота.

История создания «Энигмы» связана с постройкой в Германии ещё в феврале 1918 г. Артуром Шербиусом (Arthur Scherbius:1878–1929) шифровальной машины ENIGMA, использовавшейся изначально в дипломатической работе. Машина к началу второй мировой войны претерпела изменения – она является одной из самых секретных разработок Германии.

К 1939 г. Польша сумела добыть экземпляр ENIGMA, а существовавшая к тому времени группа польских математиков-дешифровщиков сумела разобраться в методах дешифровки текстов, передаваемых с помощью ENIGMA. В июле 1939 г. 5 валиков ENIGMA и методы дешифровки текстов были переданы англичанам (недалеко от Варшавы): дешифратору Дилвину Кноксу (Dillwin Knox: 1884–1943 гг.) и руководителю морской разведки Великобритании Алестеру Деннистоуну (Alastair Denniston). Для того, чтобы достаточно быстро проводить дешировку немецких сообщений, британское правительство решило создать вычислительную машину, способную заменить десятки (или даже сотни) вычислителей и которую обслуживал бы небольшой штат для сохранения секретности [17].

Заслуги Алана Тьюринга были по достоинству оценены: после разгрома Германии он получил орден, был включён в группу разработчиков британской электронно-вычислительной машины. В 1951 г. в Манчестере начал работать один из первых в мире компьютеров. Тьюринг занимался разработкой программного обеспечения для него.

Именем Тьюринга названа высшая награда в области программирования, которая до настоящего времени служит эквивалентом Нобелевской премии.

3.5. Эмиль Леон Пост

Эмиль Леон Пост (Post Emu Leon 11.02.1897 – 21.04.1954 гг.) родился в г. Августов (польск. Augustów) на северо-востоке Польского Царства (ныне – Польша), которое в то время (с 1815 – 1915 гг.) принадлежало Российской Империи, в семье польских евреев Арнольда и Перл Пост. Семья эмигрировала в Соединенные Штаты в мае 1904 г. и поселилась в г. Нью-Йорке.

Эмиль был чрезвычайно умный ребенок, но его жизнь была одним из величайших трагедии. В детстве он потерял руку в аварии, но это было не самое худшее его увечье, ведь на протяжении всей жизни Пост страдал тяжелой формой психического расстройства – маниакально-депрессивный синдромом, который, по словам самого Поста «делал потерю руки тривиальной травмой».

В то время существовало бесплатное среднее образование для особо одаренных детей в Нью-Йорке – это была средняя школы Таунсенд Харрис (Townsend Harris) при городском колледже или Сити-колледже (City College), в котором Пост продолжил свое обучение и в 1917 г. получил степень бакалавра. Будучи студентом колледжа, он написал свою первую работу, которая была посвящена вопросам обобщенного дифференцирования. Пост не представил данный труд вовремя для публикации в журнале Американского математического общества и статья была опубликована лишь в 1930 г. Свое обучение Пост продолжил в аспирантуре Колумбийского университета. Значительным событием для Поста была публикация труда Б.Рассела и А.Н.Уайтхеда «Основания математики» («Principia Mathematica»). Первый том был опубликован в 1910 г., второй в 1912 г., а третий в 1913 г. Данный труд увлек Поста, и он принял участие в семинаре

Кассия Кайзера в Колумбийском университете, который был посвящен основаниям математики. Посту была присуждена степень магистра в 1918 г. и кандидатская степень в 1920 г., тема диссертации была посвящена функциональному уравнению гамма-функции, докторская же диссертация – математической логике, и степень доктора он получил уже в Принстонском университете. В докторской диссертации, опубликованной в 1921 г., Пост изложил метод оценки пропозициональных формул посредством таблиц истинности, в ней же впервые получен ряд фундаментальных результатов в металогике для классической логики высказываний: непротиворечивость, дедуктивная полнота, разрешимость, функциональная полнота. В этой работе впервые построена многозначная логика более чем с тремя истинностными значениями и с произвольным числом выделенных значений. Здесь же установлено, что множество замкнутых классов в классической логике счетно. Также было опубликовано полное описание решетки замкнутых классов, каждый класс строится эффективно, и показано, что каждый замкнутый класс имеет конечный базис. Эти классы названы классами Поста. Впервые определен критерий функциональной полноты, применяемый сейчас для произвольного множества функций многозначной логики. Алгебраический эквивалент многозначным логикам Поста получил название «алгебр Поста», которые интенсивно развиваются уже на протяжении полувека. В Принстоне еще в 1921 г, он вплотную подошел к открытию неполноты, но труд не стал публиковать, а в 1931 г. прогремела статья Геделя с теми же результатами. В 1924 г. Пост перешел в Корнельский университет, но из-за постоянных приступов не смог там работать и с 1927 г. работал в качестве учителя средней школы в Нью-Йорке. В 1929 г. Пост женился на Гертруде Сингер, которая родила ему дочь Филлис, а в 1932 г. был назначен доцентом в том самом Сити-колледже, где когда то учился. Это было последним его местом работы. Умер Пост в 1954 г. в возрасте 57 лет от сердечного приступа, что явилось прямым следствием лечения «шоковой терапией» его психического недуга, которое в то время осуществлялось посредствам разрядов электрического тока.

В 1936 г. независимо от работ Тьюринга, Чёрча и Клини, Постом уточнено понятие алгоритма в терминах, как бы сегодня сказали, компьютерной программы, а именно, созданием «машины Поста». В 1943 г. Постом было впервые предложено общее понятие исчисления, имеющее фундаментальное значение для доказательства неразрешимости ряда проблем математики. В 1944 г. публикуется наиболее влиятельная работа Поста, где в первоначальном виде излагается теория степеней неразрешимости, а в 1947 г. впервые в истории математики (независимо от А. А. Маркова) был указан пример «внутриматематической» неразрешимой массовой алгоритмической проблемы, а именно проблемы А. Туэ (проблема равенства для полугрупп). Пост считал – и писал об этом К. Геделю после их долгожданной встречи в 1938 г., – что за 15 лет до революционных гёделевских работ о неполноте, он уже имел эти теоремы, хотя и не в такой законченной форме. При жизни Пост не получил того признания, которого заслуживал, хотя был человек, который по словам дочери Поста «всю жизнь боролся с одной стороны с несостоятельностью самого себя, с другой стороны – с несостоятельностью математики».

Контрольная работа №3:

1. Дано число n в восьмеричной системе счисления. Постройте машину Тьюринга, которая бы увеличивала данное число на единицу.

2. Дана десятичная запись натурального числа $n > 1$. Постройте машину Тьюринга, которая уменьшала бы данное число на 1. При этом запись числа $n – 1$ не должна содержать левый нуль. Например, 100 – 1 = 99, а не 099. Начальное положение головки – правое.

3. Дан массив из открывающихся и закрывающихся скобок. Построить машину Тьюринга, которая удаляла бы пары взаимных скобок. Например, дано: «) (( ) (( )», надо получить: «)… ((.».

4. Дана строка из букв a и b. Построить машину Тьюринга, которая переместит все буквы a в левую часть строки, а все буквы b – в правую. Считывающая головка находится над крайним левым символом строки.

5. Даны два целых положительных числа в десятичной системе счисления. Построить машину Тьюринга, которая будет находить разность этих чисел, если известно, что первое число больше второго, а между ними стоит знак «-». Считывающая головка находится над левой крайней цифрой левого числа.

6. Даны два целых положительных числа в различных системах счисления, одно – в троичной системе счисления, другое – в десятичной. Постройте машину Тьюринга, которая будет находить сумму этих чисел в десятичной системе счисления.

7. Постройте машину Тьюринга, которая преобразует запись числа из двоичной системы счисления в восьмеричную.

8. Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Построить машину Тьюринга, которая будет записывать в десятичной системе счисления число этих меток.

9. На ленте машины Тьюринга в трёх клетках записаны три различные буквы: *A, B, C.* Считывающая головка в начальном состоянии обозревает букву, расположенную справа. Построить машину Тьюринга, которая поменяет местами крайние буквы.

10. На ленте машины Тьюринга записано число в десятичной системе счисления. Построить машину Тьюринга, которая бы умножала данное число на 2. В начале работы считывающая головка находится над крайней левой цифрой числа.

11. На ленте машины Тьюринга записано целое положительное число в десятичной системе счисления. Найти произведение этого числа на 11, если считывающая головка обозревает крайнюю правую цифру числа.

12. На ленте машины Тьюринга записано слово, состоящее из букв латинского алфавита $\{a, b, c, d\}$. Подсчитать число вхождений буквы a в заданное слово. Полученное значение записать на ленту левее заданного слова через пробел. Считывающая головка обозревает крайнюю левую букву.

13. На ленте машины Тьюринга записано десятичное число. Определить, делится ли это число не 5 без остатка. Если делится, то справа от числа записать

слово «да», если не делится, то записать «нет». Считывающая головка находится где-то над числом.

14. На ленте машины Тьюринга записано число в десятичной системе счисления. Считывающая головка находится над правой цифрой. Запишите цифры этого числа в обратном порядке.

15. На ленте машины Тьюринга находится массив, состоящий только из символов *A* и *B*. Сожмите массив, удавив из него все элементы *B*.

16. Число *n* записано на ленте машины Тьюринга массивом меток. Преобразуйте это значение n по формуле:

$$\begin{cases} n-2, n>5 \\ \varphi(n)=1, n=5 \\ 2n, n<5 \end{cases}$$

Считывающая головка обозревает крайнюю левую метку.

17. На ленте машины Тьюринга массив из $2n$ меток. Уменьшите этот массив в 2 раза.

18. Даны два натуральных числа *m* и *n*, записанные в унарной системе счисления. Между этими числами стоит знак «?». Выясните отношение между заданными числами и замените знак «?» на один из подходящих знаков «<», «>» или «=».

19. Найдите произведение двух натуральных чисел *m* и *n*, записанных в унарной системе счисления. Соответствующие наборы символов «|» разделены знаком «*», а справа от последнего символа стоит знак «=». Поместите результат умножение этих чисел вслед за знаком «=».

20. На ленте машины Тьюринга в трёх клетках записаны три различные буквы: *A, B, C*. Считывающая головка в начальном состоянии обозревает букву, расположенную справа. Построить машину Тьюринга, которая поменяет местами крайние буквы.

21. На ленте машины Тьюринга записано число в десятичной системе счисления. Построить машину Тьюринга, которая бы умножала данное число на

2. В начале работы считывающая головка находится над крайней левой цифрой числа.

22. На ленте машины Тьюринга записано число в двоичной системе счисления. Найти произведение этого числа на себя, если считывающая головка обозревает крайнюю правую цифру числа.

23. На ленте машины Тьюринга записано слово, состоящее из букв латинского алфавита $\{a, b, c, d\}$. Подсчитать число вхождений буквы d в заданное слово. Полученное значение записать на ленту левее заданного слова через пробел. Считывающая головка обозревает крайнюю правую букву.

24. На ленте машины Тьюринга записано десятичное число. Определить, делится ли это число на 3 без остатка. Если делится, то справа от числа записать слово «да», если не делится, то записать «нет». Считывающая головка находится где-то над числом.

25. На ленте машины Тьюринга записано число в двоичной системе. Постройте машину Тьюринга, которая преобразует запись числа в двоичной системе счисления в восьмеричную.

26. Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Построить машину Тьюринга, которая будет записывать в двоичной системе счисления число этих меток.

27. Постройте машину Тьюринга, которая преобразует запись числа из двоичной системы счисления в восьмеричную.

Контрольная работа №4

1. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x + 3$.

2. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x \dot{-} 1$.

3. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z) = x + y + z$.

4. Построить машину Тьюринга, вычисляющую функцию $f(x) = \begin{cases} 0, & \text{если } x = 0; \\ x - 1, & \text{если } x > 0. \end{cases}$

5. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x + 1$.

6. Построить машину Тьюринга, вычисляющую функцию

$$f(x) = \begin{cases} 0, & \text{если } x > 0; \\ 1, & \text{если } x = 0. \end{cases}$$

7. Построить машину Тьюринга, вычисляющую функцию

$$f(x) = \begin{cases} 2, & \text{если } x = 0; \\ 0, & \text{если } x \neq 0. \end{cases}$$

8. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = 2$.

9. Построить машину Тьюринга, вычисляющую функцию $f(x) = 4$.

10. Построить машину Тьюринга, вычисляющую функцию $f(x) = x + 3$.

11. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = 0$.

12. Построить машину Тьюринга, вычисляющую функцию

$$f(x) = \begin{cases} 0, & \text{если } x = 1; \\ 1, & \text{если } x \neq 1. \end{cases}$$

13. Построить машину Тьюринга, вычисляющую функцию $f(x, y)$, равную остатку от деления x на 2.

14. Построить машину Тьюринга, вычисляющую функцию

$$f(k) = \begin{cases} k, & \text{если } k \text{ - нечетное;} \\ 1, & \text{если } k \text{ - четное.} \end{cases}$$

15. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x + 2$.

16. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z) = 1$.

17. Построить машину Тьюринга, вычисляющую функцию

$$f(x) = \begin{cases} 1, & \text{если } x > 0; \\ 2, & \text{если } x = 0. \end{cases}$$

18. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z) = 0$.

19. Построить машину Тьюринга, вычисляющую функцию $f(x) = x + 1$.

20. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z) = x + 1$.

21. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = 1$.

22. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x \div 2$.

23. Построить машину Тьюринга, вычисляющую функцию $f(x, y) = x + y$.

24. Построить машину Тьюринга, вычисляющую функцию $f(n) = n + 2$.

25. Построить машину Тьюринга, вычисляющую функцию $f(x, y)$, равную остатку от деления *x* на 3.

Глава 4. Сведение любого алгоритма к вычислению числовой функции. Геделевская нумерация объектов

4.1. Перевод из одного алфавита в другой (кодирование)

Говорю вам: бесконечности нет.
Если мир бесконечен, то средняя
плотность материи в нем должна
быть равна нулю. А так как она не
нуль – это мы знаем, - то, следовательно,
Вселенная конечна…И ее можно вычислить
и пронумеровать.
Е.И. Замятин «Мы»

Рассмотрим для примера русский алфавит. Он содержит 33 буквы. Присоединим сюда же несколько знаков препинания: точку, запятую, вопросительный и восклицательный знаки и некоторые другие. Сопоставим теперь каждому символу полученного алфавита некоторое слово, составленное из двух символов: короткого и длинного радиосигнала, причем так, чтобы равным буквам соответствовали разные слова. Мы получим так называемую, азбуку Морзе – широко известный алфавит радистов.

Отделяя разные буквы пустым сигналом (т. е. пустым промежутком), получаем возможность перевода слов с русского алфавита на язык радиосигналов.

Возможны и переводы слов вначале на язык цифр, а далее – цифры на язык радиосигналов.

С переводом слов из одного алфавита в другой встречаемся и в математике, прежде всего с переводом чисел из одной системы счисления в другую. Например, натуральные числа в различных счислениях задаются по-разному:

1) Современная позиционная десятичная запись чисел {1, 2, 5, 9, 10, 11, …};
2) Римская система {I,II,V,VI,IX,X,XI,XX,...};
3) Славянская система $\{\tilde{a},\tilde{в},\tilde{г},\tilde{s},\tilde{i},\tilde{к},\tilde{n},...\}$.

В настоящее время большинство электронно-вычислительных машин оперируют с числами в двоичной системе счисления. Алфавит этого счисления содержит две буквы 0 и 1. Запись чисел позиционная. Число 2^n обозначается в виде $\underbrace{00\ldots0}_{n}$. Для задания любого числа *a,* в этой системе необходимо разложить его в сумму степеней двойки. Например, $15 = 2^3 + +2^2 + 2^1 + 2^0$. Следовательно, 15 = 1111. а число 25 имеет вид 11001, т. к. $25 = 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$.

Множество натуральных чисел можно задать в виде слов однобуквенного алфавита {I}, как это делалось в §3.

Определение 4.1.1 Пусть M – совокупность каких-то объектов, A – конечный алфавит. Закодировать совокупность M в алфавите A – это значит, задать однозначное отображение φ некоторого множество слов в алфавите A на множество M**.** Слово a в алфавите A, соответствующее объекту $a \in M$ называется кодом объекта α и записывается как $\alpha = \varphi(a)$.

В этом определении не требуется, чтобы φ была взаимно однозначным отображением. Поэтому в принципе один и то же объект из M может иметь несколько кодов при данном кодировании φ.

Рассмотренные выше примеры переводов слов из одного алфавита в другой являются частными случаями кодирования. Примеры кодирований, связанные с пересчетом объектов, будут даны ниже.

4.2. Нумерация пар и n-ок натуральных чисел

Определение 4.2.1 Под нумерацией некоторой совокупности M будем понимать установление взаимно-однозначного соответствия между этим множеством и множеством натуральных чисел $N=\{0,1,2,3,4,\ldots\}$.

Все упорядоченные пары (a,b) натуральных чисел многими способами можно расположить в простую последовательность. Рассмотрим, например, следующее расположение:

$$\underbrace{(0,0)}_{a+b=0},\underbrace{(0,1),(1,0)}_{a+b=1},\underbrace{(0,2),(1,1)(2,0)}_{a+b=2},\underbrace{(0,3)(1,2),(2,1)(3,0)}_{a+b=3},\ldots \qquad (1)$$

В этой последовательности пары расположены в порядке возрастания сумм их членов. А пары с одинаковой суммой членов встречаются в одной группе и расположены в порядке возрастания первого компонента. Заметим, что в (1) любая пара (m, n) встречается только один раз, а именно: на $(m+1)$-ом месте в группе с номером $(m+n)$. Занимаемый порядковый номер пары в (1) называется его номером (канторовским) и обозначается $C(m, n)$. Так, $C(0,0)=0, C(0,1)=1, C(1,0)=2, C(1,2)=7, C(21,15)=687,\ldots$

Нетрудно выразить функцию $C(x, y)$ и в явном виде: пара (x, y) находится в группе с номером $(x+y)$. Этой группе предшествуют $(x+y)$ групп с общим числом элементов $1+2+3+\ldots+(x+y)=\frac{(x+y+1)(x+y)}{2}$. Следовательно, $C(x, y)=\frac{(x+y+1)(x+y)}{2}+x=\frac{(x+y)^2+3x+y}{2}$. По номеру пары $C(x, y)=n$ можно также вычислить и её компоненты.

Обозначим через $l(n)$ и $r(n)$ соответственно левый и правый компоненты пары с номером n. Из определений этих функций следует, что $C(l(n), r(n))=n, l(C(x, y))=x, r(C(x, y))=y$. А это значит, что имея функции $l(n)$, $r(n)$ и $C(x, y)$ мы можем при необходимости переходить от упорядоченных пар натуральных чисел к их номерам и, наоборот: по номеру находить её компоненты.

С помощью тройки функций, нумерующей пары, легко получить нумерацию упорядоченных троек, четверок и т. д. n-ок натуральных чисел:

$$C^{(3)}(x_1, x_2, x_3)=C(C(x_1, x_2), x_3),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

$$C^{(n+1)}(x_1, \ldots x_n, x_{n+1}) = C^{(n)}(C(x_1, x_2), x_3, \ldots x_{n+1}).$$

При этом, нетрудно заметить: $x_n = r(k)$, $x_{n-1} = r(l(k)), \ldots, x_1 = l(l\ldots(l(k))\ldots)$, где k-номер n-ки $(x_1, x_2, \ldots x_n)$.

4.3. Сведение любого алгоритма к числовой функции. Метод Гёделя

Любой алгоритмический процесс может быть задан на множестве, мощность которого не более чем счетна. Поэтому будем считать, что есть возможность каким-то эффективным способом занумеровать все условия задач, доступных некоторому конкретному алгоритму

$$T\colon\ A_0, A_1, A_2, \ldots A_n, \ldots \qquad (2)$$

Допустим также, что множество возможных решений задач (2) образует другую последовательность

$$B_0, B_1, B_2, \ldots, B_k, \ldots \qquad (3)$$

Будем также предполагать, что имеется возможность по номеру однозначно восстановить условие задачи и ответ задачи по номеру ответа.

Допустим теперь, что алгоритм T перерабатывает условие задачи A_i с номером i в некоторое B_j из (3), где $i = 0, 1, 2, \ldots, j = 0, 1, 2, \ldots$. При этом можно говорить и о целочисленной функции φ, отображающей множество номеров условий задачи в множество номеров ответов, т. е. $\varphi(i) = j$, т. к. мы можем, отвлекаясь от условий и ответов задач, рассматривать только их номера. Функция φ будет определена для натуральных аргументов, и её значения будут так же принадлежать множеству натуральных чисел.

Интуитивно ясно, что функция $\varphi(x)$ вычислима для любого натурального n: для этого по номеру n достаточно восстановить условие задачи A_n, применить к нему алгоритм T и получит ответ B_r. А теперь номер ответа будет являться значением φ в точке n, т. е. $\varphi(n)=r$.

Ясно также, что если функция φ вычислима для любого натурального аргумента, то можно говорить о некотором алгоритме для задач класса (2): по условию задачи восстановим номер *t*, вычислим значение $\varphi(t)=r$ и по номеру *r* ответа восстановим ответ задачи.

А теперь рассмотрим примеры однозначных нумераций бесконечных множеств объектов, где имеется возможность по номеру однозначно восстановить сам объект. Для этого будем использовать принцип однозначности разложения любого натурального числа на простые множители: любое натуральное число единственном образом представимо в виде $n = P_0^{\alpha_0} \cdot P_1^{\alpha_1} \cdot P_2^{\alpha_2} \cdot \ldots \cdot P_k^{\alpha_k}$, где $P_0 = 2, P = 3, \ldots, P_k - (k+1)^e -$ простое число $0 \le \alpha_i < n$ (основная теорема арифметики).

Например, $36=2^2 \circ 3^2$, $126=2^1 \circ 3^2 \circ 5^0 \circ 7^1$, $135=2^0 \circ 3^5 \circ 5^1$. Поэтому 36 однозначно определяется парой (2, 2), а 126 упорядоченным набором (1, 2, 0, 1).

Пример 4.3.1 Рассмотрим совокупность формул исчисления высказываний в алфавите $A = \{\rightarrow, \neg, A_1, A_2, \ldots, A_n, \ldots,), (\}$. Символам элементарных высказываний припишем номера $2 \cdot i + 1$. Так, $N(A_1) = 3$, $N(A_2) = 5$, … . Далее, если формула φ_1 имеет номер *m*, а формула φ_2 - номер *n*, то номер формулы $(\varphi_1 \rightarrow \varphi_2)$ будем считать число $2^m \circ 3^n$, а номер формулы $\overline{\varphi_1}$ -число 2^m. При этом каждая формула получит свой номер и разные формулы – разные номера. Так, номером формулы $(A_2 \rightarrow A_1)$ служит число $2^5 \circ 3^3 = 864$, а для формулы $\overline{A_2 \rightarrow A_1}$ номером служит число 2^{864}. Номером аксиомы $(A_1 \rightarrow (A_2 \rightarrow A_1))$ является натуральное число $2^3 \cdot 3^{2^2 \cdot 3^3} = 2^3 \cdot 3^{864}$.

Разложив заданное натуральное число *k* по степеням простых чисел, мы можем установить, является ли *k* номером некоторой формулы исчисления высказываний и, если является, то по разложению *k* можем восстановить и саму формулу. Например, число 500 не является номером никакой формулы, т. к. $500=2^2 3^0 5^3$. А число 501 является номером пропозиционального символа A_{250}.

Таким образом, появилась возможность вместе множества каких-то формул (например, тождественно истинных) рассматривать множество их номеров и

вместе алгоритмов над формулами рассматривать функции от их номеров. Например, $\varphi(n)=1$, если n есть номер тождественно истинной формулы, 0 в остальных случаях.

Пример 4.3.2 Рассмотрим произвольную машину Тьюринга. Она однозначно определяется своей прямоугольной таблицей размером $k\times(n+1)$, где $(n+1)$ – число символов внешнего алфавита, а $(k+1)$ – число внутренних её состояний.

Как в предыдущем примере, поставим в соответствие каждой машине Тьюринга T некоторое натуральное число ω следующим образом: символу внешнего алфавита $a_0, a_1, \ldots, a_n$ поставим в соответствие нечетные числа вида $4(n+1)+1$, а символ внутренних состояний $S_0, S_1, \ldots, S_k$ – число вида $4(k+1)+3$. Например, $N(a_0)=5$, $N(a_1)=9$, $N(S_0)=7$, $N(S_1)=11$ и т. д.

Далее, пусть $N(\Pi)=1$, $N(\Lambda)=2$, $N(ост)=7$ (т. к. останов S_0), где $\Pi, \Lambda, ост$ – означает соответственно «вправо», «влево» и «останов».

Если теперь словам A, B, C уже поставлены в соответствие числа a, b, c, то слову ABC поставим в соответствие число $2^a\cdot 3^b\cdot 5^c$. Таким образом, каждой клетке таблицы машины Тьюринга T будет поставлено в соответствие некоторое натуральное число.

Далее, если слову, расположенному на пересечении i строки и j столбца поставили в соответствие число a_{ij}, то i-ой строке будет соответственно число $b_i=2^{a_{i,0}}\cdot 3^{a_{i,1}}\cdot\ldots\cdot P_n^{a_{i,n}}$, $i=1, 2, \ldots, k$. Тогда машине T будет соответствовать число $\lambda=2^{b_1}\cdot 3^{b_2}\cdot\ldots\cdot P_{k-1}^{b_k}$.

Итак, каждой машине Тьюринга будет соответствовать некоторое натуральное число λ, которое назовем его номером. Заметим, что разные машины получат разные номера и, кроме того, разложив число λ на простые множители, можем определить, является ли оно номером некоторой машины T или нет и, если является, то какой именно машины.

Описанный в этих примерах метод называется методом Гёделя. Этот метод позволяет свести любой не числовой алгоритм к вычислению некоторой числовой функции.

В заключение обоснуем в нескольких словах то, что для определения понятия алгоритма на языке функций достаточно рассмотреть только такие числовые функции f, область определения и область значения которых является множество натуральных чисел.

Любой вычислительный процесс оперирует лишь с конечно-значными числами, иррациональные числа берутся лишь в некотором своем рациональном приближении. Например, известное число e точностью до восьмого знака задается рациональной дробью 2,71828183. Следовательно, в вычислимых процессах мы сталкиваемся только с такими функциями, которые определены на множестве рациональных чисел Q.

Хорошо известно, что любое рациональное число можно задать парой целых чисел m и $n>0$ как их отношение. Предварительно пересчитав (пронумеровав) все целые числа, например, в следующем порядке:

$$0^{\underline{ой}}, 1^{\underline{ый}}, 2^{\underline{ой}}, 3^{\underline{ий}}, 4^{\underline{ый}}, 5^{\underline{ый}}, 6^{\underline{ой}}, 7^{\underline{ой}}, 8^{\underline{ой}}, \ldots,$$

$$0, 1, -1, 2, -2, 3, -3, 4, -4, \ldots$$

мы можем рациональное число α рассматривать как отношение двух натуральных номеров целых чисел m и n. В таком случае, используя некоторую тройку нумерующих функций $C(x,y)=n$, $l(n)$ и $r(n)$, можно перейти от функций $f\left(\frac{x}{y}\right)$, заданных на множестве рациональных чисел Q к функциям, заданном на множестве N, сохраняя при этом возможность, обратного преобразования: $F(n)=f\left(\frac{l(n)}{r(n)}\right)$:

$$f\left(\frac{x}{y}\right)=F(C(x,y))=f\left(\frac{(l(C(x,y)))}{r(C(x,y))}\right).$$

Вопросы для самоконтроля

1. Закодируйте слова в русском алфавите числами в двоичном исчислении.

2. Расположите положительные рациональные числа в бесконечную матрицу следующим образом: на *i*–строке выписывайте все дроби со знаменателем *i* в порядке возрастания числителей.

Постарайтесь пересчитать с помощью этой матрицы все положительные рациональные числа без повторений.

3. Вычислите канторовские номера следующих троек: *(2, 3,0), (3, 4, 2), (1, 2, 3).*

4. В примере 4.1. вычислите номера следующих формул:

а) $\left(\overline{B} \to \overline{A}\right) \to (A \to B)$;

б) $\left(A \to \left(\overline{A} \to B\right)\right)$.

Глава 5. Рекурсивные функции

5.1. Базовые функции и операции над ними

Если понятие частично рекурсивной
функции действительно является
формальным эквивалентом эффективной
вычислимости, то формулировка этого
понятия может сыграть в истории
математики роль, уступающую по значению
лишь формулировке понятия натурального числа.
Э. Пост

Опишем класс функций, который будет рассматриваться на множестве $N^* = \{0, 1, 2, 3, \ldots\}$ и на его подмножествах. Класс будет построен из трех базовых функций, посредством трех операций.

Назовем простейшими или базовыми следующие функции:

1) $O(x) = 0$ – *нуль – функция;*

2) $S(x) = x + 1$ – *функция следования;*

3) $I_n^m = (x_1, x_2, ..., x_m, ..., x_n) = x_m (1 \le m \le n)$ – *функции выбора аргумента.*

Базовые функции определены на всем множестве *N**, далее функции, обладающие таким свойством, назовем *всюду определенные*.

К простейшим функциям будем применять следующие операции:

1) *Операция суперпозиции (подстановки)*.

Даны:

- n функций: $f_1(x_1, x_2, ..., x_m)$, $f_2(x_1, x_2, ..., x_m)$, ..., $f_n(x_1, x_2, ..., x_m)$;

- функция $\varphi(y_1, y_2, ..., y_n)$.

Говорят, что функция $\psi(x_1, x_2, ..., x_m)$ получена ***операцией суперпозиции (подстановки)*** f_i в φ, если $\psi(x_1, x_2, ..., x_n) = \varphi(f_1(x_1, x_2, ..., x_m), ..., f_n(x_1, x_2, ..., x_m))$. Заметим, что не играет никакой роли от каких переменных задана функция φ, зато принципиально количество этих переменных, ведь оно должно совпадать с числом n функций f_i. Сами же функции f_i должны быть определены от одних и тех же переменных. Не смотря на строгость определения операции суперпозиции, на практике часто применяется прием введения фиктивных переменных, если исходные функции определены от различных переменных.

Оговоримся, что для наилучшего понимания того, как действует та или иная операция, на начальном этапе будем использовать в качестве иллюстрации и базовые функции и те, которые базовыми не являются, но знакомы нам из еще из школьного курса математики, такие как сумма, произведение и т.д. Строгие формулировки для построения нового класса функций будут введены ниже.

Пример 5.1.1:

Выяснить, какая функция является результатом операции суперпозиции:

а) $f_1(x_1, x_2) = x_1 + x_2$, $f_2(x_1, x_2) = x_1 x_2$, $f_3(x_1, x_2) = x_1 + x_2 + x_1 x_2$ в

$\varphi(y_1, y_2, y_3) = y_1 + y_2 - y_3$;

б) $O(x) = 0$ в $S(x) = x + 1$;

в) $S(x) = x + 1$ в $O(x) = 0$;

г) $f_1(x_1, x_2) = x_1 + x_2$, $f_2(x) = x$, $f_3(x_1, x_2) = x_1 + x_2 + x_1 x_2$ в

$\varphi(y_1, y_2, y_3) = y_1 + y_2 + y_3$.

Решение: результатом операции будет функция

а) $\psi(x_1, x_2) = \varphi(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2)) = f_1(x_1, x_2) + f_2(x_1, x_2) - f_3(x_1, x_2) =$

$= x_1 + x_2 + x_1 x_2 - (x_1 + x_2 + x_1 x_2) = 0$. Таким образом $\psi(x_1, x_2) = 0$

б) $\psi(x) = S(O(x)) = S(0) = 1$, рассуждать можно и так $\psi(x) = S(O(x)) =$

$= O(x) + 1 = 0 + 1 = 1$

в) $\psi(x) = O(S(x)) = O(x+1) = 0$.

Заметим, что без разницы, какие из функций раскрывать первыми – f_i или φ. В а) раскрывалась сначала φ, затем – f_i, в) – наоборот, в б) – оба способа, г) количество переменных у функций $f_1(x_1, x_2)$, $f_2(x)$, $f_3(x_1, x_2)$ разное и применить операцию суперпозиции нельзя, но можно ввести фиктивные переменные, которые позволят применить данную операцию: $f_1(x_1, x_2) = x_1 + x_2$, $f_2(x_1, x_2) =$

$= x_1 + x_2^0 - 1$, $f_3(x_1, x_2) = x_1 + x_2 + x_1 x_2$, у функции f_2 была переобозначена переменная x на x_1 и введена фиктивная переменная x_2. Применим операцию суперпозиции: $\psi(x_1, x_2) = \varphi(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2)) = f_1(x_1, x_2) + f_2(x_1, x_2) + f_3(x_1, x_2) =$

$= x_1 + x_2 + x_1 + x_2^0 - 1 + x_1 + x_2 + x_1 x_2 = 3x_1 + 2x_2 + x_1 x_2$.

Пример 5.1.2:

Получить из базовых функций с помощью операции суперпозиции следующие функции:

а) $\psi(x) = 4$;

б) $\psi(x) = a$ (a - константа);

в) $\psi(x) = x + a$.

Решение:

а) $\psi(x) = S(S(S(S(O(x))))) = S(S(S(S(0)))) = S(S(S(1))) = S(S(2)) = S(3) = 4$ – это применение последовательно четырех операций суперпозиций. Первая – $O(x)$ в $S(x)$, вторая – $S(O(x))$ в $S(x)$ и т.д.

б) $\psi(x) = S(\ldots(S(O(x)))\ldots) = a$ – это применение последовательно a раз операций суперпозиций.

в) $\psi(x) = S(\ldots(S(I_1^1(x)))\ldots) = x + a$ – аналогично предыдущему примеру, только первая суперпозиция использует функцию $I_1^1(x) = x$.

2) *Операция примитивной рекурсии (рекурсии).*

Даны:

- функция от n переменных $f_1(x_1, x_2, ..., x_n)$ $(n > 0)$;
- функция от $n + 2$ переменных $f_2(x_1, x_2, ..., x_n, x_{n+1}, x_{n+2})$ $(n > 0)$.

Говорят, что функция от $n + 1$ переменной $\varphi(x_1, x_2, ..., x_n, x_{n+1})$ получена из функций f_1 и f_2 посредством ***операции рекурсии***, если:

$$\begin{cases} \varphi(x_1, x_2, \ldots, x_n, 0) = f_1(x_1, x_2, \ldots, x_n) \\ \varphi(x_1, x_2, \ldots, x_n, 1) = f_2(x_1, x_2, \ldots, x_n, 0, \varphi(x_1, x_2, \ldots, x_n, 0)) \\ \varphi(x_1, x_2, \ldots, x_n, 2) = f_2(x_1, x_2, \ldots, x_n, 1, \varphi(x_1, x_2, \ldots, x_n, 1)) \\ \varphi(x_1, x_2, \ldots, x_n, 3) = f_2(x_1, x_2, \ldots, x_n, 2, \varphi(x_1, x_2, \ldots, x_n, 2)) \\ \ldots\ldots \\ \varphi(x_1, x_2, \ldots, x_n, y) = f_2(x_1, x_2, \ldots, x_n, y-1, \varphi(x_1, x_2, \ldots, x_n, y\text{-}1)) \end{cases}$$

Функция f_1 используется только один раз – при определении функции φ на наборе $(x_1, x_2, \ldots, x_n, 0)$, в остальных наборах переменных данная функция определяется через значение f_2 от предыдущего набора переменных. Функция определяется не аналитически, а по аналогии с рекуррентными соотношениями. Поэтому основная задача при построении функций данной операцией – какой аналитический вид она будет иметь, и наоборот: какой рекурсией (иметься в виду из каких функций) данную аналитическую функцию можно получить. Для наилучшего понимая можно вспомнить числа Фибоначчи: функция вида: $f(x + 2) = f(x) + f(x + 1)$ и $f(0)=1$, $f(1)=1$, определенная на N дает ряд Фибоначчи, зато аналитически эта же функция имеет более сложный вид:

$$f(x) = \frac{\sqrt{5}}{5}\left(\frac{1+\sqrt{5}}{2}\right)^{x+1} - \frac{\sqrt{5}}{5}\left(\frac{1-\sqrt{5}}{2}\right)^{x+1}.$$

Также стоит обратить внимание, что количество переменных для функции φ – $n + 1$, и если нам в результате понадобиться функция от трех переменных, то для этого будут необходимы – функции от двух и четырех переменных.

Пример 5.1.3:

Какой аналитический вид имеет функция φ, которая получена операцией рекурсии из функций:

а) $f_1(x) = x$ и $f_2(x, y, z) = z + 1$;

б) $f_1(x) = 0$ и $f_2(x, y, z) = x + z$;

в) $f_1(x) = x$ и $f_2(x, y, z) = x + y + z$;

г) $f_1(x, y) = x \cdot y$ и $f_2(x, y, z, s) = x + y + z + s$.

Решение:

а) вычислим несколько значений функции $\varphi(x, y)$, по определению:

$$\begin{cases} \varphi(x, 0) = f_1(x) = x \\ \varphi(x, 1) = f_2(x, 0, \varphi(x, 0)) = f_2(x, 0, x) = x + 1 \\ \varphi(x, 2) = f_2(x, 1, \varphi(x, 1)) = f_2(x, 1, x + 1) = x + 2 \\ \varphi(x, 3) = f_2(x, 2, \varphi(x, 2)) = f_2(x, 2, x + 2) = x + 3 \\ \ldots. \end{cases}$$

Рекурсия работает так $\varphi(x, 1) = f_2(x, 0, \varphi(x,0)) = f_2(x, 0, x)$, т.к. $f_2(x, y, z) = z + 1$, т.е. прибавление в третьей переменной единицы, в данном случае третья переменная – x, то $f_2(x, 0, x) = x + 1$. Проанализируем полученные результаты:

$\varphi(x, 0) = x + 0$

$\varphi(x, 1) = x + 1$

$\varphi(x, 2) = x + 2$

$\varphi(x, 3) = x + 3$

…

Очевидно, что аналитический вид $\varphi(x, y) = x + y$.

б) вычислим несколько значений функции $\varphi(x, y)$, по определению рекурсии:

$$\begin{cases} \varphi(x, 0) = f_1(x) = 0 \\ \varphi(x, 1) = f_2(x, 0, \varphi(x, 0)) = f_2(x, 0, 0) = x + 0 = x \\ \varphi(x, 2) = f_2(x, 1, \varphi(x, 1)) = f_2(x, 1, x) = x + x = 2x \\ \varphi(x, 3) = f_2(x, 2, \varphi(x, 2)) = f_2(x, 2, 2x) = x + 2x = 3x \\ \ldots. \end{cases}$$

Проанализируем полученные результаты:

$\varphi(x, 0) = x \cdot 0$

$\varphi(x, 1) = x \cdot 1$

$\varphi(x, 2) = x \cdot 2$

$\varphi(x, 3) = x \cdot 3$

…

Очевидно, что аналитический вид $\varphi(x, y) = x \cdot y$.

в) вычислим несколько значений функции $\varphi(x, y)$, по определению рекурсии:

$$\begin{cases} \varphi(x, 0) = f_1(x) = x \\ \varphi(x, 1) = f_2(x, 0, \varphi(x, 0)) = f_2(x, 0, x) = x + 0 + x = 2x \\ \varphi(x, 2) = f_2(x, 1, \varphi(x,1)) = f_2(x, 1, 2x) = x + 1 + 2x = 3x + 1 \\ \varphi(x, 3) = f_2(x, 2, \varphi(x, 2)) = f_2(x, 2, 3x + 1) = x + 2 + 3x + 1 = 4x + 3 \\ \varphi(x, 4) = f_2(x, 3, \varphi(x, 3)) = f_2(x, 3, 4x + 3) = x + 3 + 4x + 3 = 5x + 6 \\ \varphi(x, 5) = f_2(x, 2, \varphi(x, 2)) = f_2(x, 2, 3x + 1) = x + 2 + 3x + 1 = 6x + 10 \\ \ldots. \end{cases}$$

Очевидно, что коэффициент при x для функции $\varphi(x, y) - y + 1$, т.е. $\varphi(x, y) = x(y + 1) + a$, где a – константа, зависящая лишь от y. Не всегда очевиден результат рекурсии, посмотрим, как получаются числа 0, 0, 1, 3, 6, 10, …

Выпишем зависимость:

$y = 0 \quad a = 0$

$y = 1 \quad a = 0$

$y = 2 \quad a = 1$

$y = 3 \quad a = 3$

$y = 4 \quad a = 6$

$y = 5 \quad a = 10$

При $y = 2$, a является суммой чисел от 0 до 1, т.е. $a = 1 + 0 = 1$. При y = 3, a также представляет собой сумму чисел, только уже от 0 до 2, т.е. $a = 2 + 1 + 0 = 3$. Если $y = 4$, a - сумма чисел от 0 до 3, т.е. $a = 3 + 2 + 1 + 0 = 6$. Обобщаем: при произвольном y, a есть сумма чисел от 0 до y - 1, т.е. $a = 0 + 1 + 2 + 3 + \ldots +$

$+ (y - 1) = \frac{y(y-1)}{2}$ - свернули по известной формуле суммы первых $(y - 1)$ членов натурального ряда. Получаем следующую формулу $\varphi(x, y) = x(y + 1) + \frac{y(y-1)}{2}$.

Заметим, что даже для $\varphi(x, 0)$ формула оказалась верна, хотя иногда именно этот набор переменных (в общем виде имеется в виду $\varphi(x_1, x_2,\ldots, x_n, 0)$) не поддается общей аналитической формуле и приходиться его доопределять отдельно. Более того, первый шаг рекурсии отличается от остального цикла, ведь $\varphi(x_1, x_2,\ldots, x_n, 0) = f_1(x_1, x_{2,\ldots}, x_i)$ фактически задает начальное положение, с которого в дальнейшем начитает крутиться весь цикл.

г) функция, которую мы получим, будет от трех переменных, т.к. изначально даны функции от двух и четырех переменных. Вычислим несколько значений функции $\varphi(x, y, z)$, по определению:

$$\begin{cases} \varphi(x, y, 0) = f_1(x, y) = x \cdot y \\ \varphi(x, y, 1) = f_2(x, y, 0, \varphi(x, y, 0)) = f_2(x, y, 0, xy) = x + y + 0 + xy = x + y + xy \\ \varphi(x, y, 2) = f_2(x, y, 1, \varphi(x, y, 1)) = f_2(x, y, 1, x + y + xy) = x + y + 1 + x + y + xy = \\ = 2x + 2y + xy + 1 \\ \varphi(x, y, 3) = f_2(x, y, 2, \varphi(x, y, 2)) = f_2(x, y, 2, 2x + 2y + xy + 1) = x + y + 2 + 2x + \\ + 2y + xy + 1 = 3x + 3y + xy + 3 \\ \varphi(x, y, 4) = f_2(x, y, 3, \varphi(x, y, 3)) = f_2(x, y, 3, 3x + 3y + xy + 3) = x + y + 3 + 3x + \\ + 3y + xy + 3 = 4x + 4y + xy + 6 \\ \ldots \end{cases}$$

Используя результаты предыдущего примера, имеем: $\varphi(x, y, z) = z(x + y) + xy + \frac{z(z-1)}{2}$.

Операции суперпозиции и рекурсии обладают следующими свойствами:

1^0. Сохранение всюду определенности.

2^0. Сохранение алгоритмической вычислимости.

Данные свойства означают, что применение операции суперпозиции к вычислимым и всюду определенным функциям дает новую функцию, которая будет также и вычислима и всюду определена. Аналогично для операции рекурсии.

Продемонстрируем применение операции рекурсии на более сложных функциях:

1. Дано: $g(x) = 1,\ h(x, y, z) = (y + 1) \cdot z$;

$$\begin{cases} f(x,0) = g(x) = 1, \\ f(x,y) = h(x, y-1, f(x,y-1)) = y \cdot f(x,y-1). \end{cases}$$

$f(x,0) = 1,$

$f(x,1) = 1 \cdot 1 = 1,$

$f(x,2) = 2 \cdot 1 = 2,$

$f(x,3) = 3 \cdot 2 = 6,$

$f(x,4) = 4 \cdot 6 = 24,$

……

$f(x,y) = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot y = y!.$

Получили: $f(x,y) = x \cdot 0 + y!;$

2. Дано: $g(x) = x,\ h(x,y,z) = x + y - z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x + (y-1) - f(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x + (1-1) - x = 0,$

$f(x,2) = x + (2-1) - 0 = x + 1,$

$f(x,3) = x + (3-1) - (x+1) = 1,$

$f(x,4) = x + (4-1) - 1 = x + 2,$

$f(x,5) = x + (5-1) - (x+2) = 2,$

……

$$f(x,y) = \frac{x - (-1)^{y+1} \cdot x}{2} + \frac{y}{2} + \frac{\left(-\frac{1}{2}\right) - (-1)^y \cdot \left(-\frac{1}{2}\right)}{2}.$$

Получили: $f(x,y) = \dfrac{x(1-(-1)^{y+1}) + y + \left(-\frac{1}{2}\right)(1-(-1)^y)}{2};$

3. Дано: $g(x) = x,\ h(x,y,z) = x - y - z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x - (y-1) - f(x,y-1). \end{cases}$$

$f(x,0)=x$,

$f(x,1)=x-(1-1)-x=0$,

$f(x,2)=x-(2-1)-0=x-1$,

$f(x,3)=x-(3-1)-(x-1)=-1$,

$f(x,4)=x-(4-1)-(-1)=x-2$,

$f(x,5)=x-(5-1)-(x-2)=-2$,

……

$$f(x,y)=\frac{x-(-1)^{y+1}\cdot x}{2}-\frac{y}{2}+(-1)^y\cdot\frac{\left(-\frac{1}{2}\right)-(-1)^y\cdot\left(-\frac{1}{2}\right)}{2}.$$

Получили: $f(x,y)=\dfrac{x(1-(-1)^{y+1})-y+(-1)^y\cdot\left(-\frac{1}{2}\right)(1-(-1)^y)}{2}$;

4. Дано: $g(x)=0$, $h(x,y,z)=x+z$;

$$\begin{cases} f(x,0)=g(x)=0, \\ f(x,y)=h(x,y-1,f(x,y-1))=x+f(x,y-1). \end{cases}$$

$f(x,0)=0$,

$f(x,1)=x+0=x$,

$f(x,2)=x+x=2x$,

$f(x,3)=x+2x=3x$,

$f(x,4)=x+3x=4x$,

$f(x,5)=x+4x=5x$,

……

$f(x,y)=y\cdot x$.

Получили: $f(x,y)=x\cdot y$;

5. Дано: $g(x)=0$, $h(x,y,z)=x-z$;

$$\begin{cases} f(x,0)=g(x)=0, \\ f(x,y)=h(x,y-1,f(x,y-1))=x-f(x,y-1). \end{cases}$$

$f(x,0)=0$,

$f(x,1) = x - 0 = x$,

$f(x,2) = x - x = 0$,

$f(x,3) = x - 0 = x$,

$f(x,4) = x - x = 0$,

……

$f(x,y) = \dfrac{x(1-(-1)^y)}{2}$.

Получили: $f(x,y) = \dfrac{x(1-(-1)^y)}{2}$;

6. Дано: $g(x) = x$, $h(x,y,z) = x + z$;

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x + f(x,y-1). \end{cases}$$

$f(x,0) = x$,

$f(x,1) = x + x = 2x$,

$f(x,2) = x + 2x = 3x$,

$f(x,3) = x + 3x = 4x$,

$f(x,4) = x + 4x = 5x$,

……

$f(x,y) = (y+1) \cdot x$.

Получили: $f(x,y) = x \cdot (y+1)$;

7. Дано: $g(x) = x$, $h(x,y,z) = x - z$;

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x - f(x,y-1). \end{cases}$$

$f(x,0) = x$,

$f(x,1) = x - x = 0$,

$f(x,2) = x - 0 = x$,

$f(x,3) = x - x = 0$,

$f(x,4) = x - 0 = x$,

……

$$f(x,y)=\frac{x(1-(-1)^{y+1})}{2}.$$

Получили: $f(x,y)=\frac{x(1-(-1)^{y+1})}{2}$;

8. Дано: $g(x)=x$, $h(x,y,z)=y^2-z$;

$$\begin{cases} f(x,0)=g(x)=x, \\ f(x,y)=h(x,y-1,f(x,y-1))=(y-1)^2-f(x,y-1). \end{cases}$$

$f(x,0)=x$,

$f(x,1)=(1-1)^2-x=-x$,

$f(x,2)=(2-1)^2-(-x)=1+x$,

$f(x,3)=(3-1)^2-(1+x)=3-x$,

$f(x,4)=(4-1)^2-(3-x)=6+x$,

$f(x,5)=(5-1)^2-(6+x)=10-x$,

$f(x,6)=(6-1)^2-(10-x)=15+x$,

……

$$f(x,y)=(-1)^y\cdot x+\frac{y(y-1)}{2}.$$

Получили: $f(x,y)=(-1)^y\cdot x+\frac{y(y-1)}{2}$;

9. Дано: $g(x)=x$, $h(x,y,z)=y^2+z$;

$$\begin{cases} f(x,0)=g(x)=x, \\ f(x,y)=h(x,y-1,f(x,y-1))=(y-1)^2+f(x,y-1). \end{cases}$$

$f(x,0)=x$,

$f(x,1)=(1-1)^2+x=x$,

$f(x,2)=(2-1)^2+x=1+x$,

$f(x,3)=(3-1)^2+(1+x)=5+x$,

$f(x,4)=(4-1)^2+(5+x)=14+x$,

$f(x,5) = (5-1)^2 + (14+x) = 30+x,$

$f(x,6) = (6-1)^2 + (30+x) = 55+x,$

……

$f(x,y) = x + \dfrac{y(y-1)(2y-1)}{6}.$

Получили: $f(x,y) = x + \dfrac{y(y-1)(2y-1)}{6};$

10. Дано: $g(x) = x,\ h(x,y,z) = x \cdot y + z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x \cdot (y-1) + f(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x \cdot (1-1) + x = x,$

$f(x,2) = x \cdot (2-1) + x = 2x,$

$f(x,3) = x \cdot (3-1) + 2x = 4x,$

$f(x,4) = x \cdot (4-1) + 4x = 7x,$

$f(x,5) = x \cdot (5-1) + 7x = 11x,$

……

$f(x,y) = x \cdot \dfrac{y^2 - y + 2}{2}.$

Получили: $f(x,y) = x \cdot \dfrac{y^2 - y + 2}{2};$

11. Дано: $g(x) = x,\ h(x,y,z) = x \cdot y - z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x \cdot (y-1) - f(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x \cdot (1-1) - x = -x,$

$f(x,2) = x \cdot (2-1) - (-x) = 2x,$

$f(x,3) = x \cdot (3-1) - 2x = 0,$

$f(x,4) = x \cdot (4-1) - 0 = 3x,$

$f(x,5) = x \cdot (5-1) - 3x = x,$

$f(x,6) = x \cdot (6-1) - x = 4x,$

$f(x,7) = x \cdot (7-1) - 4x = 2x,$

$f(x,8) = x \cdot (8-1) - 2x = 5x,$

$f(x,9) = x \cdot (9-1) - 5x = 3x,$

……

$$f(x,y) = x \cdot \left(\frac{1-(-1)^{y+1}}{2} \cdot \frac{y+2}{2} + \frac{1-(-1)^{y}}{2} \cdot \frac{y-3}{2} \right).$$

Получили: $f(x,y) = \dfrac{x(2y-1+(-1)^{y} \cdot 5)}{4};$

12. Дано: $g(x) = x,\ h(x,y,z) = z^{y+1};$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = f^{y}(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x^{1} = x,$

$f(x,2) = (x^{1})^{2} = x^{2},$

$f(x,3) = (x^{2})^{3} = x^{6},$

$f(x,4) = (x^{6})^{4} = x^{24},$

……

$f(x,y) = x^{y!}.$

Получили: $f(x,y) = x^{y!};$

13. Дано: $g(x) = x,\ h(x,y,z) = x \cdot (y+1) \cdot z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x \cdot y \cdot f(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x \cdot 1 \cdot x = x^{2},$

$f(x,2) = x \cdot 2 \cdot x^2 = 2x^3,$

$f(x,3) = x \cdot 3 \cdot 2x^3 = 6x^4,$

$f(x,4) = x \cdot 4 \cdot 6x^4 = 24x^5,$

……

$f(x,y) = y! \cdot x^{y+1}.$

Получили: $f(x,y) = x^{y+1} \cdot y!;$

14. Дано: $g(x) = x^2,\ h(x,y,z) = x^2 + (y+1)^2 + z;$

$$\begin{cases} f(x,0) = g(x) = x^2, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x^2 + y^2 + f(x,y-1). \end{cases}$$

$f(x,0) = x^2,$

$f(x,1) = x^2 + 1^2 + x^2 = 2x^2 + 1,$

$f(x,2) = x^2 + 2^2 + (2x^2 + 1) = 3x^2 + 5,$

$f(x,3) = x^2 + 3^2 + (3x^2 + 5) = 4x^2 + 14,$

$f(x,4) = x^2 + 4^2 + (4x^2 + 14) = 5x^2 + 30,$

$f(x,5) = x^2 + 5^2 + (5x^2 + 30) = 6x^2 + 55,$

……

$$f(x,y) = (y+1) \cdot x^2 + \frac{y(y+1)(2y+1)}{6}.$$

Получили: $f(x,y) = x^2 \cdot (y+1) + \frac{y(y+1)(2y+1)}{6};$

15. Дано: $g(x) = x,\ h(x,y,z) = x \cdot (y+2) + z;$

$$\begin{cases} f(x,0) = g(x) = x, \\ f(x,y) = h(x, y-1, f(x,y-1)) = x \cdot (y+1) + f(x,y-1). \end{cases}$$

$f(x,0) = x,$

$f(x,1) = x \cdot (1+1) + x = 3x,$

$f(x,2) = x \cdot (2+1) + 3x = 6x,$

$f(x,3) = x \cdot (3+1) + 6x = 10x,$

$f(x,4) = x \cdot (4+1) + 10x = 15x$,

$f(x,5) = x \cdot (5+1) + 15x = 21x$,

……

$$f(x,y) = \frac{x(y+1)(y+2)}{2}.$$

Получили: $f(x,y) = \frac{x(y+1)(y+2)}{2}$.

3) *Операция минимизации (μ минимизации).*

Дана:

- функция от n переменных $f(x_1, x_2, ..., x_n)$ $(n > 0)$;

Говорят, что функция μ от от n - 1 переменной $x_1, x_2, \ldots, x_{n-1}$ получена из функций f посредством ***операции минимизации***, если $\mu(x_1, x_2, \ldots, x_{n-1}) = x_n$, где x_n – минимальное решение уравнения $f_1(x_1, x_2, ..., x_n) = 0$.

При практическом применении данной операции фактически стоит задача выразить последнюю переменную через остальные из $f(x_1, x_2, ..., x_n) = 0$ и полученное выражение и есть искомая функция от всех переменных, кроме последней. То есть, если исходная функция от двух переменных, то достаточно выразить первую переменную через вторую из равенства нулю самой функции.

Пример 5.1.4:

Выяснить, какая функция получиться из данной с помощью операции минимизации:

а) $f(x, y) = x + y$;

б) $f(x, y, z) = - x + y + z$;

в) $f(x, y) = x \cdot y$.

Решение:

а) Функция $f(x, y) = x + y$ от двух переменных, результат применения операции минимизации будет функция от одной первой переменной, то есть $\mu(x)$ и равна такому y, что $f(x, y) = x + y = 0$. Выразим y из равенства $x + y = 0$, $y = - x$, то есть $\mu(x) = - x$.

б) Функция $f(x, y, z) = -x + y + z$ от трех переменных, результат применения операции минимизации будет функция от первых двух переменных, то есть $\mu(x, y)$ и равна такому z, что $f(x, y, z) = -x + y + z = 0$. Выразим z из равенства $-x + y + z = 0$, $z = x - y$, значит $\mu(x,y) = x - y$.

в) Функция $f(x, y) = x \cdot y - 1$ от двух переменных, результат применения операции минимизации будет функция от одной первой переменной, то есть $\mu(x)$ и равна такому y, что $f(x, y) = x \cdot y - 1 = 0$. Выразим y из равенства $x \cdot y - 1 = 0$, $y = \frac{1}{x}$, то есть $\mu(x) = \frac{1}{x}$.

Заметим, что функция $\mu(x)$ не является всюду определенной, выпадает значение $x = 0$, хотя изначальная функция $f(x, y) = x \cdot y - 1$ – всюду определенная. Это означает, что операция минимизации не обладает свойством 1^0, то есть не сохраняет всюду определенность. Свойство же 2^0 для операции минимизации имеет место быть.

5.2. Примитивно рекурсивные, общерекурсивные и частично рекурсивные функции

Определение 5.2.1 Функция, полученная из базовых функций, с помощью конечного числа операций рекурсий и/или суперпозиций называется ***примитивно рекурсивной (ПР).***

Определение 5.2.2 Функция, полученная из базовых функций, с помощью конечного числа операций рекурсий и/или суперпозиций и/или минимизаций называется ***частично рекурсивной (ЧР).***

Определение 5.2.3 Частично рекурсивная функция, которая определена на всем множестве N^* (или является всюду определенной) называется ***общерекурсивной (ОР).***

Заметим, что мы не дали определение ***общепримитивной функции***, так как обе операции, участвующие в определении примитивной рекурсивности обладают свойством сохранять всюду определенность, значит примитивно рекурсивные

функции и так всюду определены, в отличие от частично рекурсивных, в определении которых участвует операция минимизации, которая свойством сохранения всюду определенности не обладает.

Таким образом, основная задача теории рекурсивных функций сводится к доказательству примитивной или частичной рекурсивности той или иной функции. Также интересен и вопрос взаимного соотношения этих классов. Ведь если удалось показать примитивную рекурсивность функции, автоматически следует и ее частичная рекурсивность. Хотя обратное не верно, из частичной рекурсивности не следует примитивная рекурсивность. Разница в определениях частичной и примитивной рекурсивности в операции минимизации. Если функция получена из базовых с помощью операции (или операций) минимизации (при этом неважно, использовались ли иные операции или нет), то это и означает ее частичную, но не примитивную рекурсивность.

Пример 5.2.1:

Выяснить, какими являются исследуемые функции:

а) $\varphi(x, y) = x + y$;

б) $\varphi(x, y) = x \cdot y$;

в) $\varphi(x, y) = x^y$;

г) $\varphi(x, y) = -x$.

Решение:

Фактически стоит задача доказать примитивную рекурсивность и/или частичную рекурсивность. Первые два примера уже были рассмотрены, но не с позиции доказательства той или иной рекурсивности.

а) $\varphi(x, y) = x + y$, данная функция получается с помощью рекурсии $f_1(x) = x$ и $f_2(x, y, z) = z + 1$ (прим. 5.1.3, а). Очевидно, что первая функция – базовая. Действительно, $I_1^1(x) = x = f_1(x)$ (с тем же успехом возможно представить $f_1(x)$ как $I_2^1(x, y) = x$ и т.п.). Вторая же функция – это суперпозиция функции выбора аргумента в функцию следования: $S(I_2^1(x, y, z)) = z + 1$. Значит $\varphi(x, y) = x + y$ – прими-

тивно рекурсивная, т.к. получена из базовых с помощью одной операции суперпозиции и одной рекурсии.

б) $\varphi(x, y) = x \cdot y$ – это рекурсия $f_1(x) = 0$ и $f_2(x, y, z) = x + z$. Покажем, что функции, используемые для рекурсии или базовые или полученные из базовых конечным числом операции рекурсии и суперпозиции. Первая функция – это нуль-функция. Вторая уже является примитивно рекурсивной по доказанному выше (прим. 4, а). Значит и функция $\varphi(x, y) = x \cdot y$ – также примитивно рекурсивна.

в) $\varphi(x, y) = x^y$ – это рекурсия $f_1(x) = 0$ и $f_2(x, y, z) = x + z$.

Заметим, что первые три примера также являются и частично рекурсивными, и общерекурсивными функциями, так как функции, которые мы использовали для их конструирования, всюду определены.

г) $\varphi(x, y) = -x$ – воспользуемся примерами 5.1.4 а и 5.2.1 а. Доказав частичную рекурсивность $\varphi_1(x, y) = x + y$ из которой операцией минимизации получили функцию $\mu(x) = -x$ имеем частичную рекурсивность последней. Также данная функция не является и общерекурсивной, так как определена не всюду.

Рассмотрим вопрос о включениях в рассмотренных классах функций (К). Из определения ЧР и ПР следует, что $К_{ПР} \subset К_{ЧР}$, далее из определения ОР следует $К_{ОР} \subseteq К_{ЧР}$. Таким образом $К_{ПР}$ и $К_{ОР}$ – подмножества множества $К_{ЧР}$. Очевидно, что любая ПР является ОР, но существуют функции, которые не являются примитивно рекурсивными, но являются общерекурсивными [Роджерс, С.25].

Термины «частично рекурсивная функция» и «общерекурсивная функция» прижились в силу исторических причин и, по сути, являются результатом неточного перевода английских терминов partial recursive function и total recursive function, которые по смыслу более правильно переводить как «рекурсивные функции, определенные на части множества возможных аргументов» и «рекурсивные функции, определенные на всём множестве возможных аргументов». Наречие «частично» относится не к прилагательному «рекурсивные», а к области определения функции. Возможно, более правильным названием было бы «частично определённые рекурсивные функции» и просто «везде определённые рекурсивные функции», но длинные названия не прижились.

Результатом теории рекурсивных функций является ***тезис Черча***: любая вычислимая функция является частично примитивной. Данный тезис аналогичен тезису Тьюринга и позволяет сформулировать теорему Тьюринга-Черча: любая примитивно рекурсивная функция может быть вычислена на некой машине Тьюринга и любая машина Тьюринга вычисляет функцию, которая является частично примитивной. Удивительно, что столь разные по описанию теории в конечном итоге описали один и то же класс вычислимых функций, что формально «приравнивает» эти теории. Тезисы Черча и Тьюринга не могут сменить свой статус на статусы теорем, ведь они интуитивное понятие «вычислимой функции» приравнивают к строго определенным понятиям «частично рекурсивной функции» и «машине Тьюринга». По аналогии можно сформулировать и тезис Маркова, и соответственно теоремы: Маркова-Черча, Маркова-Тьюринга, проделайте это самостоятельно.

Контрольная работа №5

Какой аналитический вид имеет функция, являющаяся результатом применения операции рекурсии к данным функциям:

1. $g(x)=1;\ h(x,y,z)=(y+1)\cdot z;$
2. $g(x)=x;\ h(x,y,z)=x+y-z;$
3. $g(x)=x;\ h(x,y,z)=x-y-z;$
4. $g(x)=0;\ h(x,y,z)=x+z;$
5. $g(x)=0;\ h(x,y,z)=x-z;$
6. $g(x)=x;\ h(x,y,z)=x+z;$
7. $g(x)=x;\ h(x,y,z)=x-z;$
8. $g(x)=x;\ h(x,y,z)=y^2-z;$
9. $g(x)=x;\ h(x,y,z)=y^2+z;$

10. $g(x)=x;\ h(x,y,z)=x\cdot y+z;$

11. $g(x)=x;\ h(x,y,z)=x\cdot y-z;$

12. $g(x)=x;\ h(x,y,z)=z^{y+1};$

13. $g(x)=x;\ h(x,y,z)=x\cdot(y+1)\cdot z;$

14. $g(x)=x^2;\ h(x,y,z)=x^2+(y+1)^2+z;$

15. $g(x)=x;\ h(x,y,z)=x\cdot(y+2)+z;$

16. $g(x)=x;\ h(x,y,z)=x+y;$

17. $g(x)=x;\ h(x,y,z)=x+y+z;$

18. $g(x)=x;\ h(x,y,z)=$ ó$+z;$

19. $g(x)=x+2;\ h(x,y,z)=x+z;$

20. $g(x)=x^2;\ h(x,y,z)=x^2+y;$

21. $g(x)=x;\ h(x,y,z)=x\cdot z+y;$

22. $g(x)=x^x;\ h(x,y,z)=y+z;$

23. $g(x)=1;\ h(x,y,z)=x\cdot y\cdot z-y-z;$

24. $g(x)=x;\ h(x,y,z)=z^{y+1}+y;$

25. $g(x)=2x;\ h(x,y,z)=x\cdot(y-1)\cdot z;$

26. $g(x)=x^2;\ h(x,y,z)=x^2+(y+1)^2+z;$

27. $g(x)=x;\ h(x,y,z)=x\cdot(y+2)+z;$

28. $g(x)=x;\ h(x,y,z)=x+y;$

29. $g(x)=x;\ h(x,y,z)=x+y+\frac{x}{y};$

30. $g(x) = x^2$; $h(x, y, z) = -x - y - \frac{1}{z}$.

Какой аналитический вид имеет функция, являющаяся результатом минимизации данных функций:

1. $h(x, y, z) = x + y + z$;
2. $h(x, y, z) = x + y - z$;
3. $h(x, y, z) = x - y - z$;
4. $h(x, y, z) = x \cdot y + z$;
5. $h(x, y, z) = x \cdot y - z$;
6. $h(x, y, z) = x + y \cdot z$;
7. $h(x, y, z) = x - y \cdot z$;
8. $h(x, y, z) = x \cdot y^2 - z - 1$;
9. $h(x, y, z) = x \cdot y + x \cdot z - 1$;
10. $h(x, y, z) = x \cdot y + y \cdot z + z$;
11. $h(x, y, z) = x^2 + y - z$;
12. $h(x, y, z) = z^{x+y+1}$;
13. $h(x, y, z) = x^{x+y+z}$;
14. $h(x, y, z) = x^2 + (y+1)^2 + z$;
15. $h(x, y, z) = x \cdot (y+2) + z$;
16. $h(x, y) = -x + y$;
17. $h(x, y) = -x - y$;

18. $h(x,y)=x\cdot y+1;$

19. $h(x,y)=x+x\cdot y;$

20. $h(x,y)=x\ +y+1;$

21. $h(x,y)=x-y;$

22. $h(x,y)=y^x;$

23. $h(x,y)=x^y;$

24. $h(x,y)=\frac{x}{y}+1;$

25. $h(x,y)=x\cdot(y-1);$

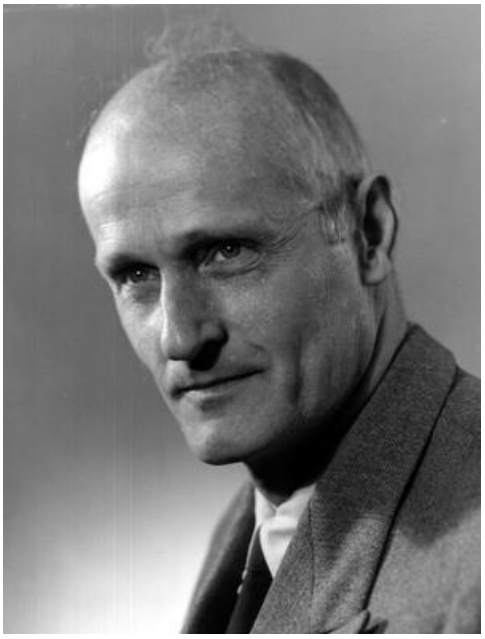
26. $h(x,y)=x^2+(y+1)^2;$

27. $h(x,y)=x\cdot(y+2);$

28. $h(x,y)=\frac{y}{x}-1;$

29. $h(x,y)=-x-y-\frac{x}{y};$

30. $h(x,y)=\log_x y.$



5.3. Стефен Коул Клини

Стефен Коул Клини (Stephen Cole Kleene 05.01.1909 – 25.01.1994 гг.) родился в 1909 г. в г. Хартфорде (Hartford), штат Коннектикут (Connecticut), но фактически его дом был на ферме своего деда по отцовской линии в г. Юнионе (Union) штат Мэн (Maine). В 1930 г. Клини окончил с отличием Амхерст-колледж (Amherst

College) и поступил в Принстонский (Princeton) университет в аспирантуру по математике.

В это время в Принстоне преподавателем математики был Освальд Веблен (Oswald Veblen), научным направлением которого на тот момент была логика и пути ее дальнейшего развития в связи с различными парадоксами теории множеств, которые в свою очередь фактически разрушили самую стройную из существующих аксиоматических теорий – теорию множеств. Его ученик Алонзо Черч (Alonzo Church), выпускник факультета Принстона, развивает идеи исследования тех функций, которые являются «вычислимые» и создает лямбда-исчисление – это теория, рассматривающая функции как правила, а не как графики и фактически выделяющая вычислительный аспект при введении функций. Клини и его однокурсник Баркли Россер (Barkley Rosser) присоединились к попытке Черча описать класс вычислимых функций.

В 1931 г. Курт Гедель доказал свою знаменитую теорему неполноты. В доказательстве Геделя используется как техническое средство класс функций, определяемых рекурсией. Данный класс называется рекурсивными функциями.

После прочтения работы Геделя и учитывая результаты Черча Клини приходит к мысли «… чтобы сделать λ-исчисление точным нам надо предварительно описать класс осмысленных выражений или функций, в связи с которыми мы будем его использовать. Будем достаточно смелыми и введем в рассмотрение некий минимальный язык – всего лишь с одним видом переменных – без типов и сортов, являющихся атомами, запас которых будем считать неограниченным. Мы получим язык из формализма Черча, удаляя остальные его элементы, которые для наших целей не понадобятся». Именно создание «минимального функционального языка» и стало основной целью Клини. Нельзя сказать, что теория рекурсивных функций в том виде, в каком она существует сейчас, как формализация понятия алгоритма создана одним Клини – нет, идея описать рекурсией некий вычислимый класс функций словно в воздухе висела и ею были увлечены многие математики того времени (Петер, Черч, Гедель, Аккерман, Сколем, Россер, Рассел, Эрбран и др.), но заслугой Клини несомненно велика. Особенно если учесть, что его

монография «Введение в метаматематику», 1957 г. – в середине прошлого века была единственной книгой переведенной на русский язык, в которой наиболее полно обрисована ситуация с конструктивными проблемами математики и на данной книге выросло целое поколение отечественных математиков. Не менее интересен и его учебник «Математическая логика» опубликованный в 1973 г. Именно переводы этих его трудов в советское время стали причиной не совсем верной, но навсегда прижившейся транслитерации его фамилии, ведь вернее было бы Клейни, с ударением на первую гласную.

В 1934 г. в Принстоне Клини защищает диссертацию «Теория натуральным чисел в формальной логике» («A Theory of Positive Integers in Formal Logic») под руководством Черча и в 1935 г. переезжает в университет г. Мэдисон штата Висконсин в качестве преподавателя, где в 1937 г. становиться доцентом, а в 1939 г. профессором. В 1941 г. женится на Нэнси Эллиот, которая впоследствии становиться матерью четырех его детей.

Под руководством Клини было написано 13 диссертаций. Среди его учеников были Джон Аддисон-младший, Клиффорд Спектор, Яннис Н. Московакис и Джоан Р. Московакис. Клини был избран в члены Национальной академии наук в 1969 г. К его исследованиям помимо рекурсивных функций добавляются и работы по теории автоматов. Его жена Нэнси умирает в 1970 г. и Клини жениться вновь лишь через 18 лет на Жанне Стейнмец. На пенсию Клини выходит в 1979 г. За четыре года до смерти Клини был награжден Национальной медалью науки.

Труды С.К. Клини, переведенные на русский язык:

1. Клини С.К. Введение в метаматематику. – М.: ИЛ, [1952]1957;

2. Клини С.К. Весли Р.И. Основания интуиционистской математики с точки зрения теории рекурсивных функций. – М.: Наука, [1965]1978;

3. Клини С.К. Математическая логика. – М.: Мир, [1967]1973;

4. Идельсон А.В, Минц Г.Е. Математическая теория логического вывода. – М.: Наука, 1967 – в сборнике переводов две статьи:

- Клини С.К. Перестановочность применений правил в генценовских исчислениях;

- Клини С.К. Конечная аксиоматизируемость теорий в исчислении предикатов с помощью дополнительных предикатных символов.

5.4. Курт Фридрих Гедель

Курт Гёдель **(**Kurt Friedrich Gödel 28.04.1906 – 14.01.1978 гг.) родился в Австро-Венгрии, в моравском городе Брно (в ту пору он назывался Брюнн). В 18 лет он поступил в Венский университет, где сначала изучал физику, но через два года переключился на математику. Известно, что такая смена научных интересов произошла во многом под влиянием книги Бертрана Рассела «Введение в философию математики». В 1930 г. защитил докторскую диссертацию по математике. В 1933–1938 гг. работал приват-доцентом Венского университета.

Диссертация Гёделя была посвящена проблеме полноты. В 1930 г. были получены некоторые результаты о полноте различных аксиоматических систем. Так, Гильберт построил искусственную систему, охватывающую часть арифметики, доказал ее полноту и непротиворечивость. Гёдель в своей диссертации доказал полноту исчисления предикатов первой ступени, и это дало надежду математикам на то, что им удастся доказать непротиворечивость и полноту всей математики. Однако уже в 1931 г. Гёдель доказал теорему о неполноте, нанесшую сокрушительный удар по этим надеждам. Статья под названием «Uber formal unentscheidbare Siitze der Principia Mathematica und verwandter Systeme» («О формально неразрешимых предложениях Principia Mathematica и родственных систем») была охарактеризована «…как одно из величайших достижений современной логики». Упомянутые в названии Principia Mathematica – это монументальный трехтомный трактат А.Н.

Уайтхеда и Б. Рассела, посвященный логике и основаниям математики [Э. Нагель, Дж. Ньюмен. Теорема Геделя. – М.:КРАСАНД, 2010].

Первая теорема Гёделя о неполноте:

В непротиворечивой теории первого порядка (в частности, во всякой непро тиворечивой теории, включающей формальную арифметику), существует такая замкнутая формула *F*, что ни *F*, ни $\bar{F}$ не являются выводимыми в этой теории.

Иначе говоря, в непротиворечивой теории существует утверждение, которое средствами самой теории невозможно ни доказать, ни опровергнуть. Например, такое утверждение можно добавить к системе аксиом, оставив её непротиворечивой.

Вторая теорема Гёделя о неполноте:

Во всякой непротиворечивой теории первого порядка (в частности, во всякой непротиворечивой теории, включающей формальную арифметику), формула *F*, утверждающая непротиворечивость этой теории, не является выводимой в ней. Иными словами, непротиворечивость теории не может быть доказана средствами этой теории. Однако вполне может оказаться, что непротиворечивость одной конкретной теории может быть установлена средствами другой, более мощной формальной теории. Но тогда встаёт вопрос о непротиворечивости этой второй теории, и т. д. Согласно этой теореме, любая процедура доказательства истинных утверждений элементарной теории чисел обречена на неполноту. Следовательно, внутренняя непротиворечивость любой математической теории не может быть доказана иначе, как с помощью обращения к другой теории, использующей более сильные допущения. Еще одним источником, оказавшим существенное влияние на формирование Гёделя как ученого, было его участие в работе «Венского кружка». Под этим именем в историю науки вошло собрание блестящих ученых - математиков, логиков, философов, которые регулярно собирались в Вене с конца 20-х и до середины 30-х гг. XIX века. В работе Венского кружка в разное время участвовали такие ученые, как Рудольф Карнап, Отто Нейрат, Герберт Фейгль, Мориц Шлик. С их деятельностью связывают становление философского позитивизма, но фактически тематика кружка охватывала осмысление общего места науч-

ного знания в познании природы и общества. Несколько международных конференций, организованных в разных европейских научных центрах, позволяют говорить о выдающейся роли, которую сыграл венский кружок в становлении фундаментального научного знания XX века. Курт Гёдель принимал участие практически во всех «четверговых» заседаниях кружка и в организованных им международных конференциях. Деятельность кружка в Австрии прервалась в 1936 г., когда его руководитель Мориц Шлик был убит студентом-нацистом на ступенях Венского университета. Большинство членов кружка эмигрировали в США. Туда же перебрался в 1940 г. и Курт Гёдель. Он проработал в Институте высших исследований (Institute for Advanced Study) в г. Принстоне штат Нью-Джерси, где с 1953 г. и до конца жизни являлся профессором.

О том, что логическое постижение мира занимало главное место в жизни ученого, говорит любопытная деталь его биографии. В 1948 г., когда решался вопрос о получении им американского гражданства, Гёдель должен был в соответствии с принятой процедурой сдать что-то вроде устного экзамена по азам американской конституции. Подойдя к вопросу со всей научной добросовестностью, он досконально изучил документ, и пришел к выводу, что в США законным путем, без нарушения конституции может быть установлена диктатура. Подобное открытие чуть не стоило ему провала на испытаниях, когда он вступил в дискуссию с принимавшим зачет чиновником, который, разумеется, считал основной закон своего государства величайшим достижением политической мысли. Друзья, среди которых был Альберт Эйнштейн, выступивший одним из двух поручителей Гёделя при получении им гражданства, уговорили его повременить с развертыванием своей аргументации хотя бы до принесения присяги. Позднее история получила любопытный эпилог: четверть века спустя другой американец, Кеннет Эрроу,

удостоился Нобелевской премии за доказательство в общем виде утверждения, к которому пришел Гёдель, изучив американскую конституцию [В мире науки, №3, 2007 г. «Теория противоречивости бытия» А. Музыкантский].

Гёдель внес важный вклад в теорию множеств. Два принципа – аксиома выбора и континуум-гипотеза – на протяжении десятилетий не поддавались доказательству, но интерес к ним не ослабевал: слишком привлекательны были их логические следствия. Гёдель доказал (1938 г.), что присоединение этих принципов к обычным аксиомам теории множеств не приводит к противоречию.

Обычно Гёделя считают австрийцем, но за свою жизнь он неоднократно менял гражданство. Рождённый подданным Австро-Венгрии, он в 12 лет принял гражданство Чехословакии после того, как Австро-Венгерская империя прекратила своё существование. В 23 года Гёдель стал гражданином Австрии, а в 32 года, после захвата Австрии Гитлером автоматически стал подданным германского Рейха, порядки которого были для него совершенно неприемлемы. В 1940 г. он уезжает в США, причём из-за опасности пути через Атлантику во время войны он едет через СССР и Японию.

К концу жизни у Геделя развилось психическое расстройство — параноидальный страх отравления. Он принимал пищу только из рук жены Адели, а после её смерти в 1977 г. отказался от пищи. Учёный скончался от недоедания 14 января 1978 г.

«Теорема Гёделя о неполноте является поистине уникальной. На нее ссылаются всякий раз, когда хотят доказать «всё на свете» – от наличия богов до отсутствия разума», – пишет выдающийся современный математик В. А. Успенский.

5.5. Алонзо Черч

Род Алонзо насчитывает четырех Алонзо Черч: сам Алонзо (Church Alonzo 14.06.1903 – 11.08.1995 гг.), прадед самого великого логика, его дядя и сын.

Его отец, Самуил Роббинс Черч, был сотрудником юстиции Муниципальный суда округа Колумбия, отсутствии зрения и слуха заставило его отказаться от этой должности и семья переехала в штат Виржинию. Инцидент в средней школе при использовании пневматического пистолета привел Черча к тому, что он ослеп на левый глаз. Окончив подготовительную школу в г. Коннектикуте в 1920 г., Черч поступил в Принстонский университет (Princeton University). Некоторое время Черч работал неполный рабочий день в столовом зале, чтобы помочь оплатить свое обучение. Первый опубликованный труд Черча, посвященный единственности преобразования Лоренца, был написан в то время, когда он был студентом бакалавриата. Целью данной работы являлось получение наборов логически независимых постулатов, однозначно определяющих преобразования Лоренца в одном измерении.

Черч окончил обучение с отличием в 1924 г. и завершил кандидатскую диссертацию «Альтернативы предположения Цермело» («Alternatives to Zermelo's Assumption») в течение трех лет (в 1927 г.) под руководством Освальда Веблена (Oswald Veblen). Будучи аспирант, он женился в 1925 г. на Марии Джулии Кучински (Kuczinski), которая обучалась на медсестру, и это несмотря на то, что его семья и коллеги считали, что он останется холостяком еще долго из-за своего «красивого лица».

История знакомства Черча с будущей супругой романтична и трагична: летом 1924 года, Черч вышел на пресечение улиц и серьезно пострадал от трамвая, который шел с левой (невидимой Черчу, вследствие его слепоты) стороны, Мария была медсестрой-стажером в больнице, в которую он впоследствии попал на лечение. Они были неразлучны следующий 51 год, до самой смерти Марии. Мария родила троих детей: Алонзо Черч-младший, родился в 1929 г., Мэри Энн в 1933 г., и Милдред в 1938 г. (Мэри Энн позже вышла замуж за логика Джона Эддисона).

Чёрч стал доцентом математики в Принстонском университете в 1929 году, профессором в 1947 г. и оставался им до 1967 г., а с 1967 г. – профессор математики и философии Калифорнийского университета (Лос-Анджелес).

Чёрч прославился разработкой теории лямбда-исчисления, последовавшей за его знаменитой статьёй 1936 г., в которой он показал существование т.н. «неразрешимых задач». Эта статья предшествовала исследованию Алана Тьюринга на тему «проблемы зависания», в котором также было продемонстрировано существование задач, неразрешимых механическими способами. Впоследствии Чёрч и Тьюринг показали, что лямбда-исчисления и машина Тьюринга имели одинаковые свойства, таким образом доказывая, что различные «механические процессы вычислений» могли иметь одинаковые возможности. Эта работа была оформлена как тезис Чёрча-Тьюринга, в самой общей форме он гласит, что любая вычислимая функция является частично рекурсивной, или, что тоже самое, может быть вычислена некоторой машиной Тьюринга.

Тезис Чёрча - Тьюринга невозможно строго доказать или опровергнуть, поскольку он устанавливает «равенство» между строго формализованным понятием частично рекурсивной функции и неформальным понятием «вычислимой функции».

Основная гипотеза же теории вычислимых функций (т. н. тезис Черча) звучит так: каждая вычислимая функция является частично рекурсивной.

В 1936-1938 гг. Алан Тьюринг писал диссертацию под руководством Алонзо Чёрча (всего защищенных под его руководством 34 работы). В этот же период была опубликована знаменитая статья Тьюринга «О вычислимости разрешимых проблем» («On Computable Numbers with an Application to the Entscheidungs problem»), которая включала в себя концепции логического проектирования универсальной машины. Фон Нейман, несомненно, был знаком с идеями Тьюринга, однако неизвестно, применял ли он их в проектировании IAS-машины десять лет спустя.

В 1966 г. Черч доказал, что проблема разрешимости для исчисления предикатов неразрешима. Эти результаты оказали большое влияние на развитие мате-

матической логики. Сделал существенный вклад в развитие комбинаторной логики; ему принадлежат исследования в области логической семантики и модальной логики.

Чёрч оставался профессором математики в Принстоне до 1967 г., после чего он переехал в Калифорнию. Помимо прочего, его система лямбда-исчисления легла в основу функциональных языков программирования.

В 1950 г. профессор Массачусетского технического университета Джон Маккарти (John McCarthy), также выпускник Принстона, заинтересовался работами Чёрча, и в 1958 г. он представил язык программирования Лисп (LISP, List Processing Language). Лисп был реализацией лямбда-исчисления для архитектуры фон Неймана. Не секрет, что множество учёных и программистов отмечают большую выразительную мощь Лиспа. В 1973 г. группа программистов из Лаборатории Искусственного Интеллекта при Массачусетском техническом университете разработали процессор, который они назвали «Лисп-машина» – настоящее «железное» воплощение лямбда-исчисления Чёрча.

Черч умер в г. Гудзоне, штат Огайо (где жил его сын), 11 августа 1995 г., он был похоронен на Принстонском кладбище, где были похоронены его жена и его родители.

Глава 6. Алгоритмически неразрешимые проблемы

6.1. Общие вопросы

— Г-голубчики, – сказал Фёдор Симеонович озадаченно...
— Это же проблема Бен Б-бецалеля.
— К-калиостро же доказал, что она н-не имеет р-решения.
— Мы сами знаем, что она не имеет решения,
— сказал Хунта, немедленно ощетиниваясь.
— Мы хотим знать, как её решать.
— К-как-то ты странно рассуждаешь, К-кристо...
— К-как же искать решение, к-когда его нет?
— Б-бессмыслица какая-то...
— Извини, Теодор, но это ты очень странно рассуждаешь.
— Бессмыслица – искать решение, если оно и так есть.
— Речь идёт о том, как поступать с задачей, которая решения не имеет.
— Это глубоко принципиальный вопрос...
А. Стругацкий, Б. Стругацкий
«Понедельник начинается в субботу»

Алгоритмически неразрешимой называется такая задача, для которой доказана невозможность построения алгоритма ее решения в некой формализации алгоритма.

Заметим, что всякий алгоритм представляет собой способ решения некоторой массовой проблемы, формулируемой в виде переработки не одного, а целого множества входных слов в соответствующие им выходные слова. Т.е. всякий алгоритм можно рассматривать как некоторое универсальное средство для решения целого класса задач. Однако алгоритмическая неразрешимость того или иного класса вовсе не означает невозможности решения любой конкретной задачи из этого класса. Речь идёт лишь о невозможности решения всех задач данного класса одним и тем же приёмом.

Алгоритмическую неразрешимость нельзя распространять на случаи, когда существование алгоритма не доказано. Например, задача алгоритмической разрешимости Великой теоремы Ферма формулируется так: «Существует ли алгоритм, который по заданному натуральному n определяет, имеет ли уравнение $x^n + y^n = z^n$ решение в целых числах?». Ответ на этот вопрос, как и на вопрос о справедливости теоремы Ферма при $n > 2$, неизвестен. Для $n = 2$ мы знаем, что $3^2 + 4^2 = 5^2$. Это значит, что нельзя говорить об алгоритмической неразрешимости этой задачи.

6.2. История вопроса

Практически все алгоритмически неразрешимые проблемы (АНП) можно сформулировать так: «Построить алгоритм, с помощью которого можно определить для каждого элемента некоторого данного множества, обладает ли некий элемент тем или иным свойством».

Также полезно разделить АНП на те, которые возникли при формализации понятия алгоритма, как, например, проблемы самоприменимости и останова машин Тьюринга, и те, которые возникли внутри неких математических исследований, как 10-я проблема Гильберта.

Практически все АНП взаимосвязаны или эквивалентны и поэтому, чтобы показать алгоритмическую неразрешимость некой новой проблемы, чаще всего в доказательстве упираются на уже известную АНП.

Кстати, сама десятая проблема была решена в 1970 г. Ю.В. Матиясевичем, который показал невозможность построения такого алгоритма, что сделало данную проблему алгоритмически неразрешимой. История же получила продолжение и породила еще две проблемы:

- существует ли у алгебраического уравнения с целыми коэффициентами решение в действительных числах;

- существует ли у алгебраического уравнения с целыми коэффициентами решение в рациональных числах.

Первую из этих проблем решил А. Тарский, им был построен алгоритм поиска таких корней, а вот вторая – не решена до сих пор.

6.3. Проблема нумерации машин Тьюринга и пример невычислимой функции

Удивительно, но созданная для выхода из кризиса теория машин Тьюринга породила и первые примеры алгоритмически неразрешимых проблем – проблемы нумерации самих машин, проблемы останова и применимости. Уточним: проблема Гильберта была сформулирована в 1900 г., решена же – лишь в 1970 г. Первый же пример чисто математической (не связанной с уточнением алгоритма) алгоритмически неразрешимой проблемы – проблема Туэ о равенстве слов в полугруппе сформулирована в 1914 г. и доказана в 1946-1947 гг. А.А.Марковым и Э.Л.Постом. Нами же будет рассмотрена проблема нумераций машин.

Опишем процесс нумерации всех машин Тьюринга, который используем при построении примера невычислимой по Тьюрингу функции. Будем считать, что для обозначения внутренних состояний машин Тьюринга используются буквы бесконечной последовательности: $q_0, q_1, q_2, ..., q_r, ...$, а для обозначения букв внешних алфавитов используются буквы последовательности: $a_0, a_1, a_2, ..., a_s, ...$

Выразим (или, как говорят, закодируем) все символы этих бесконечных последовательностей словами конечного стандартного алфавита $\{a_0, 1, q, С, П, Л\}$ по следующим правилам:

q_i ($i = 0, 1, ...$) обозначим (закодируем) словом из $i + 1$ букв q: $qq\ldots q$;

a_j ($j = 1, 2, ...$) обозначим (закодируем) словом из j единиц: 11...1.

В стандартном алфавите программу машины Тьюринга можно записать в виде слова, руководствуясь следующим правилом. Сначала все команды программы переводятся на язык стандартного алфавита, для чего в записях этих $q_i a_i \to q_1 a_m$, где $X \in \{С, П, Л\}$, опускается символ «→», а буквы q_i, a_j, q_1, a_m заменяются соответствующими словами стандартного алфавита. Затем полученные слова-команды записываются подряд в любом порядке в виде единого слова.

Нетрудно указать алгоритм, позволяющий узнавать, является ли слово в стандартном алфавите программой некоторой машины Тьюринга. Такой алгоритм может, например, состоять в следующем. Нужно анализировать все полслова данного слова, заключенные между всевозможными парами букв из {С, П, Л}. Эти подслова должны иметь следующую структуру: сначала записаны несколько букв q затем a_0 или несколько букв 1, затем снова несколько букв q, наконец, снова a_0 или несколько единиц.

Таким образом, каждая машина Тьюринга вполне определяется некоторым конечным словом в конечном стандартном алфавите. Поскольку множество всех конечных слов в конечном алфавите счетно, то и всех мыслимых машин Тьюринга (отличающихся друг от друга по существу своей работы) имеется не более чем счетное множество.

Перенумеруем теперь все машины Тьюринга, для чего все слова стандартного алфавита, представляющие собой программы всевозможных машин Тьюринга, расположим в виде фиксированной счетно-бесконечной последовательности, которую составим по такому правилу: сначала выписываются в какой-нибудь фиксированной последовательности все

однобуквенные слова: $\alpha_0, \alpha_1, ..., \alpha_\xi, ...,$ представляющие программы машин Тьюринга (множество таких слов конечно, потому что конечен стандартный алфавит, из букв которого строятся слова), затем выписываются все двухбуквенные слова $\alpha_{\xi+1}, ..., \alpha_n,$ представляющие программы машин Тьюринга (множество таких слов также конечно, потому что конечен стандартный алфавит), затем выписываются трехбуквенные слова и т.д. Получится последовательность программ $\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n, ...,$ всех машин Тьюринга. Число k будем называть ***номером машины Тьюринга***, если программа этой машины записывается словом α_k. Покажем существование невычислимых по Тьюрингу функций.

Теорема. Существует функция, невычислимая по Тьюрингу, т. е. не вычислимая ни на одной машине Тьюринга.

Доказательство. Функции, о которых идет речь, представляют собой функции, заданные и принимающие значения в множестве слов в алфавите $A_1 = \{1\}$. Ясно, что множество слов в алфавите $A_1 = \{1\}$ счетно. Следовательно, рассматривается множество всех функций, заданных на счетном множестве и принимающих значения в счетном же множестве. Как известно, это множество имеет мощность континуума. С другой стороны, поскольку множество всевозможных машин Тьюринга, как мы установили выше, перенумеровав их, счетно, это и множество функций, вычислимых по Тьюрингу, также счетно. Континуальная мощность строго больше счетной. Следовательно, существуют функции, невычислимые по Тьюрингу.

Доказанная теорема есть теорема существования. Интересно получить пример конкретной функции, невычислимой по Тьюрингу.

Укажем конкретную функцию, которую нельзя вычислить ни на какой машине Тьюринга. На основании тезиса Тьюринга это будет означать, что не существует вообще никакого алгоритма для вычисления значений такой функции.

Рассмотрим следующую функцию $\Psi(\alpha)$ на словах в алфавите $A_1 = \{1\}$. Для произвольного слова α длиной n в алфавите $A_1 = \{1\}$ положим: $\Psi(\alpha) = \beta_n 1$, если слово а перерабатывается машиной Тьюринга с номером n (см. предыдущий

пункт) в слово β_n алфавита $A_1 = \{1\}$; $\Psi(\alpha) = 1$ в противном случае. Докажем, что функция $\Psi(\alpha)$ невычислима по Тьюрингу.

Допустим противное. Это означает, что существует машина Тьюринга Т со стандартным алфавитом $\{a_0, 1, q, С, П, Л\}$, вычисляющая эту функцию. Пусть k — номер этой машины в нумерации, описанной в предыдущем пункте. Посмотрим, чему равно слово $\Psi(1^k)$ (напомним, что 1^k = 11...1 — слово из k единиц), являющееся значением функции $\Psi(\alpha)$ при $\alpha = 1^k$. Предположим, что машина Т перерабатывает слово 1^k в слово β^k в том же алфавите $A_1 = \{1\}$. Тогда по определению вычислимости функции $\Psi(\alpha)$ на машине Т это означает, что $\Psi(1^k) = \beta^k$. Но с другой стороны, по самому определению функции это означает, что $\Psi(1^k) = \beta^k 1$. Полученное противоречие доказывает, что машины Тьюринга, вычисляющей функцию $\Psi(\alpha)$, не существует.

Принимая во внимание тезис Тьюринга, заключаем, что не существует вообще никакого алгоритма для вычисления значений функции $\Psi(\alpha)$. Это означает, что проблема нахождения значений функции $\Psi(\alpha)$ для всевозможных значений аргумента алгоритмически неразрешима.

Рассмотрим еще одну алгоритмическую проблему и докажем ее неразрешимость. Это проблема определения общерекурсивности алгоритмов (и функций), т.е. проблема определения того, ко всяким ли допустимым начальным данным применим алгоритм. Прежде докажем лемму.

Лемма. Для любого перечисления любого множества Φ всюду определенных вычислимых (т. е. общерекурсивных) функций существует общерекурсивная функция, не входящая в это перечисление.

Доказательство. Пусть y — перечисление множества Φ, порождающее последовательность A_0, A_1, A_2, ... всюду определенных вычислимых функций. Введем функцию $B(x) = A_x(x) + 1$. Так как A_x, общерекурсивна, то и $B(x)$ общерекурсивна. Если предположить, что $B \in A$, то B имеет некоторый номер x_0 и, следовательно, $B(x) = A_{x_0}(x)$. Тогда в точке x = x_0 по определению $B(x_0) = A_{x_0}(x_0) + 1$, а в силу последнего соотношения имеем $B(x_0) = A_{x_0}(x_0)$.

Получаем противоречие, из которого следует, что B не входит в перечисление, порождаемое y.

Заметим, что если бы в перечислении допускались частичные функции, то такое определение функции B не привело бы к противоречию, а означало бы лишь, что B не определена в точке x_0).

Теорема. Проблема определения общерекурсивности алгоритмов неразрешима, т. е. не существует алгоритма $B(x)$, такого, что для любого алгоритма A_x

$$B(x) = \begin{cases} 1, \text{ если } A_x \text{ всюду определен,} \\ 0, \text{ если } A_x \text{ не всюду определен.} \end{cases}$$

Доказательство. Допустим противное, т. е. такой алгоритм $B(x)$ существует. Тогда он определяет общерекурсивную функцию $f(x)$. Определим функцию $g(x)$ следующим образом:

$$g(x)=\mu y\,[f(y)=1]$$

$$g(x+1)=\mu y\,[y > g(x) \wedge f(y)=1]$$

Так как номеров всюду определенных функций (и, следовательно, точек y, в которых $f(y) = 1$) бесконечное множество, то функция $g(x)$ всюду определена. Ясно, что функция $g(x)$ принимает значение 1 на каждой всюду определенной вычислимой (т.е. общерекурсивной) функции, т.е. является перечислением множества всех общерекурсивных функций. Но, на основании предыдущей леммы, такого перечисления не существует. Противоречие. Следовательно, не существует и алгоритма $B(x)$, определенного в условии теоремы, т.е. проблема определения общерекурсивности алгоритмов неразрешима.

Раз нельзя указать единого алгоритма, отличающего всюду определенные вычислимые (т.е. общерекурсивные) функции от частично рекурсивных, попытаемся подойти к проблеме частично рекурсивных функций с другой стороны. Может быть, возможно *каждую* частично рекурсивную функцию доопределить на неопределимых точках так, чтобы получилась рекурсивная (т.е. общерекурсивная) функция. Оказывается, и эта задача неразрешима, что следует из теоремы, приведенной ниже.

Теорема. Существует такая частично рекурсивная функция f, что никакая общерекурсивная функция g не является ее доопределением.

Доказательство. Как и прежде, считаем, что выбрана некоторая вычислимая нумерация φ: $N \to Al^*$ и для каждого алгоритма A_x значение $\varphi^{-1}(A_x) = x$ - его номер в этой нумерации. Рассмотрим следующую частичную функцию:

$$f(x) = \begin{cases} A_x(x) + 1, \text{ если } A_x \text{ всюду определен,} \\ \text{не определена, если } A_x \text{ не всюду определен.} \end{cases}$$

Ясно, что функция $f(x)$ вычислима и, значит, частично рекурсивна.

Пусть теперь $g(x)$ - произвольная общерекурсивная функция и x_g - ее номер в нумерации φ, т.е. $\varphi^{-1}(g) = x_g$. Так как g всюду определена, то $g(x_g) = A_{x_g}(x_g)$ определена и, следовательно, $f(x_g) = g(x_g) + 1$. Таким образом, для любой общерекурсивной функции g имеем $f(x_g) \neq g(x_g)$. Это и означает, что $f \neq g$. Теорема доказана.

Следовательно, существуют частичные алгоритмы, которые нельзя доопределить до всюду определенных алгоритмов.

Рассмотренные выше теоремы и алгоритмически неразрешимые проблемы из них вытекающие носят довольно экзотический характер: все они были так или иначе связаны с самоприменимостью алгоритма, когда алгоритм работает с собственным описанием (находится значение вычислимой функции f_x в точке, являющейся ее собственным номером в выбранной нумерации вычислимых функций: $f(x)$). Обратим внимание на теорему Райса, которая опираясь на неразрешимость вышерассмотренной проблемы нумераций, устанавливает алгоритмическую неразрешимость вообще всякого нетривиального свойства вычислимых функций.

По-прежнему имеется некоторая нумерация алгоритмов φ: $N \to Al^*$, в которой каждый алгоритм A_x имеет номер $\varphi^{-1}(A_x) = x$. Этот же номер имеет и функция f_x, которую вычисляет алгоритм A_x.

Теорема Райса. Пусть C - любой непустой собственный класс вычислимых функций от одного аргумента (существуют как функции, принадлежащие C, так и

вычислимые функции, не принадлежащие C). Тогда не существует алгоритма, который бы по номеру x функции f_x определял бы, принадлежит f_x классу C или нет. Иначе говоря, множество $\{x: f_x \in C\}$ неразрешимо.

Доказательство. Допустим противное, т. е. множество $\{x: f_x \in C\}$ разрешимо. Тогда оно обладает вычислимой характеристической функцией:

$$\chi_M(x) = \begin{cases} 1, \text{ если } x \in M, \text{ т.е. } f_x \in C, \\ 0, \text{ если } x \notin M. \text{ т.е. } f_x \notin C. \end{cases}$$

Пусть f_0 нигде не определенная функция. Рассмотрим сначала случай, когда $f_0 \notin C$. Выберем тогда какую-нибудь конкретную вычислимую функцию $f_a \in C$ и определим функцию $F(x, y)$:

$$F(x, y) = \begin{cases} f_a(y), \text{ если значение } f_x(x) \text{ определено}, \\ f_0(y), \text{ если значение } f_x(x) \text{ не определено}. \end{cases}$$

Функция $F(x, y)$ вычислима. Для ее вычисления надо вычислять $f_x(x)$: если $f_x(x)$ определена, то этот процесс когда-нибудь остановится и нужно будет перейти к вычислению $f_a(y)$, если же $f(x)$ не определена, то процесс не остановится, что равносильно вычислению функции $f_0(y)$.

Зафиксируем в функции $F(x, y)$ аргумент x. Тогда F станет вычислимой функцией от одного аргумента y. Номер этой функции в единой нумерации φ вычислимых функций зависит от значения x т.е. является всюду определенной функцией $g(x)$. Ясно, что функция $g(x)$ вычислима, так как является суперпозицией двух вычислимых функций: $g(x) = \varphi^{-1}(F(x, y))$. Таким образом, $F(x, y) = f_{g(x)}$ и, значит,

$$f_{g(x)}(y) = \begin{cases} f_a(y), \text{ если } f_x(x) \text{ определена}, \\ f_0(y), \text{ если } f_x(x) \text{ не определена}. \end{cases}$$

Отсюда ясно, что $f_{g(x)} \in C$ тогда и только тогда, когда $f_{g(x)} = f_a$ (так как $f_a \in C$), т.е. $f_x(x)$ определена. В случае же, когда $f_x(x)$ не определена, мы имеем $f_{g(x)} = f_0$ и, следовательно, $f_{g(x)} \notin C$, так как $f_0 \notin C$. Другими словами, $g(x) \in M$ тогда и только тогда, когда $f_x(x)$ определена. Это означает, что характеристическая функция χ_M на аргументах $g(x)$ имеет вид:

$$\chi_M(g(x)) = \begin{cases} 1, \text{ если } f_x(x) \text{ всюду определена,} \\ 0, \text{ если } f_x(x) \text{ не всюду определена.} \end{cases}$$

Последнее означает следующее. Поскольку функции χ_M и $g(x)$ вычислимы, то мы для каждого номера x можем выяснить, определено значение $f_x(x)$ или нет. Это, в свою очередь, означает, что построена разрешающая функция $\chi_M \circ g$ для проблемы самоприменимости алгоритма, что невозможно.

Если рассмотреть случай, когда $f_0 \in C$, то нужно выбрать $f_a \notin C$. Проведя аналогичные рассуждения, придем к следующей разрешающей функции для проблемы самоприменимости 1 - $\chi_M(g(x))$, что также противоречит доказанной ранее неразрешимости этой алгоритмической проблемы. Теорема доказана.

Теорема Райса означает, что не существует единого алгоритма, который для каждой вычислимой функции (по ее номеру) определял бы, обладает эта функция тем или иным свойством или нет, например, является ли эта функция постоянной, монотонной, периодической, ограниченной и т.п., что делает ее обобщенным аналогом и проблемы самоприменимости (то есть, что невозможно по номеру машины определить является ли она самоприменимой) и иных алгоритмически неразрешимых проблем. Данный факт позволяет выдвинуть гипотезу, что все алгоритмически неразрешимые проблемы равносильны одной: не существует алгоритма, который по данному элементу некоторого данного множества, определяет обладает ли этот элемент неким свойством. Стоит добавить, что и множество и то самое свойство должны обладать некоторыми требованиями. Во-первых, множество не должно быть конечным, иначе перебрав все элементы можно их проверить на наличие любого свойства. Во-вторых, свойство должно быть самопредикативным или, иначе говоря, самоссылочным.

Материалы для самостоятельной работы

Итоговая контрольная работа №1 «Машины Тьюринга и машины Поста»

Вариант №1

1. Дано число n в восьмеричной системе счисления. Постройте машину Тьюринга, которая бы увеличивала данное число на единицу.

2. Построить машину Тьюринга, вычисляющую функцию $f(x,y)$, равную остатку от деления x на 3.
3. На ленте машины Поста находятся n массивов меток. Каретка находится где-то над первым массивом. Удалите все массивы с четными номерами (соседние массивы разделены тремя пустыми секциями).

Вариант № 2

1. Дана десятичная запись натурального числа $n > 1$. Постройте машину Тьюринга, которая уменьшала бы данное число на 1. При этом запись числа $n - 1$ не должна содержать левый нуль. Например, 100 – 1 = 99, а не 099. Начальное положение головки – правое.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y) = x + y$.
3. Найти НОД двух чисел, находящихся на ленте машины Поста. Между этими числами находится произвольное количество пустых секций. Каретка находится над левой меткой левого числа.

Вариант № 3

1. Дан массив из открывающихся и закрывающихся скобок. Построить машину Тьюринга, которая удаляла бы пары взаимных скобок. Например, дано: «) (( ) (( )», надо получить: «)… ((.».
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y) = x \div 2$.
3. Составьте программу сложения произвольного количества чисел, записанных на ленте машины Поста через одну пустую секцию. Каретка обозревает крайнюю левую секцию левого числа.

Вариант № 4

1. Дана строка из букв a и b. Построить машину Тьюринга, которая переместит все буквы a в левую часть строки, а все буквы b – в правую. Считывающая головка находится над крайним левым символом строки.
2. Построить машину Тьюринга, вычисляющую функцию $f(n) = n + 2$.

3. Дан массив меток. Каретка обозревает первую пустую секцию перед началом массива. Составьте машину Поста, которая раздвигает массив так, чтобы после каждой метки была пустая секция.

Вариант № 5

1. Даны два целых положительных числа в десятичной системе счисления. Построить машину Тьюринга, которая будет находить разность этих чисел, если известно, что первое число больше второго, а между ними стоит знак «-». Ссчитывающая головка находится над левой крайней цифрой левого числа.
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z) = x + 1$.
3. На ленте машины Поста отмечен массив n меток. Найдите число $2n+1$ и проверьте, делится ли оно на 3. Если да, то после числа через одну пустую секцию поставьте две метки, если нет – поставьте три метки. Каретка находится над крайней левой отмеченной секцией.

Вариант № 6

1. Даны два целых положительных числа в различных системах счисления, одно – в троичной системе счисления, другое – в десятичной. Постройте машину Тьюринга, которая будет находить сумму этих чисел в десятичной системе счисления.
2. Построить машину Тьюринга, вычисляющую функцию $f(x) = x + 1$.
3. На ленту машины Поста нанесены два массива меток на некотором расстоянии друг от друга. Соедините эти два массива в один. Каретка находится над крайней левой меткой левого массива.

Вариант № 7

1. Постройте машину Тьюринга, которая преобразует запись числа из двоичной системы счисления в восьмеричную.
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z)=x$.
3. На ленте машины Поста изображено n массивов меток, отделенных друг от друга одной свободной ячейкой. Каретка находится над первой меткой первого массива. Определите количество массивов.

Вариант № 8

1. Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Построить машину Тьюринга, которая будет записывать в десятичной системе счисления число этих меток.
2. Построить машину Тьюринга, вычисляющую функцию $f(x)=\begin{cases}1, & \text{если } x>0;\\ 2, & \text{если } x=0.\end{cases}$
3. На информационной ленте машины Поста находится массив меток. Каретка находится где-то над массивом (но не над крайней меткой). Сотрите все метки, кроме крайних, таким образом, чтобы положение каретки при этом не изменилось.

Вариант № 9

1. На ленте машины Тьюринга в трёх клетках записаны три различные буквы: *A*, *B*, *C*. Считывающая головка в начальном состоянии обозревает букву, расположенную справа. Построить машину Тьюринга, которая поменяет местами крайние буквы.
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y, z)=0$.
3. На ленте машины Поста находится n массивов меток, после последнего массива на расстоянии более трех пустых секций находится одна метка. Массивы разделены тремя пустыми ячейками. Количество меток в массивах не может быть меньше двух. Произвести обработку массивов следующим образом: если количество меток в массиве кратно трем, то стереть метки в данном массиве через одну, иначе – массив стереть полностью. Каретка находится над крайней левой меткой первого массива.

Вариант № 10

1. На ленте машины Тьюринга записано число в десятичной системе счисления. Построить машину Тьюринга, которая бы умножала данное число на 2. В начале работы считывающая головка находится над крайней левой цифрой числа.
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y)=x+2$.

3. Задача В.А.Успенского. На информационной ленте либо вправо, либо влево от секции, над которой расположена каретка, находится массив меток. Расстояние до массива выражается произвольным числом. Необходимо составить программу, работая по которой машина Поста найдет этот массив и установит каретку на начало этого массива.

Вариант № 11

1. На ленте машины Тьюринга записано целое положительное число в десятичной системе счисления. Найти произведение этого числа на 11, если считывающая головка обозревает крайнюю правую цифру числа.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y)$, равную остатку от деления x на 2.
3. На информационной ленте машины Поста находятся два массива в M и N меток. Составьте программу выяснения, одинаковы ли массивы по длине.

Вариант № 12

1. На ленте машины Тьюринга записано слово, состоящее из букв латинского алфавита $\{a, b, c, d\}$. Подсчитать число вхождений буквы a в заданное слово. Полученное значение записать на ленту левее заданного слова через пробел. Считывающая головка обозревает крайнюю левую букву.
2. Построить машину Тьюринга, вычисляющую функцию $f(x) = \begin{cases} 0, & \text{если } x = 1; \\ 1, & \text{если } x \neq 1. \end{cases}$.
3. На ленте машины Поста расположены два массива. Составьте программу стирания того из массивов, который имеет большее количество меток.

Вариант № 13

1. На ленте машины Тьюринга записано десятичное число. Определить, делится ли это число не 5 без остатка. Если делится, то справа от числа записать слово «да», если не делится, то записать «нет». Считывающая головка находится где-то над числом.

2. Построить машину Тьюринга, вычисляющую функцию

$$f(k)=\begin{cases}k, & \text{если } k \text{ - нечетное;}\\ 1, & \text{если } k \text{ - четное.}\end{cases}$$

3. На ленте машины Поста расположен массив из $2N$-1 меток. Составьте программу отыскания средней метки массива и стирания ее.

Вариант № 14

1. На ленте машины Тьюринга записано число в десятичной системе счисления. Считывающая головка находится над правой цифрой. Запишите цифры этого числа в обратном порядке.
2. Построить машину Тьюринга, вычисляющую функцию

$$f(x)=\begin{cases}0, & \text{если } x=1;\\ 1, & \text{если } x\neq 1.\end{cases}$$

3. На ленте машины Поста расположен массив из $2N$ отмеченных секций. Составьте программу, по которой машина Поста раздвинет на расстояние в одну секцию две половины данного массива.

Вариант № 15

1. На ленте машины Тьюринга находится массив, состоящий только из символов *A* и *B*. Сожмите массив, удавив из него все элементы *B*.
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y)=2$.
3. Составьте программу нахождения разности двух неотрицательных целых чисел *a* и *b*, находящихся на ленте машины Поста. Каретка находится над левой меткой левого числа. Неизвестно, какое число больше: *a* или *b*.

Вариант № 16

1. Число *n* записано на ленте машины Тьюринга массивом меток. Преобразуйте это значение n по формуле:

$$\begin{cases}n-2, n>5\\ \varphi(n)=1, n=5\\ 2n, n<5\end{cases}$$

Считывающая головка обозревает крайнюю левую метку.

2. Построить машину Тьюринга, вычисляющую функцию $f(x)=x+3$.

3. Число k представлено на ленте машины Поста $k+1$ идущими подряд метками. Найдите остаток от деления целого неотрицательного числа k на 3, если известно, что каретка находится справа от заданного числа.

Вариант № 17

1. На ленте машины Тьюринга массив из $2n$ меток. Уменьшите этот массив в 2 раза.
2. Построить машину Тьюринга, вычисляющую функцию $f(x)=4$.
3. Число k представлено на ленте машины Поста $k+1$ идущими подряд метками. Найдите остаток от деления целого неотрицательного числа k на 3, если известно, что каретка находится справа от заданного числа.

Вариант № 18

1. Даны два натуральных числа m и n, записанные в унарной системе счисления. Между этими числами стоит знак «?». Выясните отношение между заданными числами и замените знак «?» на один из подходящих знаков «<», «>» или «=».
2. Построить машину Тьюринга, вычисляющую функцию $f(x, y)=2$.
3. На ленте машины Поста расположен массив из N меток. Составьте программу, действуя по которой машина выяснит, делится ли число на 3. Если да, то после массива через одну пустую секцию поставьте метку V.

Вариант № 19

1. Найдите произведение двух натуральных чисел m и n, записанных в унарной системе счисления. Соответствующие наборы символов «|» разделены знаком «*», а справа от последнего символа стоит знак «=». Поместите результат умножение этих чисел вслед за знаком «=».
2. Построить машину Тьюринга, вычисляющую функцию
$$f(x)=\begin{cases}2, & \text{если } x=0;\\ 0, & \text{если } x\neq 0.\end{cases}$$
3. Составьте программу сложения произвольного количества целых неотрицательных чисел, записанных на ленте машины Поста на расстоянии

одной пустой секции друг от друга. Каретка находится над крайней левой меткой левого числа.

Вариант № 20

1. На ленте машины Тьюринга в трёх клетках записаны три различные буквы: *A, B, C*. Считывающая головка в начальном состоянии обозревает букву, расположенную справа. Построить машину Тьюринга, которая поменяет местами крайние буквы.
2. Построить машину Тьюринга, вычисляющую функцию $f(x)=\begin{cases}0, & \text{если } x>0;\\ 1, & \text{если } x=0.\end{cases}$
3. Составьте программу сложения двух целых неотрицательных чисел *a* и *b*, расположенных на ленте машины Поста. Каретка расположена над одной из меток, принадлежащих числу *a*. Число *b* находится правее числа *a* через несколько пустых секций.

Вариант № 21

1. На ленте машины Тьюринга записано число в десятичной системе счисления. Построить машину Тьюринга, которая бы умножала данное число на 2. В начале работы считывающая головка находится над крайней левой цифрой числа.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y)=x+1$.
3. Число *k* представляется на ленте машины Поста $k+1$ идущими подряд метками. Одна метка соответствует нулю. Составьте программу прибавления 1 к произвольному числу *k*. Каретка расположена над одной из меток, принадлежащих заданному числу *k*.

Вариант № 22

1. На ленте машины Тьюринга записано число в двоичной системе счисления. Найти произведение этого числа на себя, если считывающая головка обозревает крайнюю правую цифру числа.

2. Построить машину Тьюринга, вычисляющую функцию $f(x)=\begin{cases} 0, & \text{если } x=0; \\ x-1, & \text{если } x>0. \end{cases}$
3. Игра Баше. В игре участвуют двое (человек и машина Поста). Напишите программу, по которой всегда будет выигрывать машина Поста. Суть игры заключается в следующем: имеется 21 предмет. Первым ходит человек. Каждый из играющих может брать 1, 2, 3 или 4 предмета. Проигрывает тот, кто берет последний предмет.

Вариант № 23

1. На ленте машины Тьюринга записано слово, состоящее из букв латинского алфавита $\{a, b, c, d\}$. Подсчитать число вхождений буквы d в заданное слово. Полученное значение записать на ленту левее заданного слова через пробел. Считывающая головка обозревает крайнюю правую букву.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y,z)=x+y+z$.
3. На информационной ленте машины Поста расположено N массивов меток, отделенных друг от друга свободной ячейкой. Каретка находится над крайней левой меткой первого массива. Определите число массивов.

Вариант № 24

1. На ленте машины Тьюринга записано десятичное число. Определить, делится ли это число на 3 без остатка. Если делится, то справа от числа записать слово «да», если не делится, то записать «нет». Считывающая головка находится где-то над числом.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y)=x \dot{-} 1$.
3. На ленте машины Поста расположен массив из N меток (метки расположены через пробел). Надо сжать массив так, чтобы все N меток занимали N расположенных подряд секций.

Вариант № 25

1. На ленте машины Тьюринга записано число в двоичной системе. Постройте машину Тьюринга, которая преобразует запись числа в двоичной системе счисления в восьмеричную.
2. Построить машину Тьюринга, вычисляющую функцию $f(x,y)=x+3$.
3. На ленте машины Поста расположен массив в N отмеченных секциях. Необходимо справа от данного массива через одну пустую секцию разместить массив вдвое больший (он должен состоять из $2N$ меток). При этом исходный массив может быть стерт.

Итоговая контрольная работа №2 «Рекурсивные функции»

Вариант № 1

1. Применить операцию рекурсии к данным функциям
$g(x)=1,\ h(x,y,z)=(y+1)\cdot z$;
2. Применить операцию минимизации к данной функции
$h(x,y,z)=x+y+z$;
3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=x+y$.

Вариант № 2

1. Применить операцию рекурсии к данным функциям
$g(x)=1,\ h(x,y,z)=x+y-z$;
2. Применить операцию минимизации к данной функции
$h(x,y,z)=5+z$;
3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=x^2$.

Вариант № 3

1. Применить операцию рекурсии к данным функциям
$g(x)=x,\ h(x,y,z)=x-y-z$;
2. Применить операцию минимизации к данной функции

$h(x, y, z) = x + z + 1$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x \cdot y$.

Вариант № 4

1. Применить операцию рекурсии к данным функциям $g(x) = 0$, $h(x, y, z) = x + z$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = x + z^2$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = 5x$.

Вариант № 5

1. Применить операцию рекурсии к данным функциям $g(x) = 1$, $h(x, y, z) = x - z$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = 1 - z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x^y$.

Вариант № 6

1. Применить операцию рекурсии к данным функциям $g(x) = x$, $h(x, y, z) = x + z$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = x + y + z - x \cdot y \cdot z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = y!$.

Вариант № 7

1. Применить операцию рекурсии к данным функциям $g(x) = x$, $h(x, y, z) = x - z$;

2. Применить операцию минимизации к данной функции

$h(x,y,z)=\sqrt{z}-1;$

3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=(x+y)^2$.

Вариант № 8

1. Применить операцию рекурсии к данным функциям $g(x)=x,\ h(x,y,z)=y^2-z;$

2. Применить операцию минимизации к данной функции $h(x,y,z)=\frac{1}{z}+1;$

3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=x-y$.

Вариант № 9

1. Применить операцию рекурсии к данным функциям $g(x)=x,\ h(x,y,z)=y^2+z;$

2. Применить операцию минимизации к данной функции $h(x,y,z)=x-y-z;$

3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=-x+y$.

Вариант № 10

1. Применить операцию рекурсии к данным функциям $g(x)=x,\ h(x,y,z)=x\cdot y+z;$

2. Применить операцию минимизации к данной функции $h(x,y,z)=x+y+z^2;$

3. Получить из базовых функций с помощью основных операций функцию $f(x,y)=-x$.

Вариант № 11

1. Применить операцию рекурсии к данным функциям $g(x)=x,\ h(x,y,z)=x\cdot y-z;$

2. Применить операцию минимизации к данной функции

$h(x, y, z) = z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = \frac{1}{x}$.

Вариант № 12

1. Применить операцию рекурсии к данным функциям

$g(x) = x$, $h(x, y, z) = z^{y+1}$;

2. Применить операцию минимизации к данной функции

$h(x, y, z) = -z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = \frac{x}{y}$.

Вариант № 13

1. Применить операцию рекурсии к данным функциям

$g(x) = 1$, $h(x, y, z) = x \cdot (y + 1) \cdot z$;

2. Применить операцию минимизации к данной функции

$h(x, y, z) = x + y + z + \frac{1}{x + y}$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x^x$.

Вариант № 14

1. Применить операцию рекурсии к данным функциям

$g(x) = x^2$, $h(x, y, z) = x^2 + z$;

2. Применить операцию минимизации к данной функции

$h(x, y, z) = x + y + z + \frac{1}{z}$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = 2^x$.

Вариант № 15

1. Применить операцию рекурсии к данным функциям

$g(x) = x,\ h(x, y, z) = x \cdot (y + 2) \cdot z;$

2. Применить операцию минимизации к данной функции

$h(x, y, z) = -x + y + z - 1;$

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = \sqrt{x}$.

Вариант № 16

1. Применить операцию рекурсии к данным функциям

$g(x) = x,\ h(x, y, z) = x + y;$

2. Применить операцию минимизации к данной функции

$h(x, y, z) = z^2 + y + 1;$

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = \dfrac{1}{x + y}$.

Вариант № 17

1. Применить операцию рекурсии к данным функциям

$g(x) = x,\ h(x, y, z) = x + y + z;$

2. Применить операцию минимизации к данной функции

$h(x, y, z) = x(z + y) + z;$

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x + 1$.

Вариант № 18

1. Применить операцию рекурсии к данным функциям

$g(x) = x,\ h(x, y, z) = x + z;$

2. Применить операцию минимизации к данной функции

$h(x, y, z) = x \cdot y + z + 1;$

3. Получить из базовых функций с помощью основных операций

функцию $f(x,y)=\frac{y}{x}+1$.

Вариант № 19

1. Применить операцию рекурсии к данным функциям

$g(x)=x+1,\ h(x,y,z)=x+z$;

2. Применить операцию минимизации к данной функции

$h(x,y,z)=x\cdot y\cdot z+1$;

3. Получить из базовых функций с помощью основных операций

функцию $f(x,y)=\frac{1}{y-x}$.

Вариант № 20

1. Применить операцию рекурсии к данным функциям

$g(x,y)=1,\ h(x,y,z,k)=x\cdot z+y\cdot k$;

2. Применить операцию минимизации к данной функции

$h(x,y,z)=x-y\cdot z$;

3. Получить из базовых функций с помощью основных операций

функцию $f(x,y)=\frac{x+y}{x-y}$.

Вариант № 21

1. Применить операцию рекурсии к данным функциям

$g(x)=x,\ h(x,y,z)=x\cdot y+z$;

2. Применить операцию минимизации к данной функции

$h(x,y,z)=x+y\cdot z$;

3. Получить из базовых функций с помощью основных операций

функцию $f(x,y)=x+y+1$.

Вариант № 22

1. Применить операцию рекурсии к данным функциям

$g(x)=x^2,\ h(x,y,z)=y+z$;

2. Применить операцию минимизации к данной функции

$h(x, y, z) = x \cdot y - z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x + y + x \cdot y$.

Вариант № 23

1. Применить операцию рекурсии к данным функциям $g(x) = 1,\ h(x, y, z) = x \cdot y \cdot z - x - y - z$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = x \cdot y + z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = 5(x + y)$.

Вариант № 24

1. Применить операцию рекурсии к данным функциям $g(x) = 1,\ h(x, y, z) = x + y + z - 1$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = x - y - z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x + y - 1$.

Вариант № 25

1. Применить операцию рекурсии к данным функциям $g(x) = 1,\ h(x, y, z) = x + y + \frac{x}{y}$;

2. Применить операцию минимизации к данной функции $h(x, y, z) = x + y - z$;

3. Получить из базовых функций с помощью основных операций функцию $f(x, y) = x! + y!$.

Список тем курсовых и выпускных работ и методические рекомендации для их выполнения

Тема 1. Метод диагонализации в математической логике

В математической логике, теории множеств и других разделах математики широко применяется метод диагонализации, в основе которого лежит возможность построения антидиагонального счетного множества для любой последовательности счетных множеств. В курсовой работе необходимо изучить метод диагонализации и с его помощью построить примеры невычислимых функций. Рекомендуется следующий план работы:

- рассмотреть понятие счетного множества и изучить метод диагонализации (/1/, с. 12-30).
- рассмотреть понятие машины Тьюринга и методом диагонализации построить пример невычислимой функции (/1/, с. 36-45, 66-74).
- рассмотреть проблему остановки машины Тьюринга и с помощью тезиса Черча доказать ее неразрешимость (/1/, с. 47-48, 74-76).
-рассмотреть понятие диагонализации выражения и доказать лемму о диагонализации и теорему Черча о неразрешимости (/1/, с. 228-235).
- разобрать решения всех примеров из цитированных разделов /1/ и решить задачи 3.9, 3.10 из упражнения на стр. 45-48 и задачи 5.1-5.4 из упражнения на стр. 76-77 в книге /1/.

Литература, рекомендуемая для изучения темы:

1. Булос Дж., Джеффри Р. Вычислимость и логика. - М.: Мир, 1994.

Тема 2. Машины Тьюринга и невычислимые функции

Машина Тьюринга и вычислимость являются фундаментальными понятиями математической логики. В курсовой работе необходимо изучить основные свойства машины Тьюринга и с ее помощью построить невычислимую функцию. Рекомендуется следующий план работы.

- разобрать такие основополагающие понятия математической логики, как машина Тьюринга, вычислимая функция и тезис Черча (/1/, с. 36-54; /2/, с. 228-229, 249-255).

- рассмотреть понятие продуктивности машины Тьюринга и доказать ее основные свойства (/1/, с. 46, 55-60; /2/, с. 12-25).

- доказать невычислимость функции продуктивность машины Тьюринга (/1/, с. 60-64).

-рассмотреть проблему остановки машины Тьюринга и доказать ее неразрешимость (/1/, с. 47-48, 53-54, 64-65).

Разобрать решения всех примеров из литературных источников /1/,/2/ и решить задачи 3.1-3.10 из упражнения на стр. 45-48 в книге /1/.

Литература, рекомендуемая для изучения темы:

1. Булос Дж., Джеффри Р. Вычислимость и логика. - М.: Мир, 1994.

2. Мендельсон Э. Введение в математическую логику. - М.: Наука, 1971.

Тема 3. Вычислимость на абаке и рекурсивные функции

Рекурсивная функция и вычислимость являются фундаментальными понятиями математической логики. В курсовой работе необходимо изучить вычислимость на абаке, вычислимость машиной Тьюринга и доказать их эквивалентность понятию рекурсивной функции. Рекомендуется следующий план работы:

- разобрать такие основополагающие понятия математической логики, как машина Тьюринга, рекурсивная функция и тезис Черча (/1/, с. 36-54).

- рассмотреть понятие «обычного» компьютера, введенное Иоахимом Ламбеком и названное им абаком, доказать, что вычислимость функции абаком сводится к вычислимости ее машиной Тьюринга (/1/, с. 78-95).

- доказать, что рекурсивные функции вычислимы на абаках (/1/, с. 100122).

- доказать, что вычислимые функции рекурсивны (/1/, с. 100-122).

- разобрать решения всех примеров из цитированных разделов книги /1/ и решить задачи 6.1-6.4 из упражнения на стр. 96 в книге /1/.

Литература, рекомендуемая для изучения темы:

1. Булос Дж., Джеффри Р. Вычислимость и логика. - М.: Мир, 1994.

Тема 4. Представимость рекурсивных функций и отрицательные результаты математической логики

Главными отрицательными результатами математической логики являются теорема Черча о неразрешимости логики, теорема Тарского о неопределимости истинности и первая теорема Геделя о неполноте систем арифметики. В курсовой работе необходимо изучить доказательства этих теорем с помощью представления рекурсивных функций в специальном расширении арифметики. Рекомендуется следующий план работы:

- разобрать такие основополагающие понятия математической логики, как язык арифметики и рекурсивная функция (/1/, с. 103-108, 141-145).
- рассмотреть понятие представимости функций в теории и доказать представимость рекурсивных функций в специальном непротиворечивом расширении арифметики (/1/, с. 212-226).
- рассмотреть понятие геделевой нумерации и доказать главные отрицательные результаты математической логики (/1/, с. 228-240).
- разобрать решения всех примеров из цитированных разделов книги /1/ и решить задачи 14.1-14.2 из упражнения на стр. 226-227 и задачи 15.1-15.4 из упражнения на стр. 240 в книге /1/.

Литература, рекомендуемая для изучения темы

1. Булос Дж., Джеффри Р. Вычислимость и логика. - М.: Мир, 1994.

Тема 5. Теорема Геделя о неполноте формальной арифметики

Теорема Геделя о неполноте формальной арифметики по праву считается одним из наиболее замечательных достижений математической логики, поскольку в своей семантической формулировке устанавливает невозможность доказательства любого истинного утверждения этой формальной теории. В курсовой работе необходимо изучить основы формальной арифметики и разобрать доказательство семантической формулировки теоремы Геделя о ее неполноте. Рекомендуется следующий план работы:

- изучить постановку задачи о неполноте формальной арифметики (/1/, с. 7-11).

- рассмотреть начальные понятия теории алгоритмов и примеры их применения (/1/, с. 12-21).

- доказать простейшие критерии неполноты (/1/, с. 21-25).

-изучить основы формальной арифметики и доказать семантическую формулировку теоремы Геделя о ее неполноте (/1/, с. 25-42).

Литература, рекомендуемая для изучения темы

1. Успенский В.А. Теорема Геделя о неполноте. - М.: Наука, 1982.

Библиографический список

1. Абрамов С. А. Математические построения и программирование. – М.: Наука, 1978.

2. Айзерман М.А., Гусев Л.А., Розоноэр Л.И.. Смирнова И.М., Таль А.А. Логика. Автоматы. Алгоритмы. – М.: ФИЗМАТЛИТ, 1963.

3. Алферова З.В. Теория алгоритмов. – М.: Статистика, 1973.

4. Болибрух А.А. Проблемы Гильберта (100 лет спустя). – М.: МЦНМО, 1999.

5. Булос Дж., Джеффри Р. Вычислимость и логика. – М.: Мир, 1994.

6. Верещагин Н.К., Шень А. Вычислимые функции. – М.: МЦНМО, 2002.

7. Гаврилов Г.П., Сапоженко А.А. Задачи и упражнения по курсу дискретной математики. – М.: Наука, 1992.

8. Галиев Ш.И. Математическая логика и теория алгоритмов. – Казань: КГТУ, 2002.

9. Гиндикин С.Г. Алгебра логики в задачах. – М.: Наука, 1972.

10. Головешкин В.А., Ульянов М.В. Теория рекурсий для программистов. – М.: ФИЗМАТЛИТ, 2006.

11. Гуц А.К. Математическая логика и теория алгоритмов. – Омск: Наследие, 2003.

12. Дискретная математика и математические вопросы кибернетики. Под ред. С.В. Яблонского. – М.: Наука, 1974.

13. Донской В.И. Дискретная математика. – Симферополь: Сонат, 2000.
14. Ерусалимский Я.М. Дискретная математика: теория, задачи, упражнения. – М.: Вузовская книга, 2000.
15. Ершов А. П. Алгоритмический язык. – Наука и жизнь. – 1985. – №11.
16. Заварыкин В.М., Житомирский В.Г., Лапчик М.П. Техника вычислений и алгоритмизация. – М.: Просвещение, 1987.
17. Зарисовки по истории компьютерных наук : учебное пособие : в
3 ч. / В.П.Одинец. – Сыктывкар: Коми пединститут, 2011.- Ч.1
18. Игошин В.И. Задачник-практикум по математической логике и теории алгоритмов. – М.: Просвещение, 1986.
19. Игошин В.И. Математическая логика и теория алгоритмов. – Саратов: Сарат. ун-та, 1991.
20. Идельсон А.В, Минц Г.Е. Математическая теория логического вывода. – М.: Наука, 1967.
21. Катленд Н. Вычислимость. Введение в теорию рекурсивных функций. – М.: Мир, 1983.
22. Колмогоров А.Н. Теория информации и теория алгоритмов. – М.: Наука, 1987.
23. Колмогоров А.Н., Драгалин А.Г. Математическая логика. Дополнительные главы. – М.: Моск.ун-та, 1984.
24. Клини С.К. Введение в метаматематику. – М.: ИЛ, 1957.
25. Клини С.К., Весли Р.И. Основания интуиционистской математики с точки зрения теории рекурсивных функций. – М.: Наука, 1978.
26. Клини С.К. Математическая логика. – М.: Мир, 1973.
27. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теорем алгоритмов. – М.: Наука, 1984.
28. Лапчик М.П. Вычисления. Алгоритмизация. Программирование. – М.: Просвещение, 1988.
29. Лапчик М.П. Вычислительная техника и программирование. – М.: Просвещение, 1987.
30. Лихтарников Л.М., Сукачева Т.Г. Математическая логика. – СПб: Лань, 1998.

31. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: Наука, 1986.
32. Марков А.А., Нагорный Н.М. Теория алгорифмов. – М.: ФАЗИС, 1996. -448с.
33. Матросов В.Л. Теория алгоритмов. - М.: Прометей, 1989.
34. Мендельсон Э. Введение в математическую логику. – М.: Наука, 1971.
35. Михайлов А.Б., Рыжова Н.И., Швецкий М.В. Лекции по основам математической логики. Элементы теории алгорифмов. – СПб.: РГПУ им. А.И.Герцена, 1997.
36. Никольская И.Л. Математическая логика. – М.: Высшая школа, 1981.
37. Новиков П.С. Конструктивная математическая логика с точки зрения классической. – М.: Наука, 1977.
38. Носов В.А. основы теории алгоритмов и анализа их сложности. – М.: МГУ, 1992.
39. Основы информатики и вычислительной техники. Пробное учебное пособие для средних учебных заведений. Ч. 1. Под ред. А. П. Ершова и В. М. Монахова. – М: Просвещение, 1985.
40. Петер Р. Рекурсивные функции. – М.: ИЛ, 1954.
41. Роджерс Х. Теория рекурсивных функций и эффективная вычислимость. – М.: Мир, 1972.
42. Сборник тем курсовых работ по математике (алгебра, математическая логика, дискретная математика)/ В.А. Молчанов, В.Е. Новиков, Т.М. Отрыванкина, П.Н. Пронин, В.Е. Фирстов. – Оренбург: ГОУ ОГУ, 2004.
43. Справочная книга по математической логике: В 4-х частях/Под ред. Дж. Барвайса. – М.: Наука. ФИЗМАТЛИТ, 1982.
44. Трахтенброт Б.А. Алгоритмы и вычислительные автоматы. – М.: Сов. радио, 1974.
45. Успенский В.А. Теорема Геделя о неполноте. – М.: Наука, 1982.
46. Успенский В.А. Машина Поста. – М.: Наука, 1979.
47. Успенский В.А., Смирнов А.Л. Теория алгоритмов: основные открытия и приложения. – М.: Наука, 1987.
48. Шарова Л.И. Уравнения и неравенства. Пособие для подготовительных отделений. – Киев: Вища школа, 1981.

49. Шенфилд Дж. Математическая логика. – М.: Наука, 1975.
50. Яблонский С.В. Введение в дискретную математику. – М.: Наука, 1986.
51. Яглом И.М. Необыкновенная алгебра. – М.: Наука, 1968.